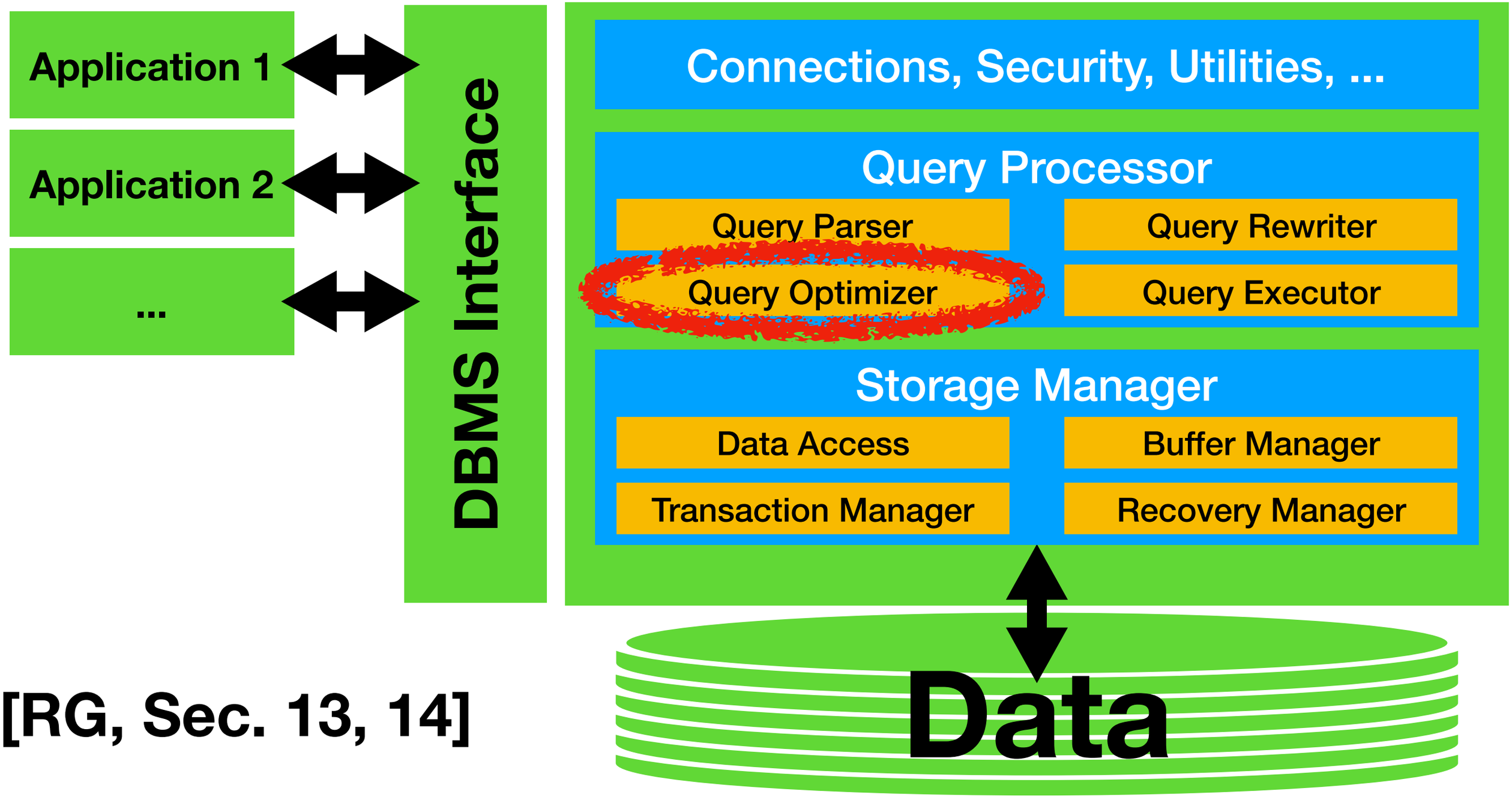
Query Optimization

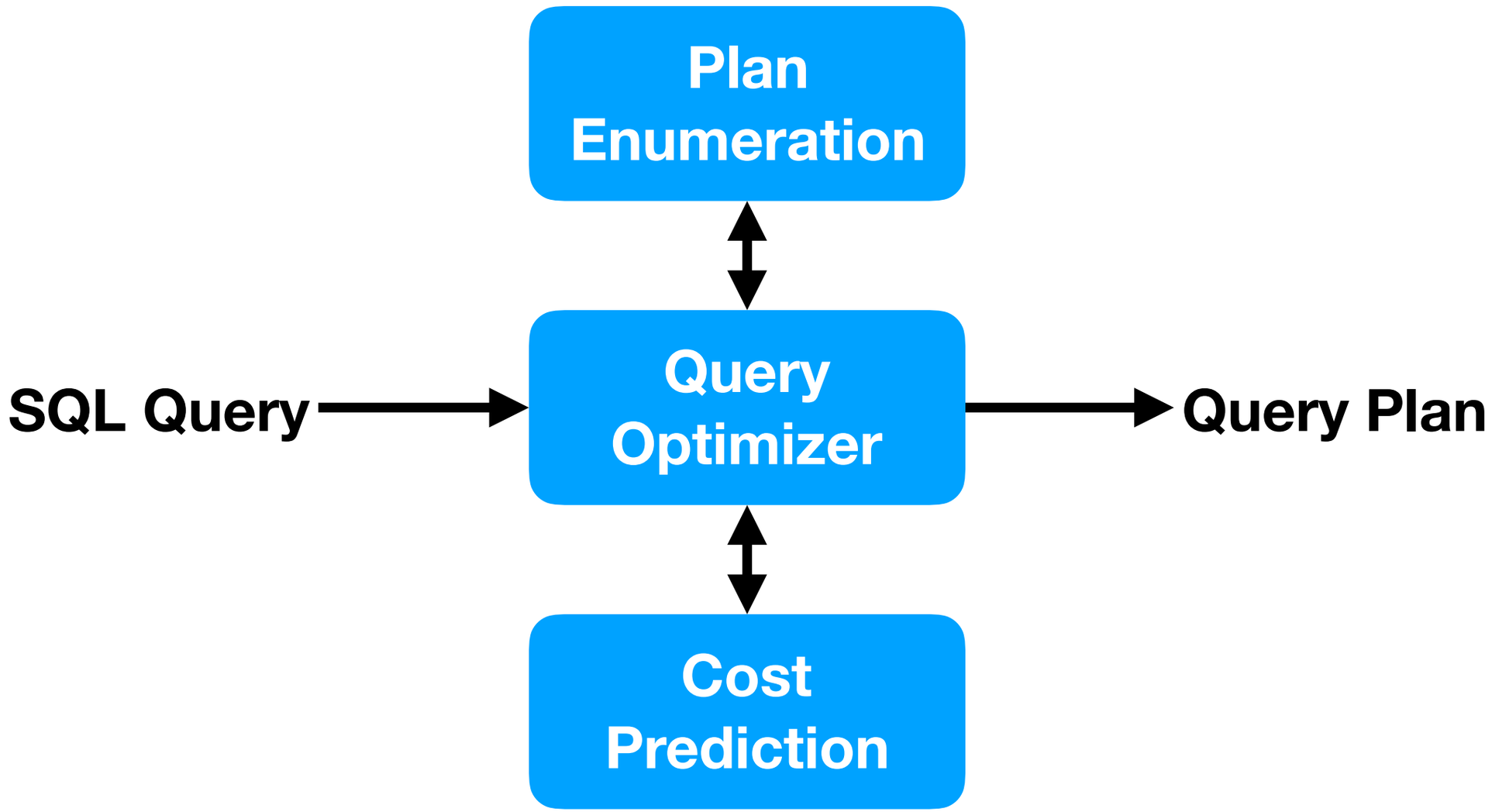
Immanuel Trummer
itrummer@cornell.edu
www.itrummer.org

Database Management Systems (DBMS)

