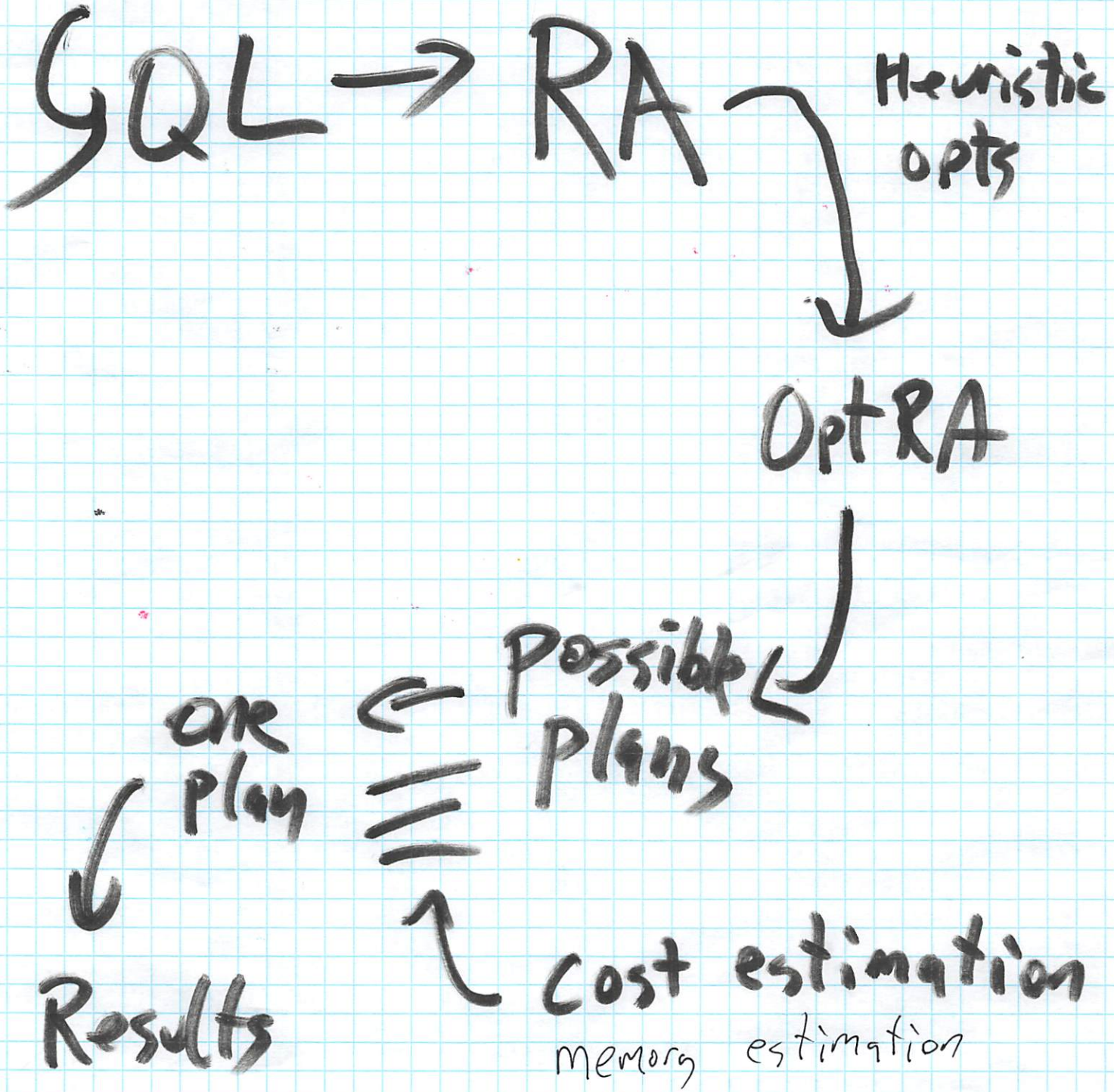
