

Outline

What can we optimize?

Rule-based optimization

Data statistics

Cost models

Cost-based plan selection

What Are Data Statistics?

Information about the tuples in a relation that can be used to estimate size & cost

- » Example: # of tuples, average size of tuples, # distinct values for each attribute, % of null values for each attribute

Typically maintained by the storage engine as tuples are added & removed in a relation

- » File formats like Parquet can also have them

Some Statistics We'll Use

For a relation R ,

$T(R)$ = # of tuples in R

$S(R)$ = average size of R 's tuples in bytes

$B(R)$ = # of blocks to hold all of R 's tuples

$V(R, A)$ = # distinct values of attribute A in R

Example

R:

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

A: 20 byte string

B: 4 byte integer

C: 8 byte date

D: 5 byte string

Example

R:

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

A: 20 byte string

B: 4 byte integer

C: 8 byte date

D: 5 byte string

$$T(R) = 5$$

$$V(R, A) = 3$$

$$V(R, B) = 1$$

$$S(R) = 37$$

$$V(R, C) = 5$$

$$V(R, D) = 4$$

Challenge: Intermediate Tables

Keeping stats for tables on disk is easy, but what about intermediate tables that appear during a query plan?

Examples:

$\sigma_p(R)$ ← We already have $T(R)$, $S(R)$, $V(R, a)$, etc, but how to get these for tuples that pass p ?

$R \bowtie S$ ← How many and what types of tuple pass the join condition?

Should we do $(R \bowtie S) \bowtie T$ or $R \bowtie (S \bowtie T)$ or $(R \bowtie T) \bowtie S$?

Stat Estimation Methods

Algorithms to estimate subplan stats

An ideal algorithm would have:

- 1) Accurate estimates of stats
- 2) Low cost
- 3) Consistent estimates (e.g. different plans for a subtree give same estimated stats)

Can't always get all this!

Size Estimates for $W = R_1 \times R_2$

$$S(W) =$$

$$T(W) =$$

Size Estimates for $W = R_1 \times R_2$

$$S(W) = S(R_1) + S(R_2)$$

$$T(W) = T(R_1) \times T(R_2)$$

Size Estimate for $W = \sigma_{A=a}(R)$

$$S(W) =$$

$$T(W) =$$

Size Estimate for $W = \sigma_{A=a}(R)$

$S(W) = S(R)$ ← Not true if some variable-length fields are correlated with value of A

$T(W) =$

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$

$$W = \sigma_{Z=val}(R) \quad T(W) =$$

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$

what is probability this
tuple will be in answer?

$$W = \sigma_{Z=val}(R) \quad T(W) =$$

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$

$$W = \sigma_{Z=val}(R)$$

$$T(W) = \frac{T(R)}{V(R,Z)}$$

Assumption:

Values in select expression $Z=val$ are **uniformly distributed** over all $V(R, Z)$ values

Alternate Assumption:

Values in select expression $Z=val$ are **uniformly distributed** over a domain with $DOM(R, Z)$ values

Example

R	A	B	C	D
	cat	1	10	a
	cat	1	20	b
	dog	1	30	a
	dog	1	40	c
	bat	1	50	d

Alternate assumption

$$V(R,A)=3, \text{ DOM}(R,A)=10$$

$$V(R,B)=1, \text{ DOM}(R,B)=10$$

$$V(R,C)=5, \text{ DOM}(R,C)=10$$

$$V(R,D)=4, \text{ DOM}(R,D)=10$$

$$W = \sigma_{Z=\text{val}}(R) \quad T(W) =$$

Example

R

A	B	C	D
cat	1	10	a
cat	1	20	b
dog	1	30	a
dog	1	40	c
bat	1	50	d

Alternate assumption

$V(R,A)=3$, $DOM(R,A)=10$

$V(R,B)=1$, $DOM(R,B)=10$

$V(R,C)=5$, $DOM(R,C)=10$

$V(R,D)=4$, $DOM(R,D)=10$

what is probability this
tuple will be in answer?

$$W = \sigma_{Z=val}(R) \quad T(W) =$$

Example

R	A	B	C	D
	cat	1	10	a
	cat	1	20	b
	dog	1	30	a
	dog	1	40	c
	bat	1	50	d

Alternate assumption

$$V(R,A)=3, \text{ DOM}(R,A)=10$$

$$V(R,B)=1, \text{ DOM}(R,B)=10$$

$$V(R,C)=5, \text{ DOM}(R,C)=10$$

$$V(R,D)=4, \text{ DOM}(R,D)=10$$

$$W = \sigma_{Z=\text{val}}(R)$$

$$T(W) = \frac{T(R)}{\text{DOM}(R,Z)}$$

Selection Cardinality

$SC(R, A)$ = average # records that satisfy equality condition on R.A

$$SC(R, A) = \left\{ \begin{array}{l} \frac{T(R)}{V(R, A)} \\ \frac{T(R)}{DOM(R, A)} \end{array} \right.$$

What About $W = \sigma_{z \geq \text{val}}(R)$?