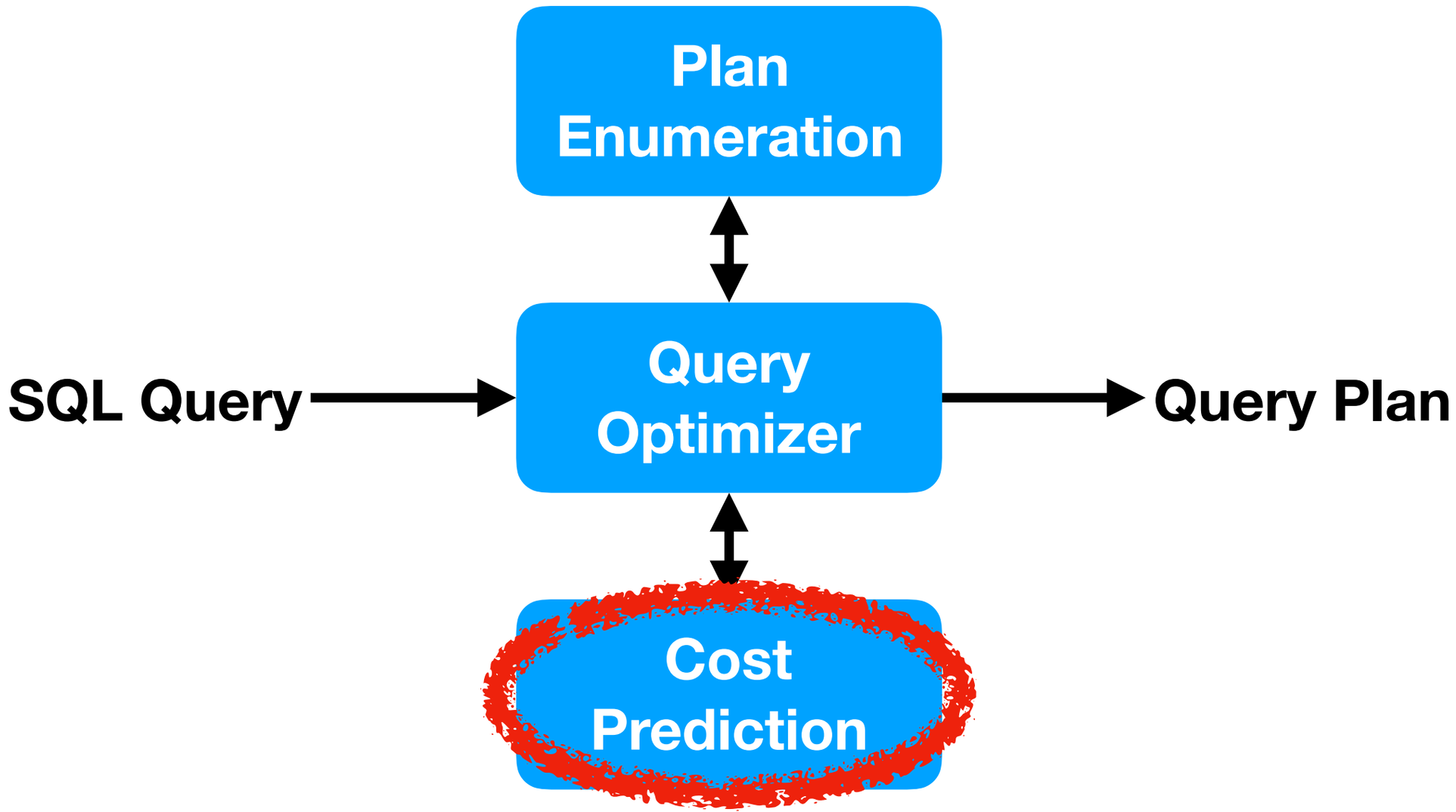


[RG, Sec. 13, 14]