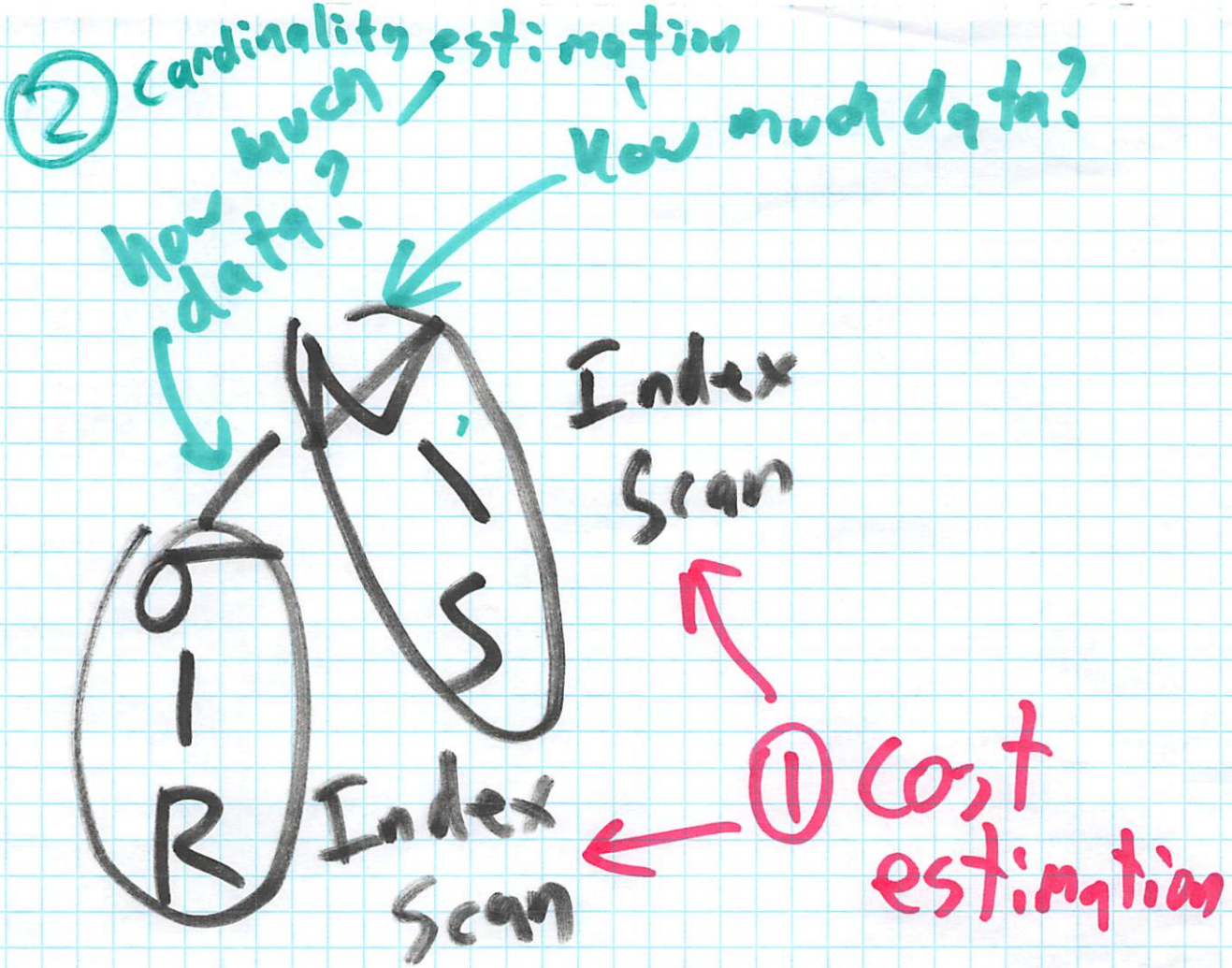