$T(W) = ?$

What About $W = \sigma_{z \geq \text{val}}(R)$?

$T(W) = ?$

Solution 1: $T(W) = T(R) / 2$

What About $W = \sigma_{z \geq \text{val}}(R)$?

$T(W) = ?$

Solution 1: $T(W) = T(R) / 2$

Solution 2: $T(W) = T(R) / 3$

Solution 3: Estimate Fraction of Values in Range

Example: R

	Z

Min=1

$V(R,Z)=10$



$W = \sigma_{Z \geq 15}(R)$

Max=20

$$f = \frac{20-15+1}{20-1+1} = \frac{6}{20} \quad (\text{fraction of range})$$

$$T(W) = f \times T(R)$$

Solution 3: Estimate Fraction of Values in Range

Equivalently, if we know values in column:

f = fraction of distinct values $\geq \text{val}$

$$T(W) = f \times T(R)$$

Size Estimate for $W = R_1 \bowtie R_2$

Let X = attributes of R_1

Y = attributes of R_2

Case 1: $X \cap Y = \emptyset$:

Same as $R_1 \times R_2$

Case 2: $W = R_1 \bowtie R_2, X \cap Y = A$

R_1	A	B	C

R_2	A	D

Case 2: $W = R_1 \bowtie R_2, X \cap Y = A$

R_1	A	B	C

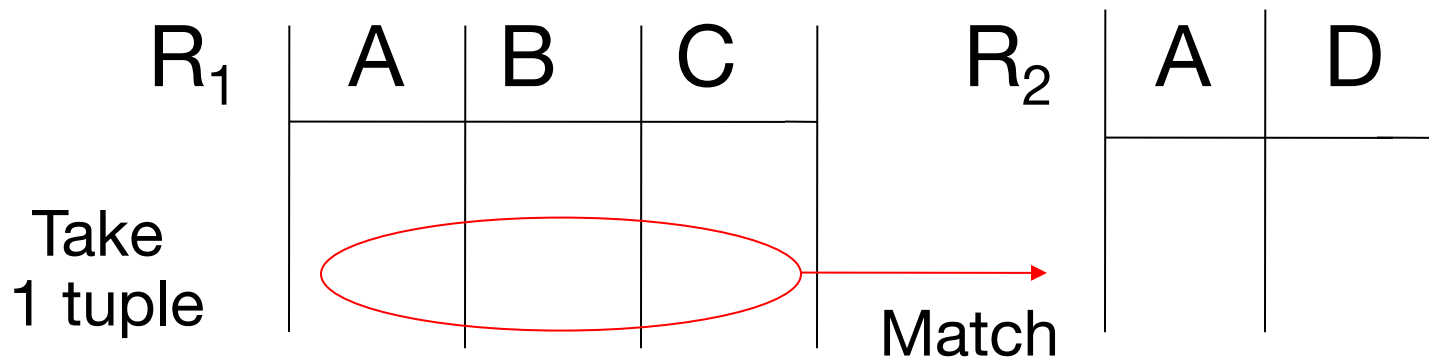
R_2	A	D

Assumption (“containment of value sets”):

$V(R_1, A) \leq V(R_2, A) \Rightarrow$ Every A value in R_1 is in R_2

$V(R_2, A) \leq V(R_1, A) \Rightarrow$ Every A value in R_2 is in R_1

Computing $T(W)$ when $V(R_1, A) \leq V(R_2, A)$



1 tuple matches with $\frac{T(R_2)}{V(R_2, A)}$ tuples...

so
$$T(W) = \frac{T(R_1) \times T(R_2)}{V(R_2, A)}$$

$$V(R_1, A) \leq V(R_2, A) \Rightarrow T(W) = \frac{T(R_1) \times T(R_2)}{V(R_2, A)}$$

$$V(R_2, A) \leq V(R_1, A) \Rightarrow T(W) = \frac{T(R_1) \times T(R_2)}{V(R_1, A)}$$