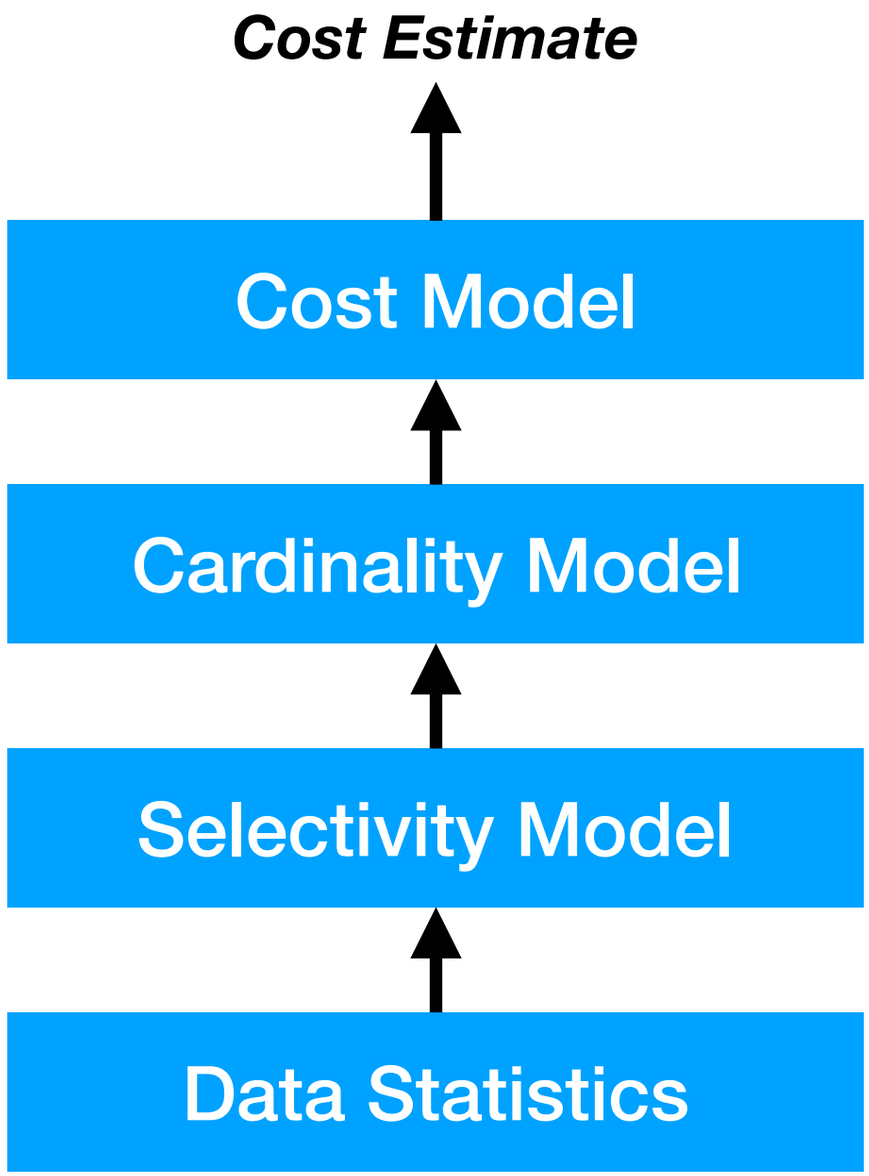
Optimizer Overview



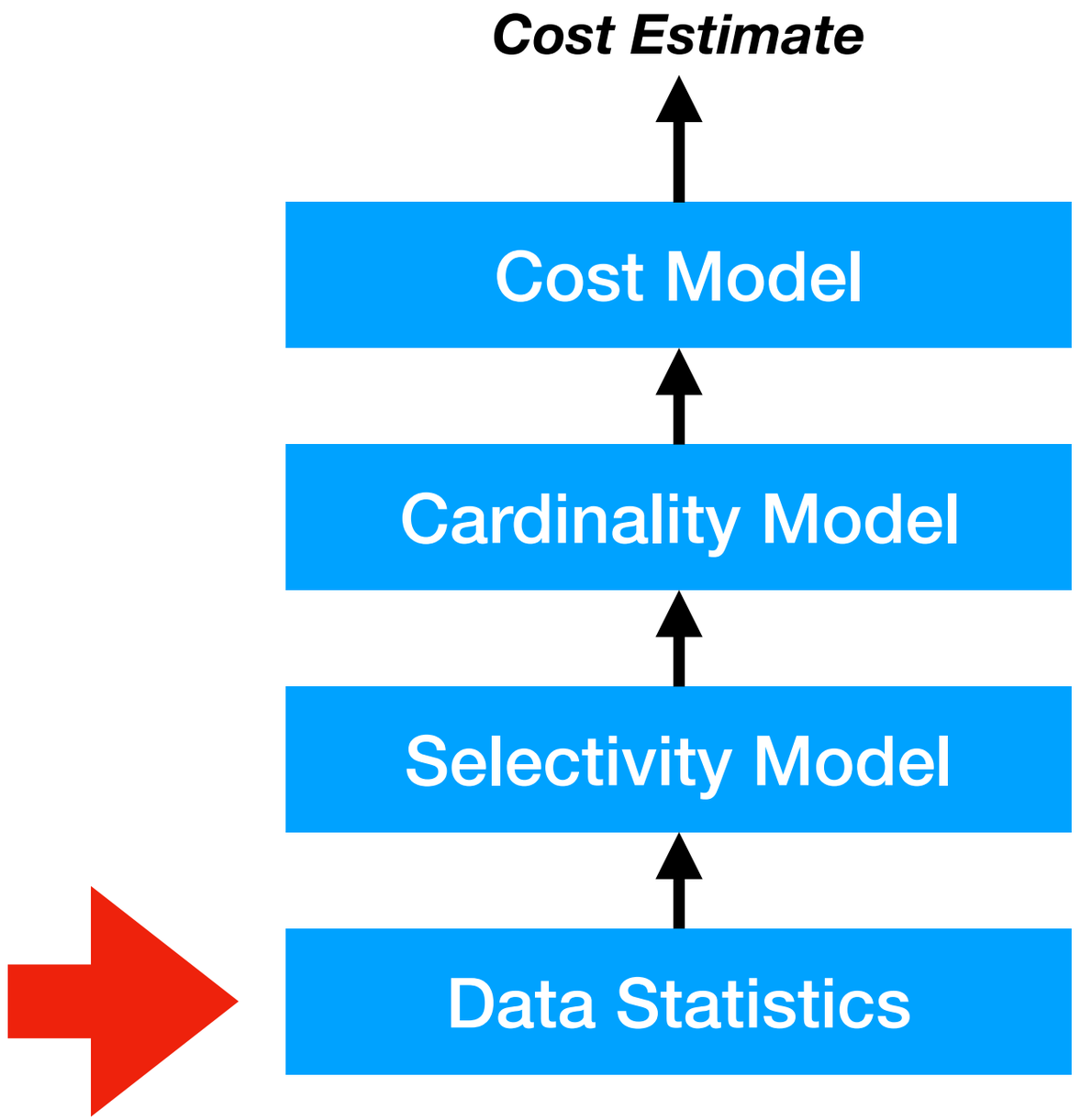
Optimizer Overview



Cost Model Structure



Cost Model Structure



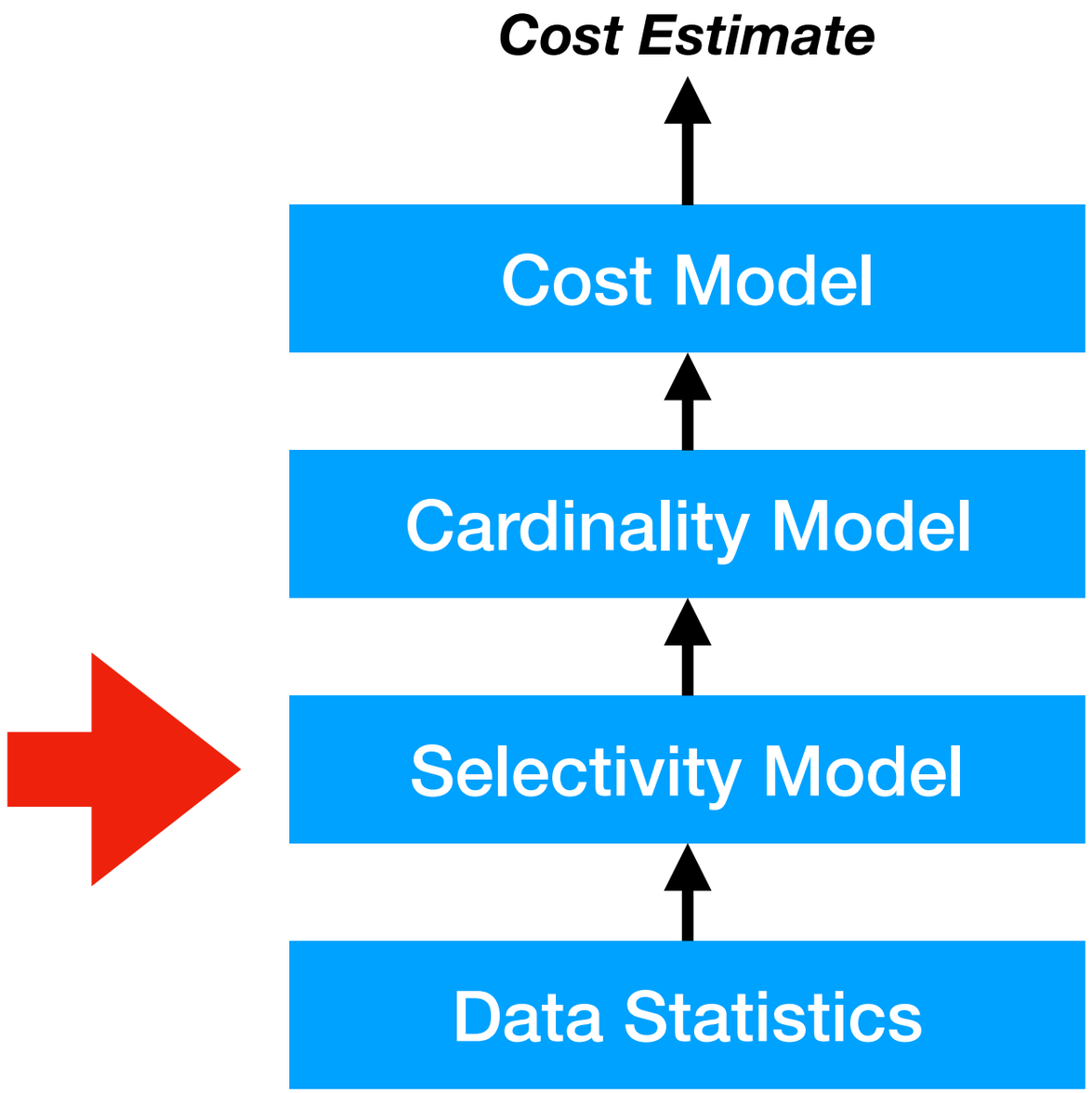
Data Statistics

- Different DBMS collect **different** statistics
- **Example statistics** (collected in Postgres):
  - Number of **distinct values** in column
  - Ratio of SQL **null values** in column
  - Most **frequent values** with associated frequency
  - **Histograms** approximating column data distribution
  - ...

Data Statistics in PG

- Force DBMS to create statistics via **VACUUM ANALYZE**
- Generated **statistics** are exploited by **query optimizer**
  - Inexplicably bad performance? Might **miss statistics**
- Can access column statistics via **pg_stats** view
  - E.g., **tablename**, **attname**, **n_distinct**, **null_frac**, ...

Cost Model Structure



Estimating Selectivity

- **Selectivity**: probability that row satisfies predicate

- Estimate via **constraints** and data **statistics**

- E.g., **Selectivity(Column=Value) = 1/NrValues**

- E.g., **NrRows(Column=Value) ≤ 1 if Key Column**

- **Selectivity(A∧B) = Selectivity(A) * Selectivity(B)**

Which Simplifying Assumptions Do We Make?

Estimating Selectivity

- **Selectivity**: probability that row satisfies predicate

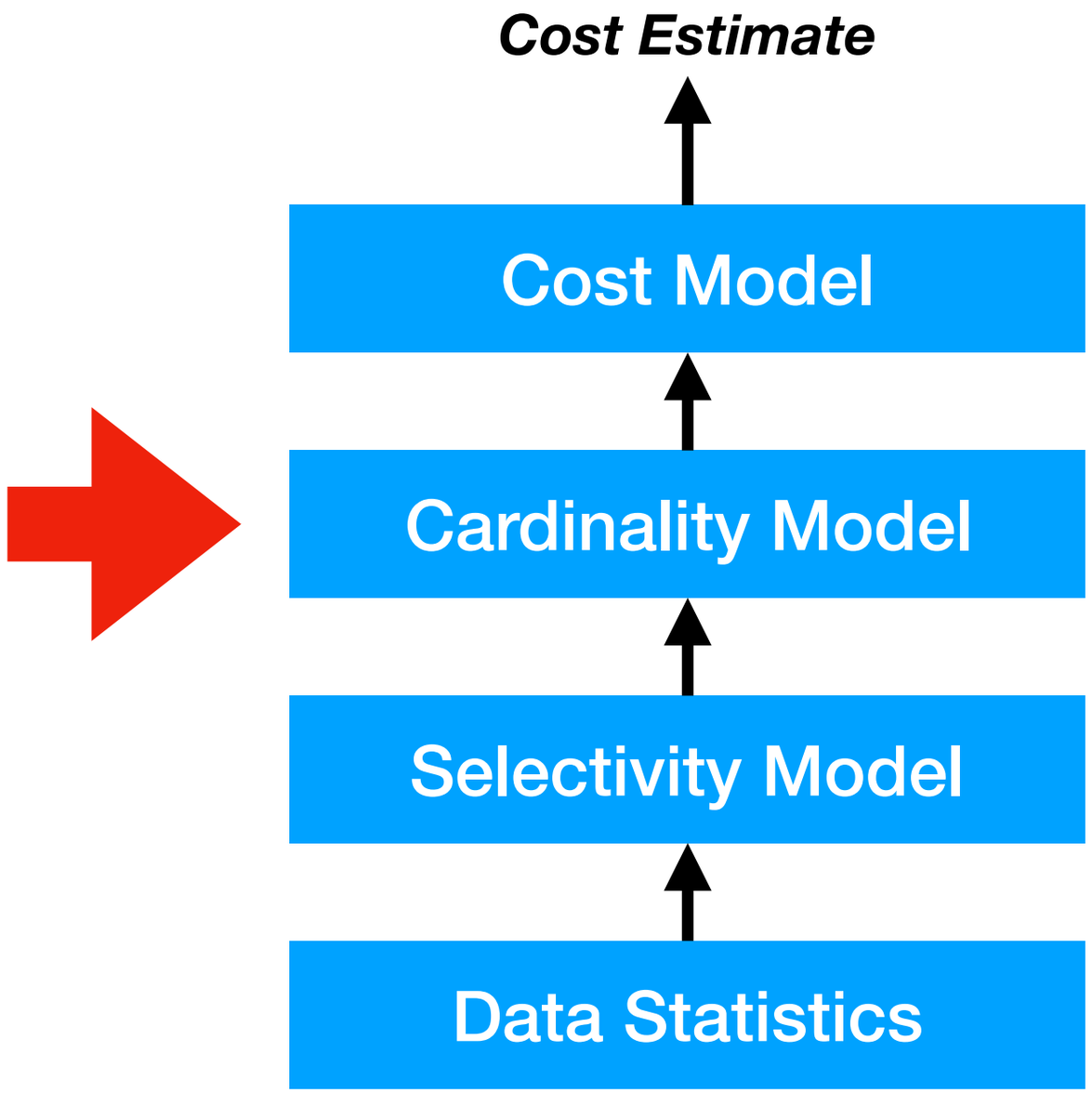
- Estimate via **constraints** and data **statistics**

- E.g., **Selectivity(Column=Value) = 1/NrValues**
  Assumes Uniform Data

- E.g., **NrRows(Column=Value) ≤ 1 if Key Column**

- **Selectivity(A∧B) = Selectivity(A) * Selectivity(B)**
  Assumes Independent Predicates