GQL → RA
Heuristic opts
OptRA
Possible plans
one plan
Results
cost estimation
memory estimation



RMS costs

- 1p H Join
- 1p I Join
- Sort Merge Join
- INLJoin
- 2p H Join

agg →

IOs

○

○

Merge: 0 + Sort: |R| log |R| + |S| log |S|

← can skip if R(or S) is already sorted

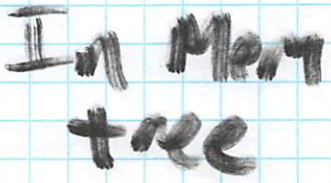
either #records R · 1
or #records R · log |S|

2(|S| + |R|)

1p HJ

HashTable t
for s in S:
    add s to t
for r in R:
    find matches for r in t:
        emit(match or r)

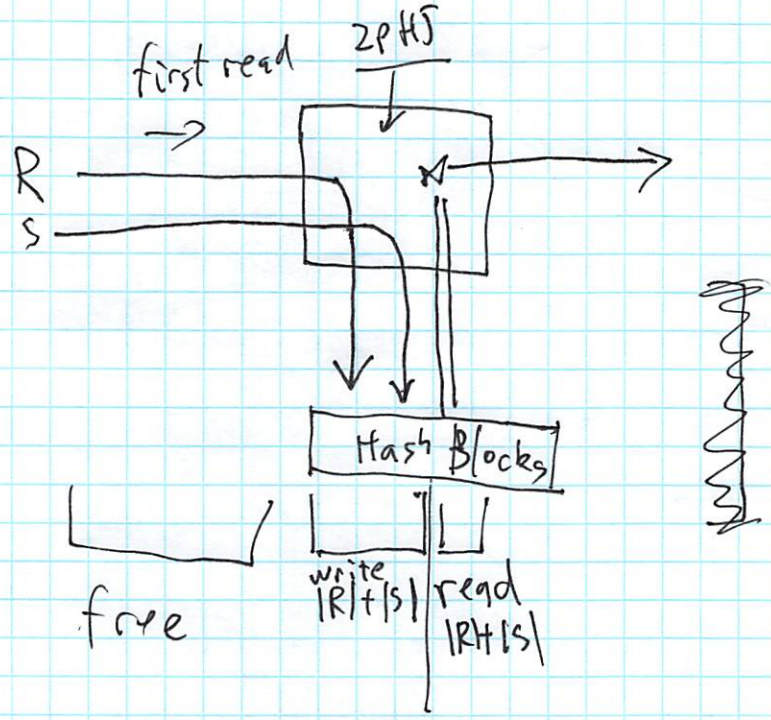
1p TJ



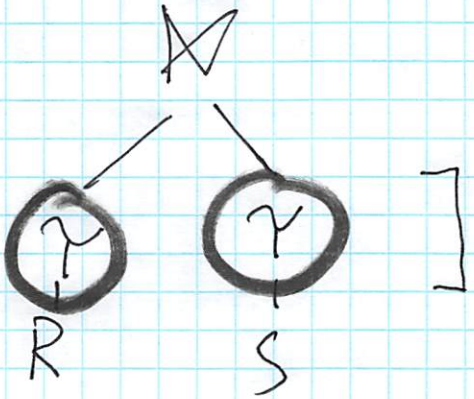
2p HJ

for s in S:
    write s to on-disk bucket hash(s)
for r in R:
    write r to on-disk bucket hash(r)

for b in bucket:
    in mem join on all tuples in b


first read
2PHJ
R
S
Hash Blocks
free
write
|R|+|S|
read
|R+|S|

SMJ



INLJ

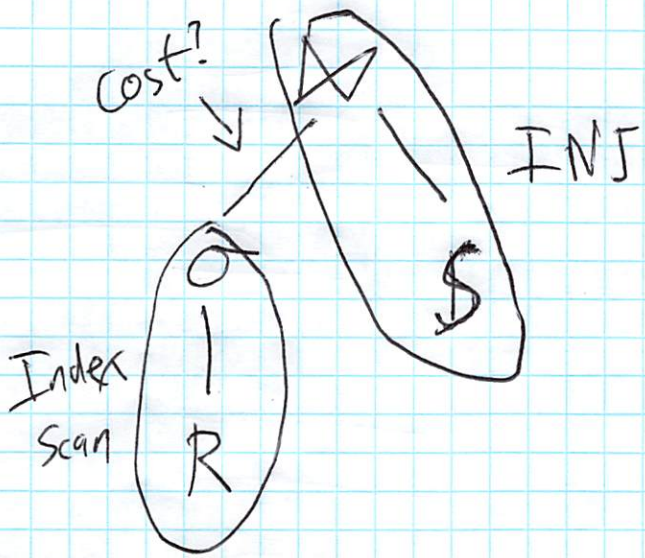
```
for r in R:
  for s in
  S.index lookup(r):
    emit(r,s)
```