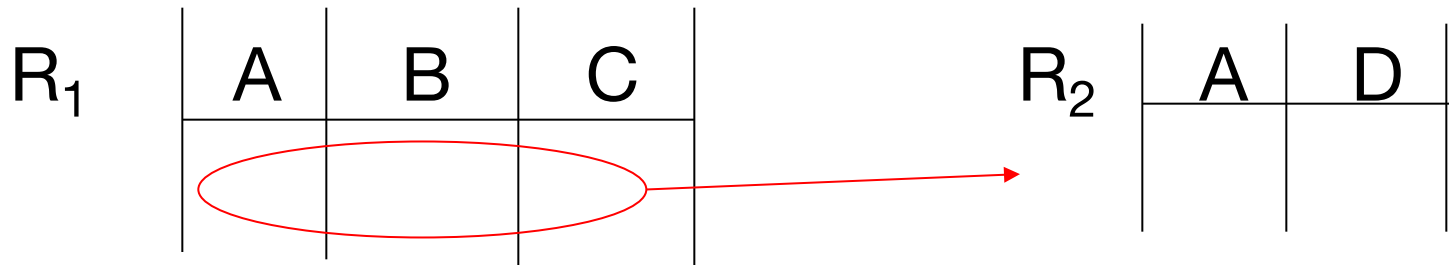
In General for $W = R_1 \bowtie R_2$

$$T(W) = \frac{T(R_1) \times T(R_2)}{\max(V(R_1, A), V(R_2, A))}$$

Where A is the common attribute set

Case 2 with Alternate Assumption

Values uniformly distributed over domain



This tuple matches $T(R_2) / \text{DOM}(R_2, A)$, so

$$T(W) = \frac{T(R_1) T(R_2)}{\text{DOM}(R_2, A)} = \frac{T(R_1) T(R_2)}{\text{DOM}(R_1, A)}$$

Assume these are the same

Tuple Size after Join

In all cases:

$$S(W) = S(R_1) + S(R_2) - S(A)$$

size of attribute A



Using Similar Ideas, Can Estimate Sizes of:

$$\Pi_{A,B}(R)$$

$$\sigma_{A=a \wedge B=b}(R)$$

$R \bowtie S$ with common attributes A, B, C

Set union, intersection, difference, ...

For Complex Expressions, Need Intermediate T, S, V Results

E.g. $W = \underbrace{\sigma_{A=a}(R_1)} \bowtie R_2$

Treat as relation U

$$T(U) = T(R_1) / V(R_1, A) \qquad S(U) = S(R_1)$$

Also need $V(U, *)$!!

To Estimate V

E.g., $U = \sigma_{A=a}(R_1)$

Say R_1 has attributes A, B, C, D

$$V(U, A) =$$

$$V(U, B) =$$

$$V(U, C) =$$

$$V(U, D) =$$

Example

R_1

A	B	C	D
cat	1	10	10
cat	1	20	20
dog	1	30	10
dog	1	40	30
bat	1	50	10

$$V(R_1, A)=3$$

$$V(R_1, B)=1$$

$$V(R_1, C)=5$$

$$V(R_1, D)=3$$

$$U = \sigma_{A=a}(R_1)$$

Example

R_1

A	B	C	D
cat	1	10	10
cat	1	20	20
dog	1	30	10
dog	1	40	30
bat	1	50	10

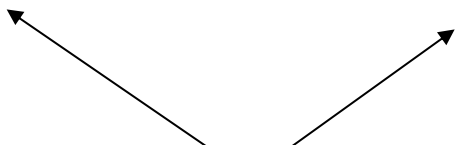
$$V(R_1, A)=3$$

$$V(R_1, B)=1$$

$$V(R_1, C)=5$$

$$V(R_1, D)=3$$

$$U = \sigma_{A=a}(R_1)$$

$$V(U, A) = 1 \quad V(U, B) = 1 \quad V(U, C) = \frac{T(R_1)}{V(R_1, A)}$$


$V(U, D) = \text{somewhere in between...}$

Possible Guess in $U = \sigma_{A \geq a}(R)$

$$V(U, A) = V(R, A) / 2$$

$$V(U, B) = V(R, B)$$

For Joins: $U = R_1(A,B) \bowtie R_2(A,C)$

We'll use the following estimates:

$$V(U, A) = \min(V(R_1, A), V(R_2, A))$$

$$V(U, B) = V(R_1, B)$$

$$V(U, C) = V(R_2, C)$$

Called “preservation of value sets”