Cost Model Structure



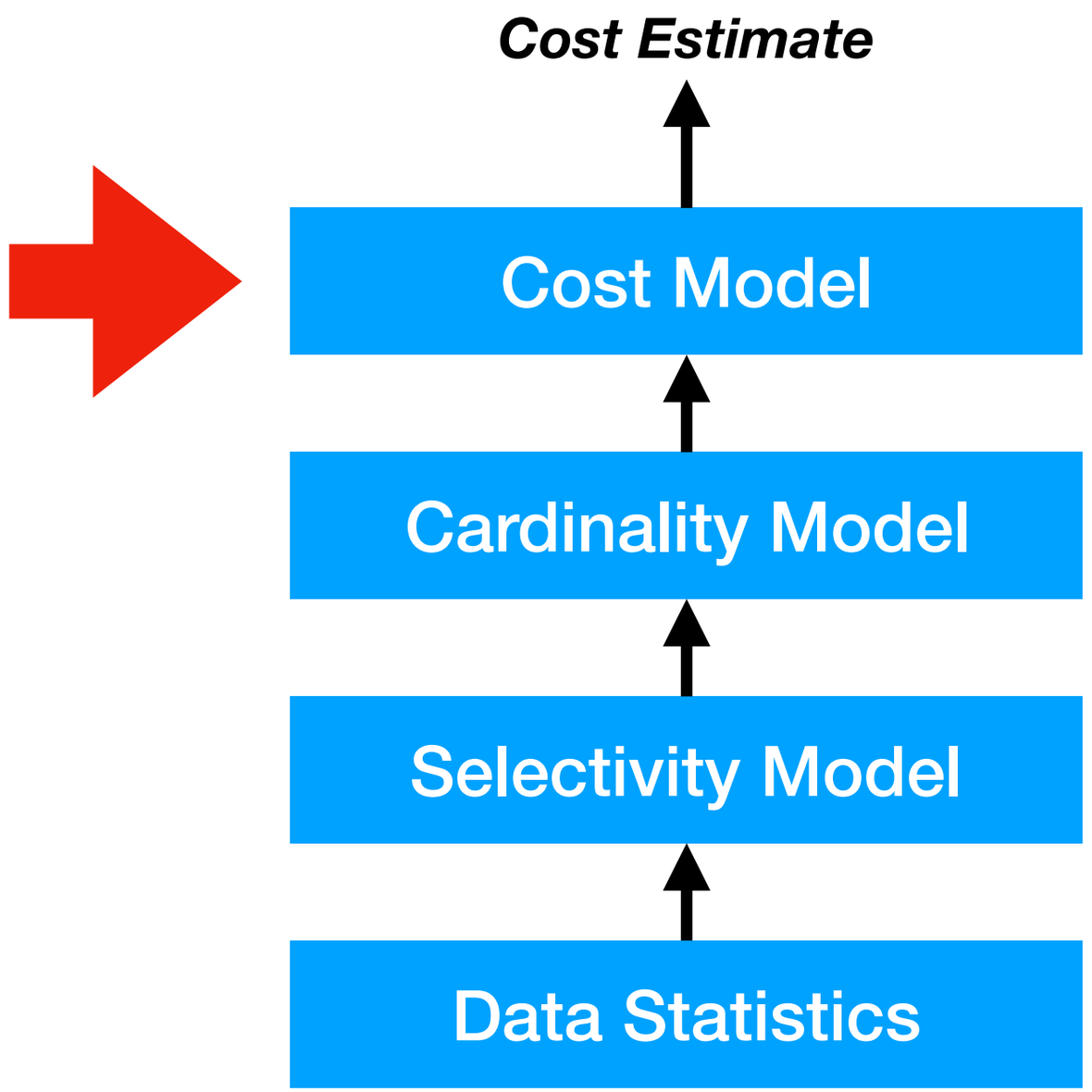
Estimating Cardinality

- Assume that **cardinality of single tables** is given
  - DBMS **store that information** for each base table
- Calculate **cardinality product** for tables in from clause
  - = Number of **result rows** if predicates are always true
- Now **multiply with selectivity** estimates for all predicates
  - Estimate how many rows **pass** predicate filter

Estimating Page Size

- Cost functions are based on number of data **pages**
- Calculate average byte **size per record** for each result
- Calculate how many records **fit** on one data page
  - Pages cannot store fractional records → **round down**
- **Divide** number of rows by number of records per page
  - Result is size, measured in **pages**

Cost Model Structure



Estimating Cost

- Generally only count cost of **page reads and writes**
- **Sum up cost** over all operators in the plan
- For each operator consider the following:
  - For **each input**, how often is it read from disk?
  - For **each output**, is it written to disk? Is it final result?
  - Does the operator read/write **intermediate results**?

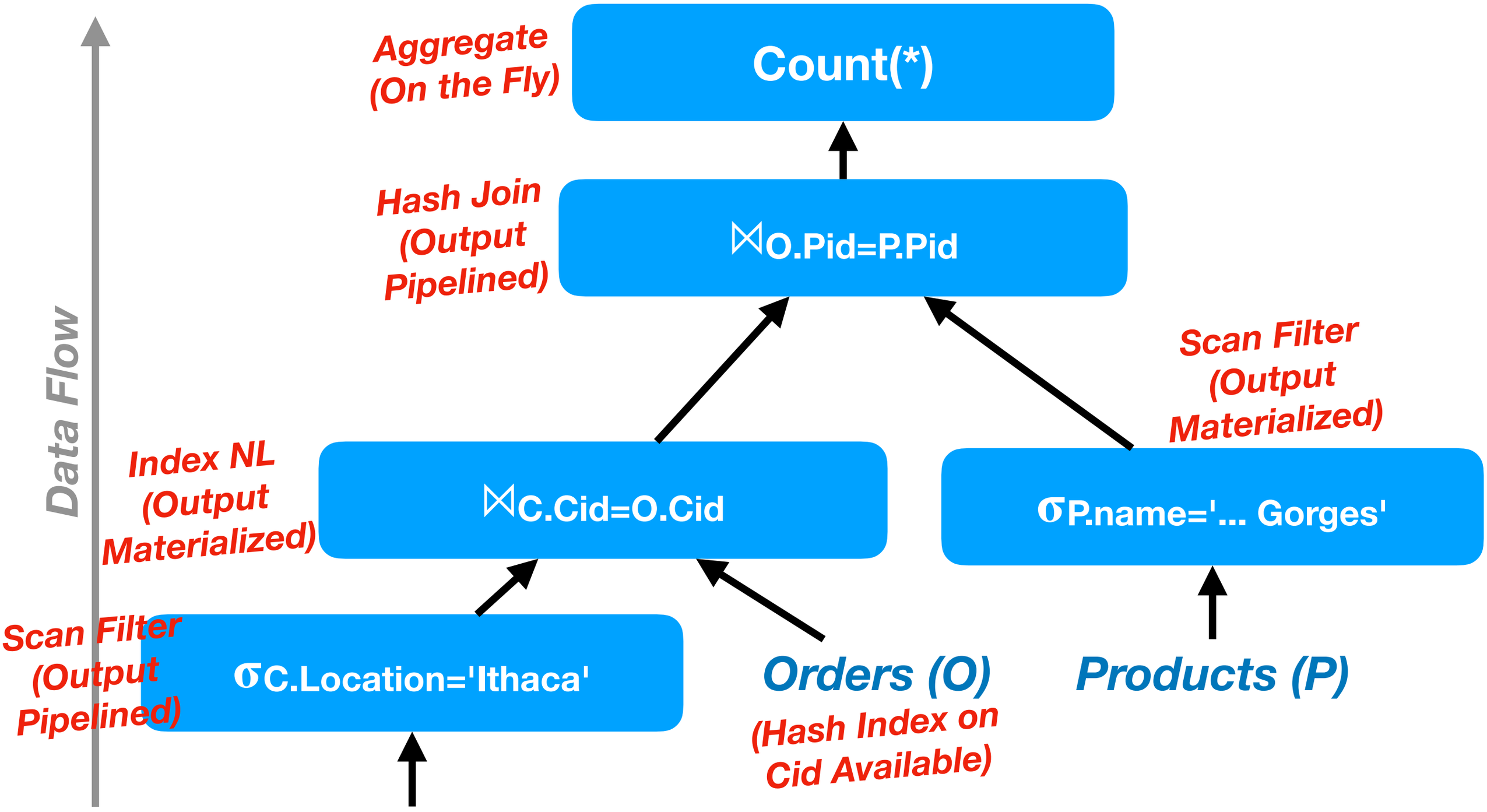
Cost Estimation Example

Input Query and Database

```
SELECT Count(*)
FROM Customers C JOIN Orders O USING (cid)
  JOIN Products P USING (pid)
WHERE C.location = 'Ithaca' AND
  P.name = 'Book Ithaca is Gorges'
```

Customers(Cid, name, location)
Orders(Cid, Pid, date)
Products(Pid, name, price)

Query Plan Candidate



Selectivity Estimation

Element	Property	Value
Customers	Cardinality	10,000
Orders	Cardinality	100,000
Products	Cardinality	5,000
Customers.Location	# Values	100
Products.Name	# Values	2,500

Foreign Key: Orders.Cid to Customers.Cid
Foreign Key: Orders.Pid to Products.Pid

Predicate	Selectivity
σC.Location='Ithaca'	?
C.Cid=O.Cid	?
σP.name='... Gorges'	?
O.Pid=P.Pid	?

