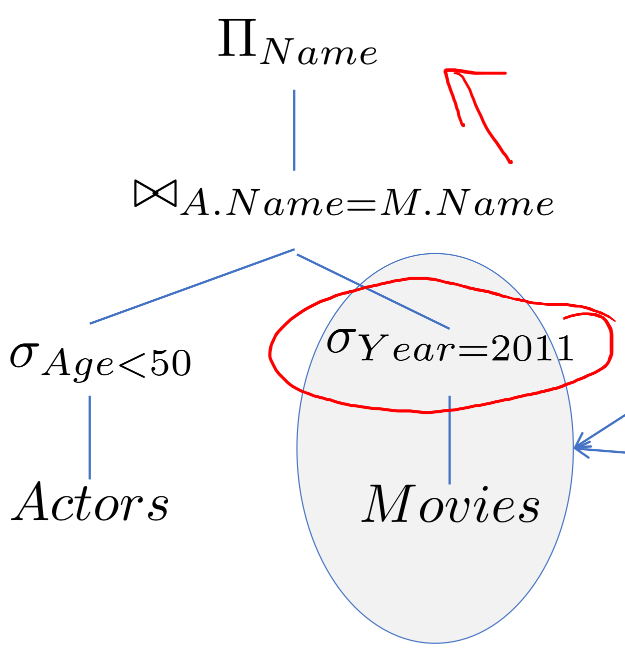
COL 362 & COL 632

Processing SQL – Physical Plans

14 Feb 2023

cardinality estimation

Physical operators – Selection



- SCAN the tuples in Movies one by one
- Retain tuples which satisfy the condition

- If an index is present on Year, then perform an INDEX-SCAN
- Fetch tuples which satisfy the condition

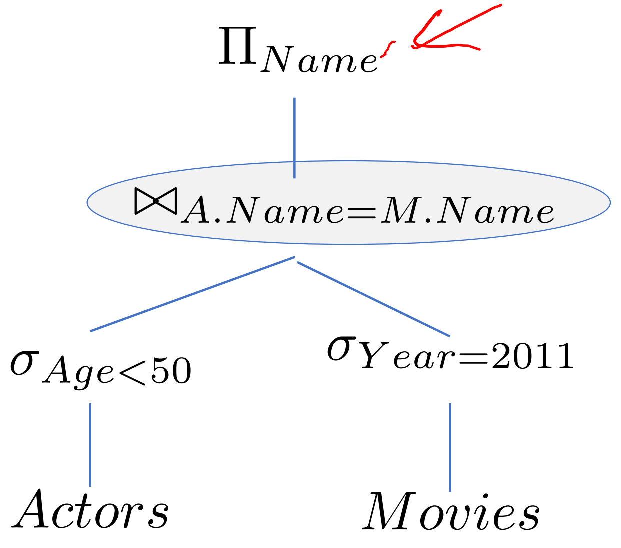
The output of the operator: disk or memory?

$[r_i, s_i]$

Physical Operators – Joins (1/2)

$\bowtie_{A.age}$ $\boxed{\geq}$ now

M.release



Nested-loop Join (R,S)

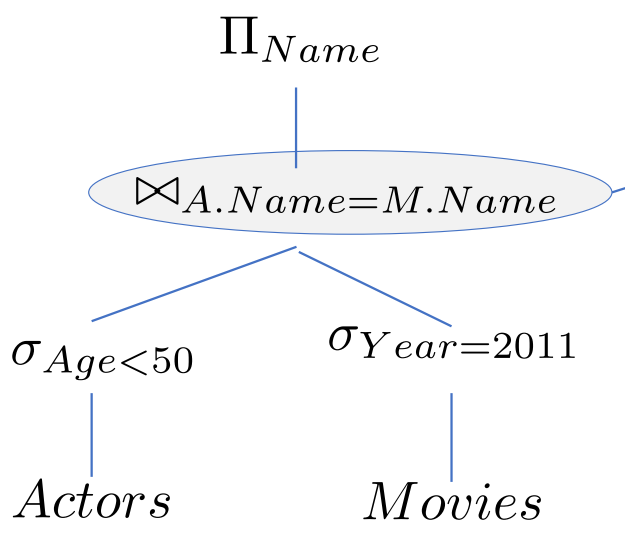
```
iterate over $r_i$ in R
  iterate over $s_i$ in S
     if (join-condition satisfied)
        output  join($r_i$,$s_i$)
  end iteration
end iteration
```

When won’t this work?