Example:

$$Z = R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,D)$$

R_1

$$T(R_1) = 1000 \quad V(R_1,A)=50 \quad V(R_1,B)=100$$

R_2

$$T(R_2) = 2000 \quad V(R_2,B)=200 \quad V(R_2,C)=300$$

R_3

$$T(R_3) = 3000 \quad V(R_3,C)=90 \quad V(R_3,D)=500$$

Partial Result: $U = R_1 \bowtie R_2$

$$T(U) = \frac{1000 \times 2000}{200}$$

$$V(U,A) = 50$$

$$V(U,B) = 100$$

$$V(U,C) = 300$$

End Result: $Z = U \bowtie R_3$

$$T(Z) = \frac{1000 \times 2000 \times 3000}{200 \times 300}$$

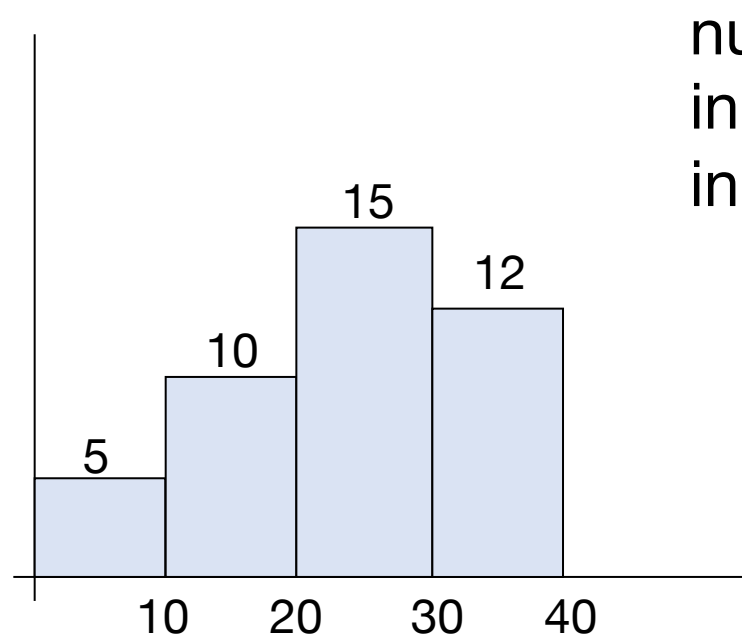
$$V(Z,A) = 50$$

$$V(Z,B) = 100$$

$$V(Z,C) = 90$$

$$V(Z,D) = 500$$

Another Statistic: Histograms



number of tuples
in R with A value
in a given range

$$\sigma_{A \geq a}(R) = ?$$

$$\sigma_{A = a}(R) = ?$$

Requires some care to set bucket boundaries

Outline

What can we optimize?

Rule-based optimization

Data statistics

Cost models

Cost-based plan selection

Cost Models

How do we measure a query plan's cost?

Many possible metrics:

» Number of disk I/Os

← We'll focus on this

» Number of compute cycles

» Combined time metric

» Memory usage

» Bytes sent on network

» ...

Example: Index vs Table Scan

Our query: $\sigma_p(R)$ for some predicate p

$s = p$'s selectivity (fraction tuples passing)

Table scan:

block size



R has $B(R) = T(R) \times S(R) / b$
blocks on disk

Cost: $B(R)$ I/Os

Index search:

Index lookup for p takes L I/Os

We then have to read part of R ;

$\Pr[\text{read block } i]$

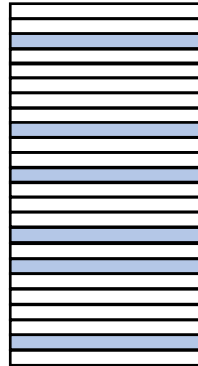
$\approx 1 - \Pr[\text{no match}]^{\text{records in block}}$

$= 1 - (1-s)^{b / S(R)}$

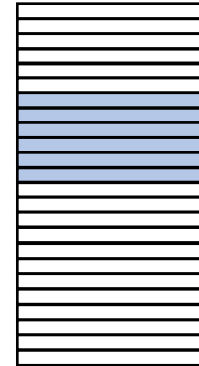
Cost: $L + (1-(1-s)^{b/S(R)}) B(R)$

What If Results Were Clustered?