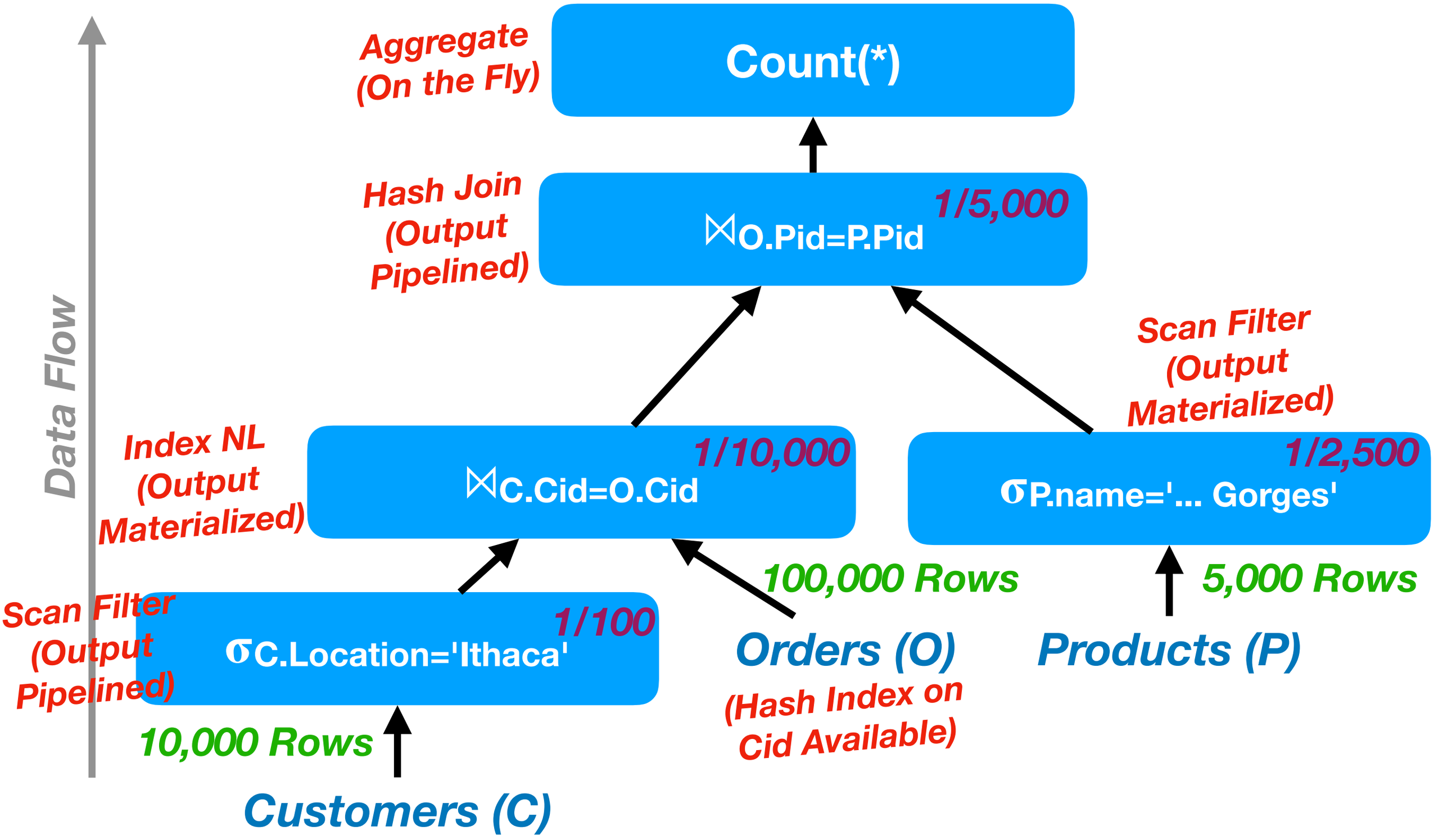
Selectivity Estimation

Element	Property	Value
Customers	Cardinality	10,000
Orders	Cardinality	100,000
Products	Cardinality	5,000
Customers.Location	# Values	100
Products.Name	# Values	2,500

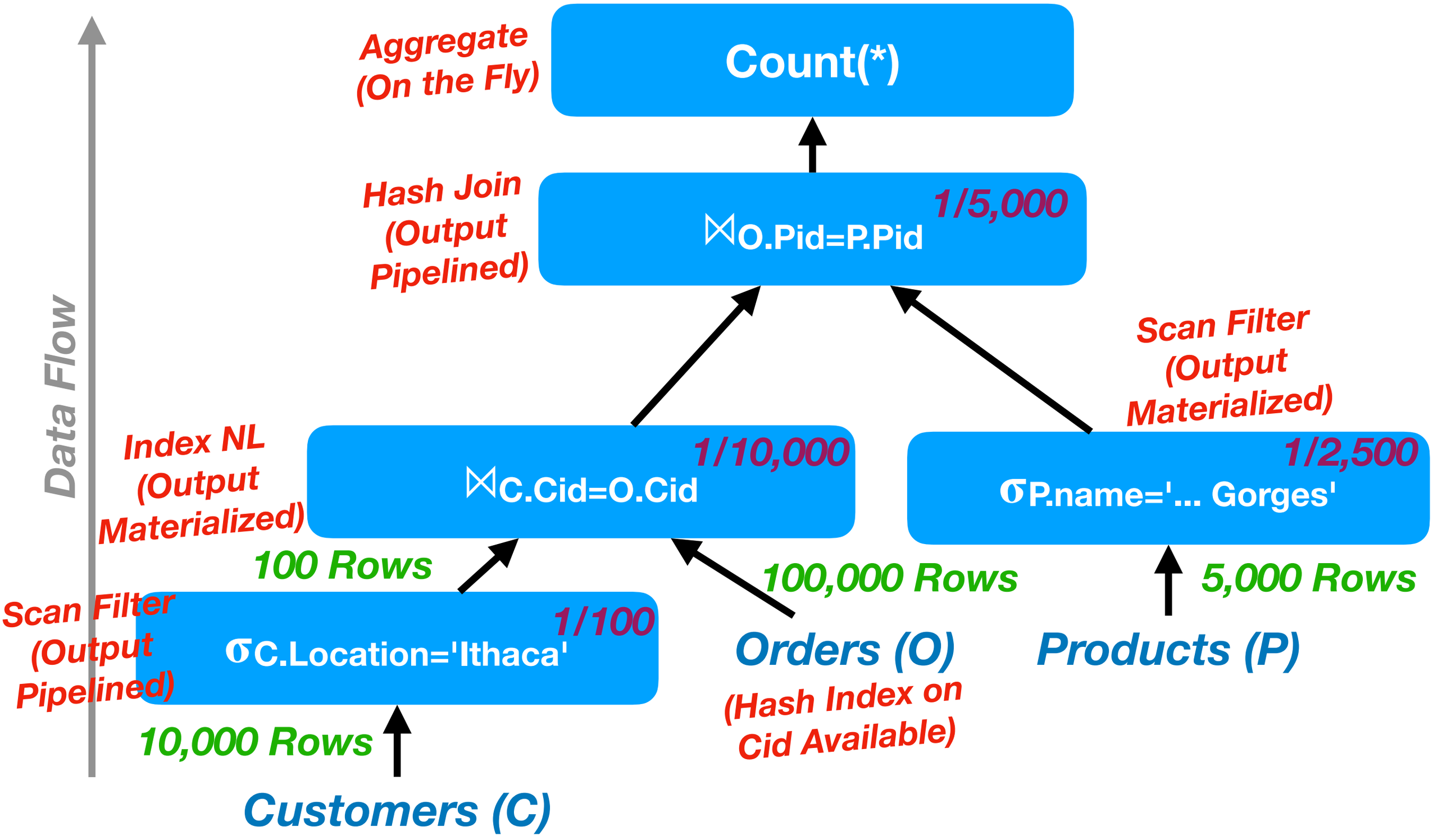
Foreign Key: Orders.Cid to Customers.Cid
Foreign Key: Orders.Pid to Products.Pid

Predicate	Selectivity
σC.Location='Ithaca'	1/100
C.Cid=O.Cid	1/10,000
σP.name='... Gorges'	1/2,500
O.Pid=P.Pid	1/5,000