Hash Join (R,S)

```
iterate over $r_i$ in R
  hash $r_i$
iterate over $s_i$ in S
  hash $s_i$
join tuples in same bucket ✓
```

Physical Operators – Joins (2/2)

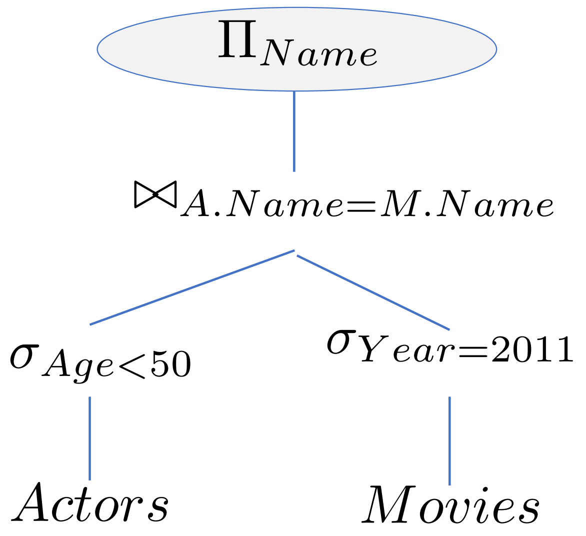


Sort-Merge Join (R,S)
Sort R
Sort S
Merge based on join condition

Index-based Join (R,S)
Indexes provide sorted access

What kind of indexes are ideal?

Physical Operators – Projection



- SCAN the tuples one by one
- Retain required attributes

Grouping and Aggregation

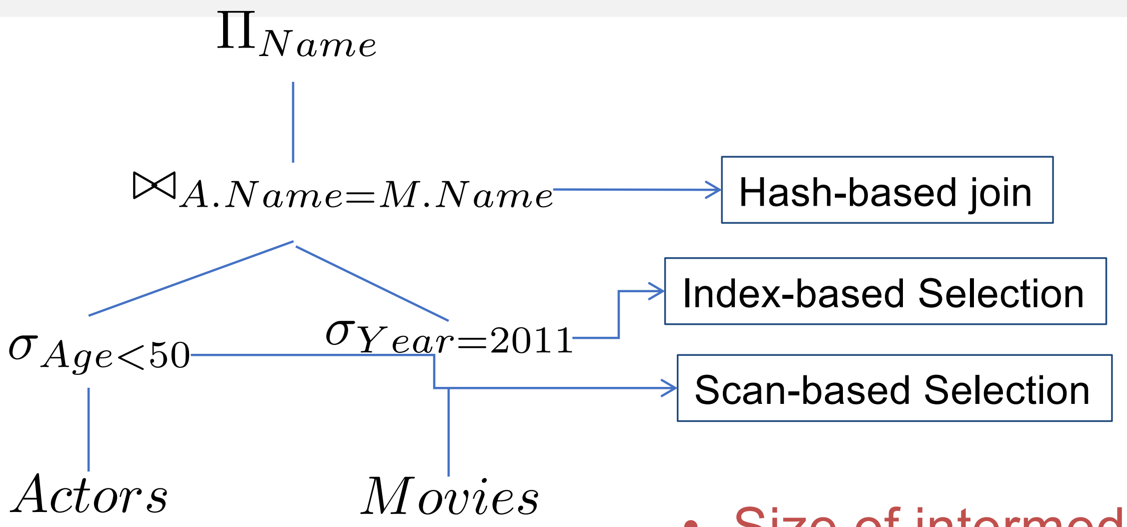
- Hash-based grouping
- Scan-based aggregation

$$\gamma_{\{G,A\}}(S)$$

For each group, compute aggregates A

Hash each tuple on the grouping attributes G

Cost of a plan



- Size of intermediate results
  - Order of operations
  - Choice of algorithms
- Various system parameters
  - Available memory ✓
  - Disk contention ✓

Cost of a Query

① Number of disk blocks read/written

② Number of "random" I/O.

- Depends on where you store the data ✓
  - Memory, Disk, SSD, Tapes, Cloud, …
- How you access the data ✓
  - Sequential vs. Random
  - Blocked transfer vs. "just what you need"
- If one has access path (or index) and its cost ✓

- CPU cost per tuple ✓
- CPU cost for each index entry (+I/O) ✓
- CPU cost per operator ✓

Analyzing Query Plans

- EXPLAIN statement

HW: Understand the workings of the EXPLAIN statement in PostgreSQL

```
# explain select * from people, managers where people.playerid=managers.playerid;
                                       QUERY PLAN
---------------------------------------------------------------------------------------
 Merge Join  (cost=265.09..1341.11 rows=3370 width=185)
   Merge Cond: ((people.playerid)::text = (managers.playerid)::text)
   ->  Index Scan using people_pkey on people  (cost=0.00..982.40 rows=18599 width=144)
   ->  Sort  (cost=263.16..271.58 rows=3370 width=41)
         Sort Key: managers.playerid
         ->  Seq Scan on managers  (cost=0.00..65.70 rows=3370 width=41)
(6 rows)
```