Unclustered:
records that
match p are
spread out
uniformly



Clustered:
records that
match p are
close together
in R 's file



We'd need to change our estimate of C_{index} :

$$C_{\text{index}} = L + \underbrace{s}_{\text{Fraction of } R\text{'s blocks read}} B(R)$$

Less than C_{index} for
unclustered data

Join Operators

Join **orders** and **algorithms** are often the choices that affect performance the most

For a multi-way join $R \bowtie S \bowtie T \bowtie \dots$, each join is selective, and order matters a lot

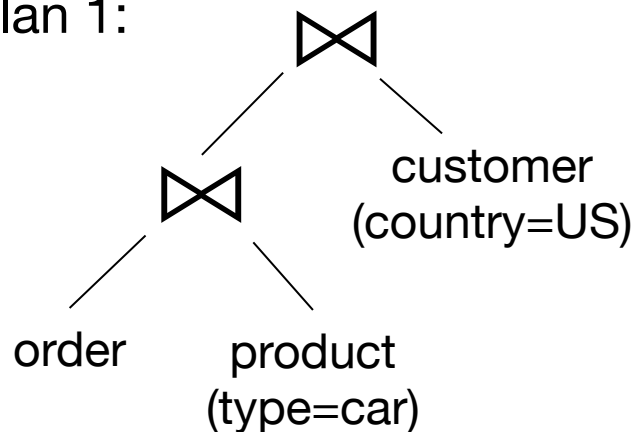
- » Try to eliminate lots of records early

Even for one join $R \bowtie S$, algorithm matters

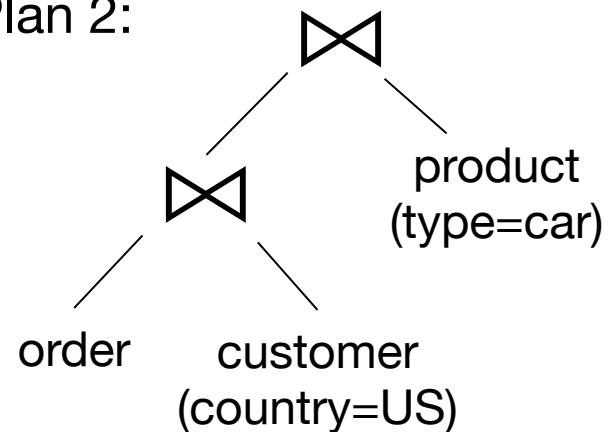
Example

```
SELECT order.date, product.price, customer.name
FROM order, product, customer
WHERE order.product_id = product.product_id } join conditions
      AND order.cust_id = customer.cust_id   }
      AND product.type = "car"              } selection predicates
      AND customer.country = "US"
```

Plan 1:



Plan 2:



When is each plan better?

Common Join Algorithms

Iteration (nested loops) join

Merge join

Join with index

Hash join

Iteration Join

```
for each  $r \in R_1$ :  
  for each  $s \in R_2$ :  
    if  $r.C == s.C$  then output  $(r, s)$ 
```

I/Os: one scan of R_1 and $T(R_1)$ scans of R_2 , so
cost = $B(R_1) + T(R_1) B(R_2)$ reads

Improvement: read M **blocks** of R_1 in RAM at
a time then read R_2 : **$B(R_1) + B(R_1) B(R_2) / M$**

Note: cost of writes is always $B(R_1 \bowtie R_2)$

Merge Join

```
if  $R_1$  and  $R_2$  not sorted by  $C$  then sort them  
 $i, j = 1$   
while  $i \leq T(R_1) \ \&\& \ j \leq T(R_2)$ :  
    if  $R_1[i].C = R_2[j].C$  then outputTuples  
    else if  $R_1[i].C > R_2[j].C$  then  $j += 1$   
    else if  $R_1[i].C < R_2[j].C$  then  $i += 1$ 
```

Query Optimization

Merge Join

```
procedure outputTuples:
  while  $R_1[i].C == R_2[j].C \ \&\& \ i \leq T(R_1)$ :
    jj = j
    while  $R_1[i].C == R_2[jj].C \ \&\& \ jj \leq T(R_2)$ :
      output ( $R_1[i]$ ,  $R_2[jj]$ )
      jj += 1
    i += 1
```