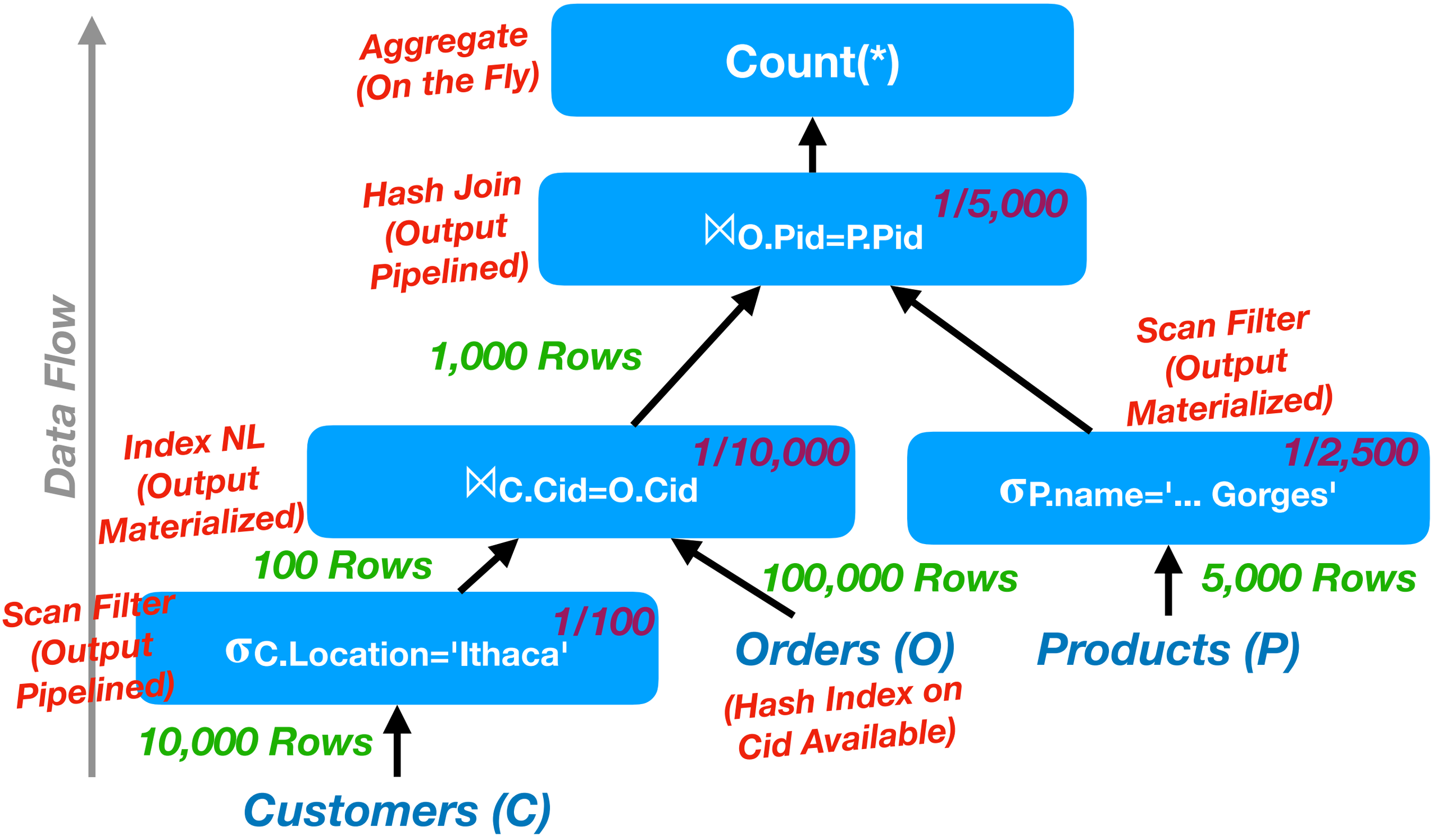
Cardinality Estimation



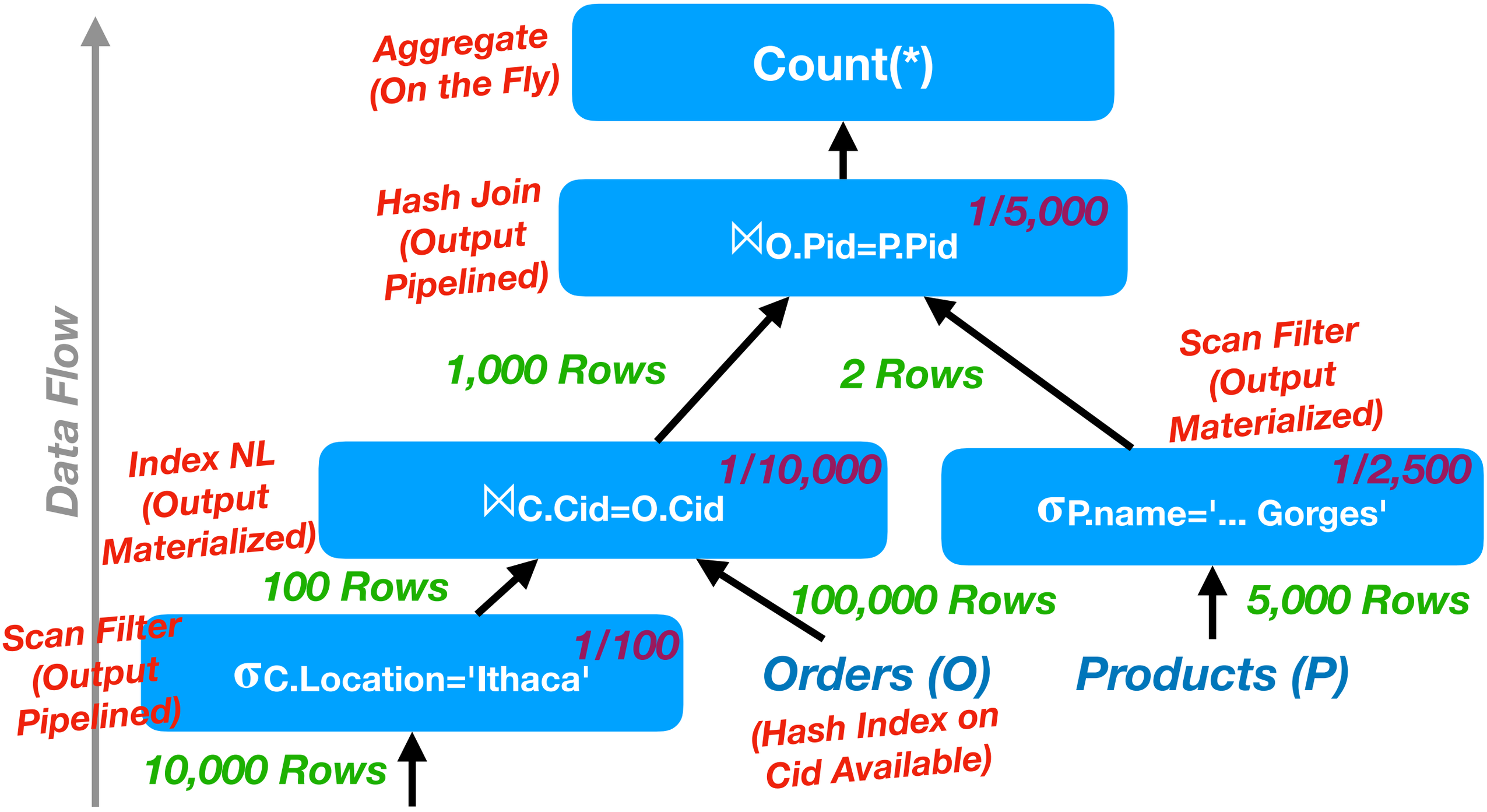
Cardinality Estimation



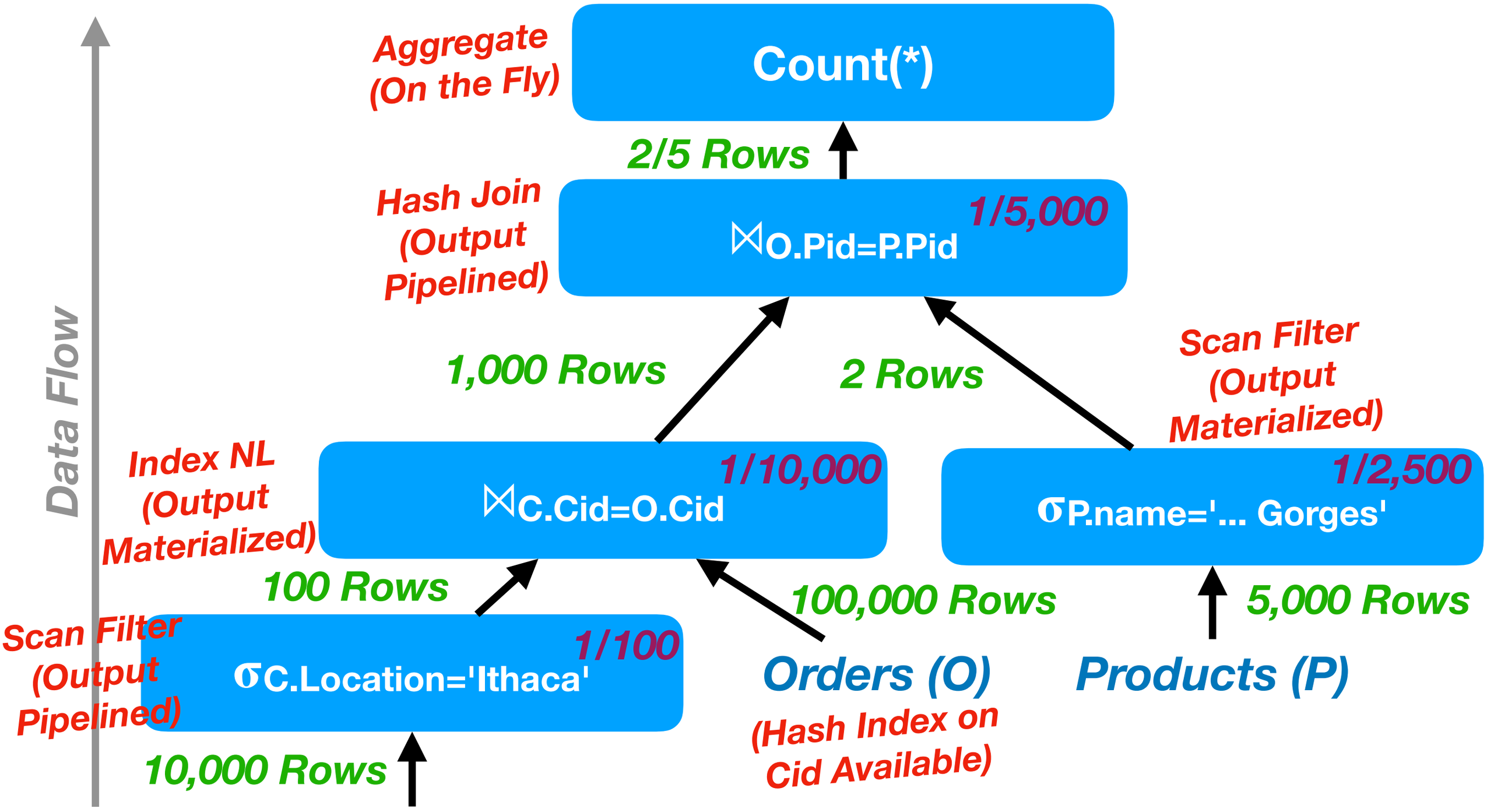
Cardinality Estimation



Cardinality Estimation



Cardinality Estimation



Is That Realistic ... ?