loop for # of records in R

cost depends on type of index

Hash: $O(1) \rightarrow |$

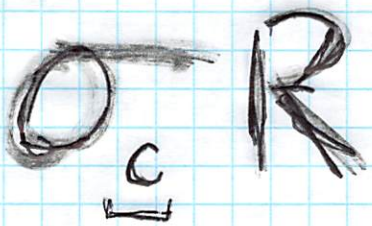
Tree: $\log|S|$



Π, τ : no change

$R \cup S$: sum of $|R|, |S|$

$R \times S$: product of $|R|, |S|$



C $A = const$

$A < const$, $A_{const} < A < const$

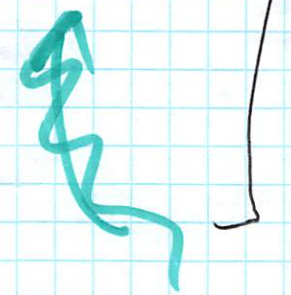
$A \neq const$

$A_i = A_2$

$A = Const$

Idea 1

SELECT A, count(*)
FROM R
GROUP BY A



] statistic

TMI

Idea 2

Store only Max # of ~~rows~~ rows with a given A

↑ upper bound on $|\sigma_{C=A} R|$

Avg : avg of $|\sigma_{C=A} R|$

Min : lower bound

Idea 3

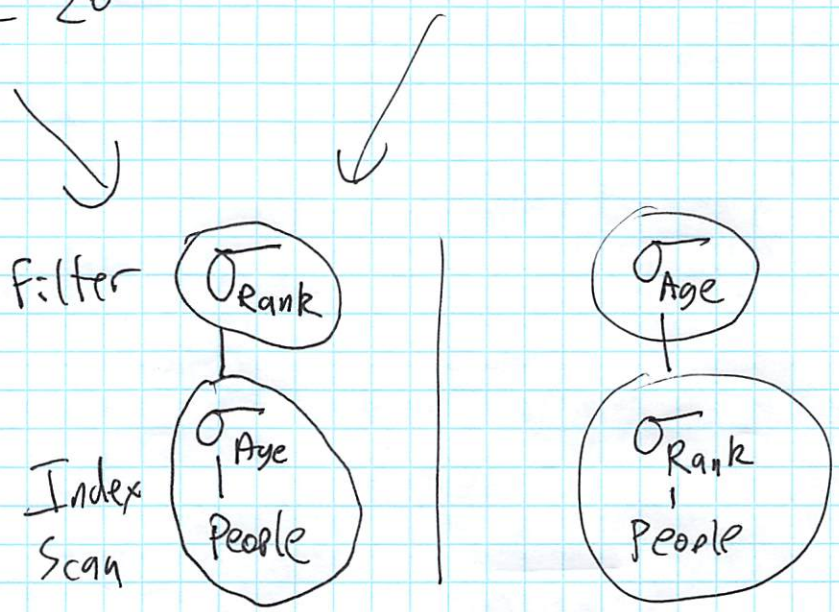
Store min, max of A]

```
SELECT Name
FROM People
WHERE Rank = 3
  AND Age = 20
```

People

Index on Rank

Index on Age



Rank → #	Age → #
1 → 3	19 → 1
2 → 2	20 → 4
3 → 3	21 → 2
	22 → 1

Optimal for (3, 20)

	Rank	age
count	max 3	max 4
	avg 2.6	avg 2
	min 2	min 1
values	min 1	min 19
	max 3	max 22
overall size	8	8

$|\sigma_{rank} R|$ vs $|\sigma_{age} R|$

Using Statistics

Idea 2 → no more than 3 tuples in $\sigma_r R$ ✓
no more than 4 tuples in $\sigma_a R$

Idea 2.1 → avg of $2.\overline{6}$ tuples in σ_r
2 tuples in σ_a ✓

Idea 3 → avg of $\frac{8}{3}$ tuples in σ_r
avg of $\frac{8}{4}$ tuples in σ_a ✓