Example

i	$R_1[i].C$	$R_2[j].C$	j
1	10	5	1
2	20	20	2
3	20	20	3
4	30	30	4
5	40	30	5
		50	6
		52	7

Cost of Merge Join

If R_1 and R_2 already sorted by C , then

cost = $B(R_1) + B(R_2)$ reads

(+ write cost of $B(R_1 \bowtie R_2)$)

Cost of Merge Join

If R_i is not sorted, can sort it in $4 B(R_i)$ I/Os:

- » Read runs of tuples into memory, sort
- » Write each sorted run to disk
- » Read from all sorted runs to merge
- » Write out results

Join with Index

```
for each  $r \in R_1$ :  
    list = index_lookup( $R_2$ , C,  $r.C$ )  
    for each  $s \in \text{list}$ :  
        output ( $r$ ,  $s$ )
```

Read I/Os: 1 scan of R_1 , $T(R_1)$ index lookups on R_2 , and $T(R_1)$ data lookups

$$\text{cost} = B(R_1) + T(R_1) (L_{\text{index}} + L_{\text{data}})$$

Can be less when R_1 is sorted/clustered by C!

Hash Join (R_2 Fits in RAM)

```
hash = load  $R_2$  into RAM and hash by C
for each  $r \in R_1$ :
    list = hash_lookup(hash,  $r.C$ )
    for each  $s \in \text{list}$ :
        output ( $r, s$ )
```

Read I/Os: $B(R_1) + B(R_2)$

Hash Join on Disk

Can be done by hashing both tables to a common set of buckets on disk

» Similar to merge sort: $4 (B(R_1) + B(R_2))$

Trick: hash only (key, pointer to record) pairs

» Can then sort the pointers to records that match and fetch them near-sequentially

Other Concerns

Join selectivity may affect how many records we need to fetch from each relation

- » If very selective, may prefer methods that join pointers or do index lookups

Summary

Join algorithms can have different performance in different situations

In general, the following are used:

- » Index join if an index exists
- » Merge join if at least one table is sorted
- » Hash join if both tables unsorted

Outline

What can we optimize?