Size Estimation (Preparation)

Table	Columns	Size/Entry	Rows/Page
Customers	Cid, name, location	?	?
Orders	Cid, Pid, date	?	?
Products	Pid, name, price	?	?
C⋈O	Cid, name, location, pid, date	?	?
C⋈O⋈P	Cid, C.name, location, pid, date, P.name, price	?	?

Assume page size of 8,000 bytes,
4 bytes for ints and dates, 10 bytes for strings

Size Estimation (Preparation)

Table	Columns	Size/Entry	Rows/Page
Customers	Cid, name, location	24	?
Orders	Cid, Pid, date	?	?
Products	Pid, name, price	?	?
C⋈O	Cid, name, location, pid, date	?	?
C⋈O⋈P	Cid, C.name, location, pid, date, P.name, price	?	?

Assume page size of 8,000 bytes,
4 bytes for ints and dates, 10 bytes for strings

Size Estimation (Preparation)

Table	Columns	Size/Entry	Rows/Page
Customers	**Cid, name, location**	**24**	**333**
Orders	**Cid, Pid, date**	**?**	**?**
Products	**Pid, name, price**	**?**	**?**
C⋈O	**Cid, name, location, pid, date**	**?**	**?**
C⋈O⋈P	**Cid, C.name, location, pid, date, P.name, price**	**?**	**?**

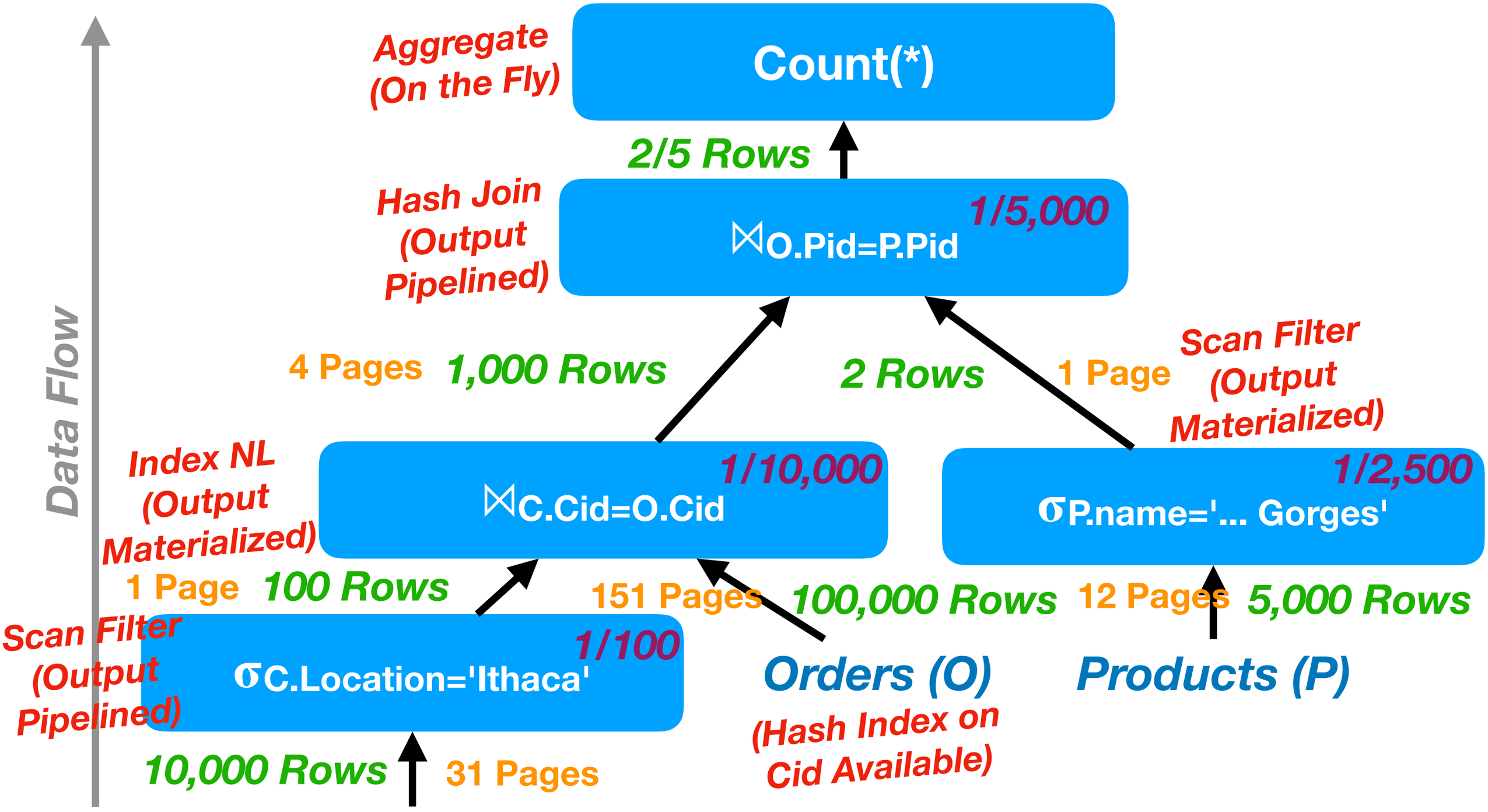
Assume page size of 8,000 bytes,
4 bytes for ints and dates, 10 bytes for strings

Size Estimation (Preparation)

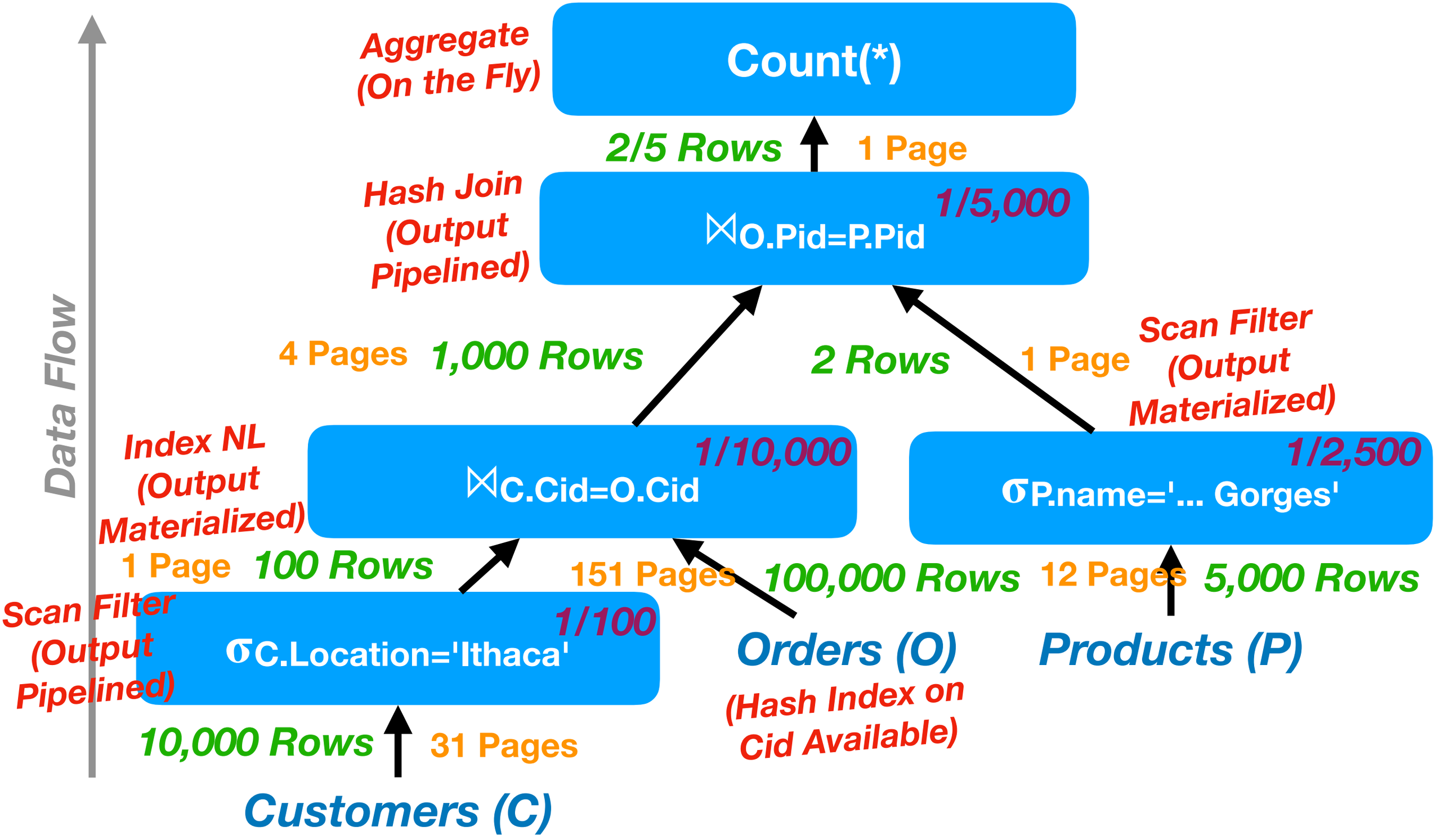
Table	Columns	Size/Entry	Rows/Page
Customers	Cid, name, location	24	333
Orders	Cid, Pid, date	12	666
Products	Pid, name, price	18	444
C⋈O	Cid, name, location, pid, date	32	250
C⋈O⋈P	Cid, C.name, location, pid, date, P.name, price	46	173