Rule-based optimization

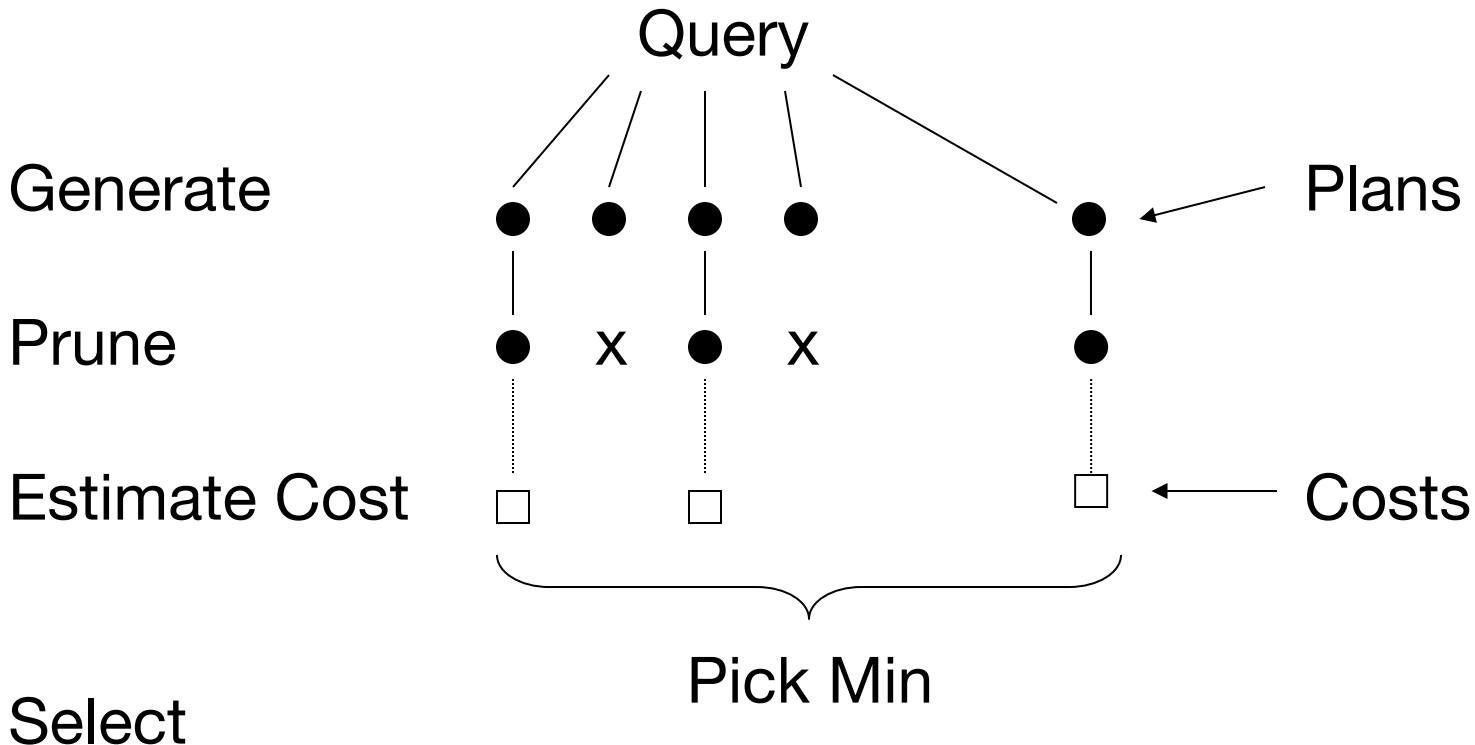
Data statistics

Cost models

Cost-based plan selection

Complete CBO Process

Generate and compare possible query plans



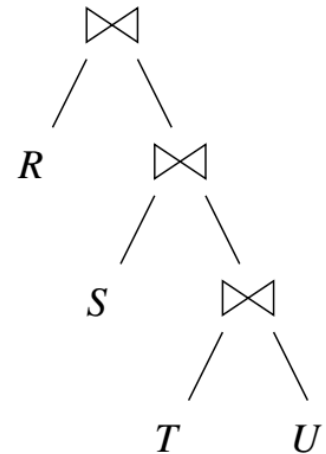
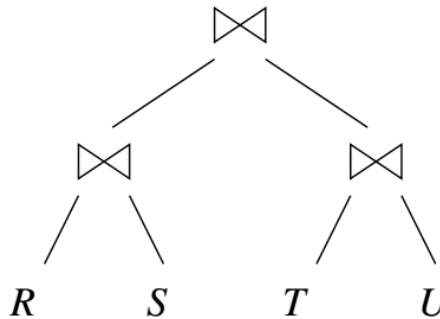
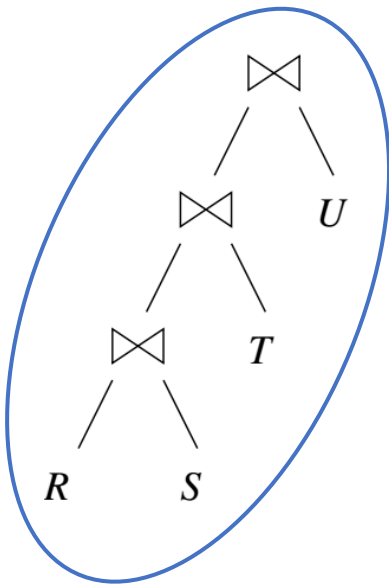
How to Generate Plans?

Simplest way: recursive search of the options for each planning choice

Access paths for table 1 × Access paths for table 2 × Algorithms for join 1 × Algorithms for join 2 × ...

How to Generate Plans?

Can limit search space: e.g. many DBMSes only consider “left-deep” joins



Often interacts well with conventions for specifying join inputs in asymmetric join algorithms (e.g. assume right argument has index)

How to Generate Plans?

Can prioritize searching through the most impactful decisions first

» E.g. join order is one of the most impactful

How to Prune Plans?

While computing the cost of a plan, throw it away if it is worse than best so far

Start with a **greedy algorithm** to find an “OK” initial plan that will allow lots of pruning

Memoization and Dynamic Programming

During a search through plans, many subplans will appear repeatedly

Remember cost estimates and statistics ($T(R)$, $V(R, A)$, etc) for those: “memoization”

Can pick an order of subproblems to make it easy to reuse results (dynamic programming)

Resource Cost of CBO

It's possible for cost-based optimization itself to take longer than running the query!

Need to design optimizer to not take too long

- » That's why we have shortcuts in stats, etc

Luckily, a few “big” decisions drive most of the query execution time (e.g. join order)