Assume page size of 8,000 bytes,
4 bytes for ints and dates, 10 bytes for strings

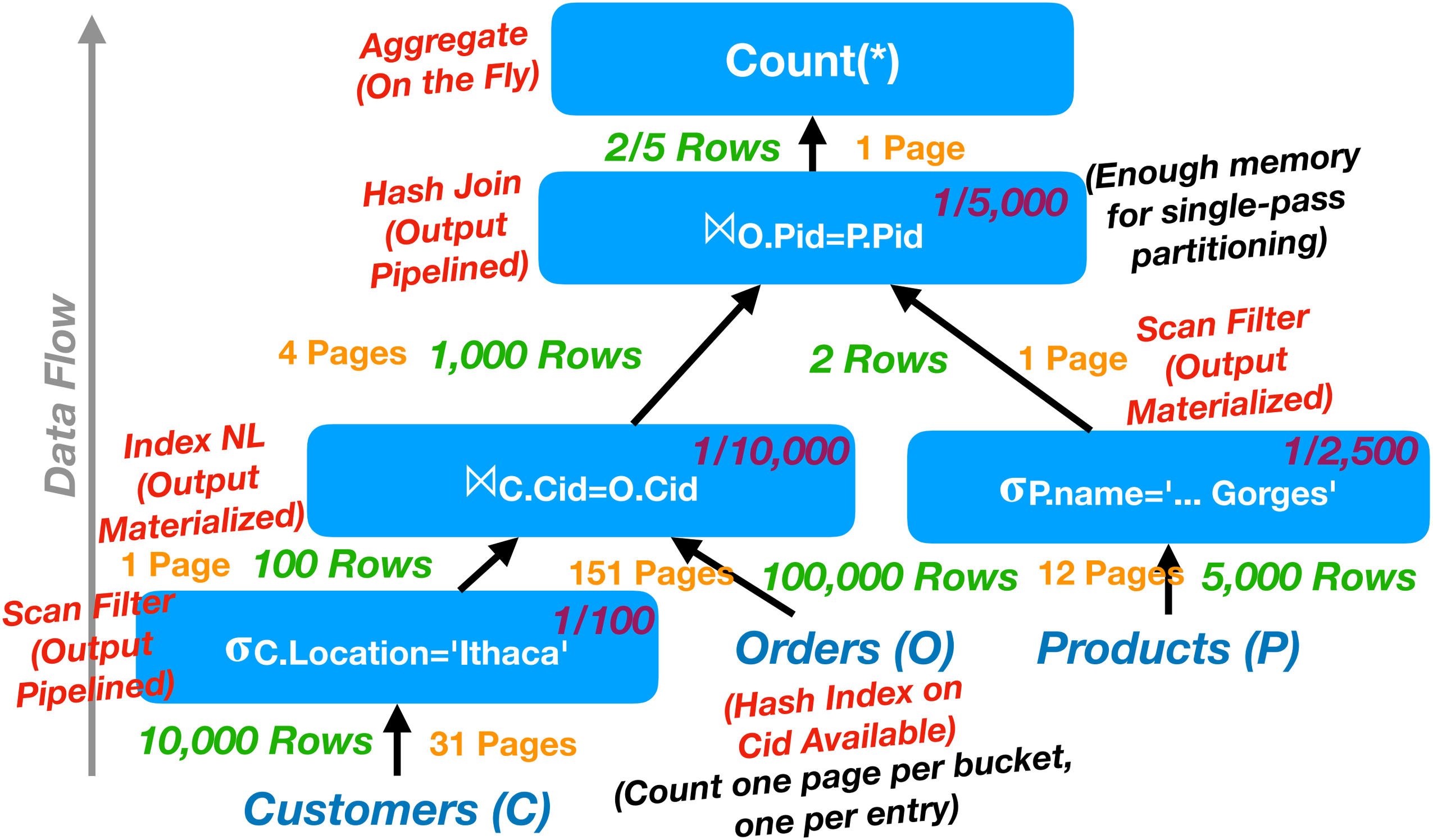
Size Estimation



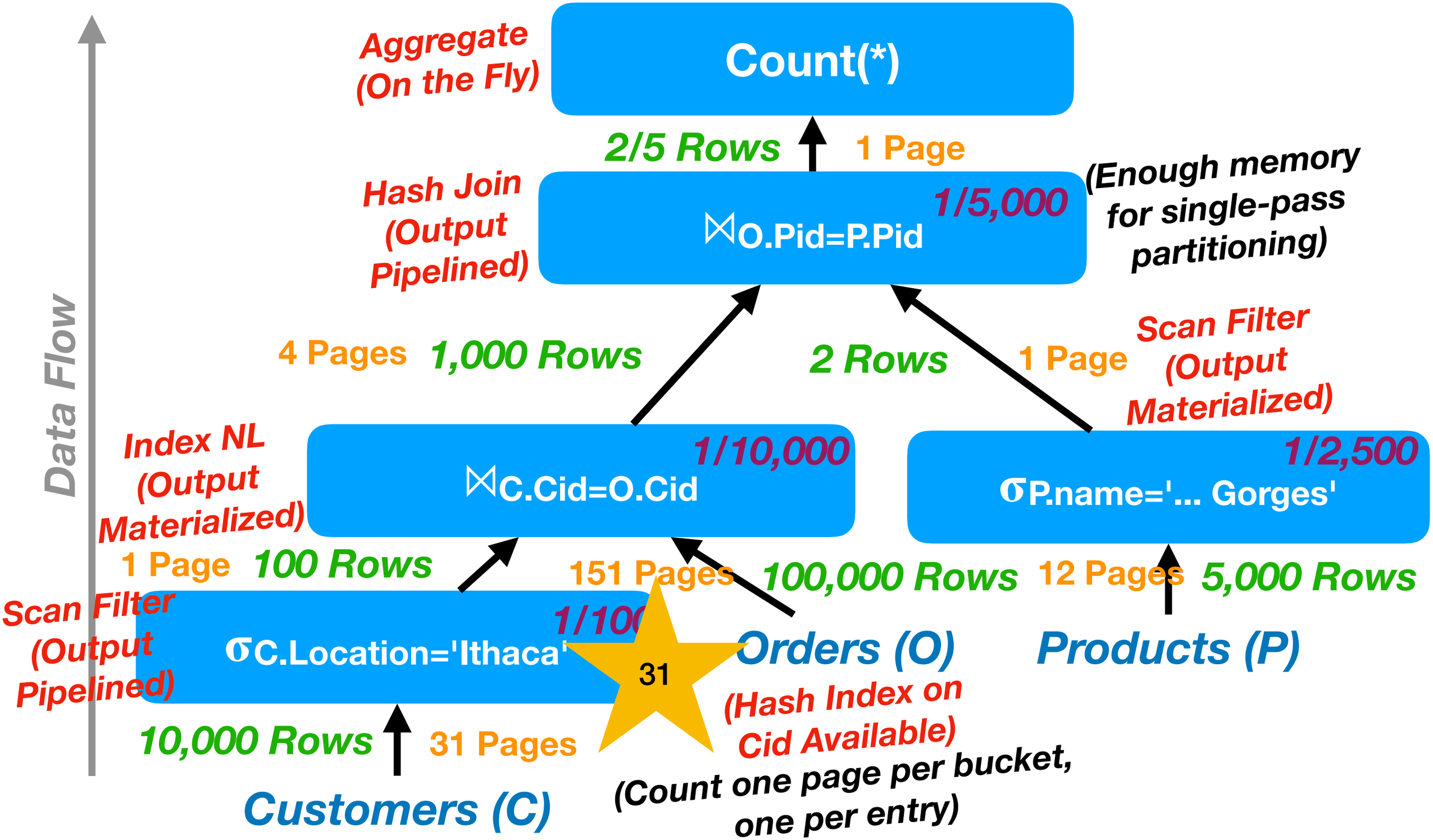
Size Estimation



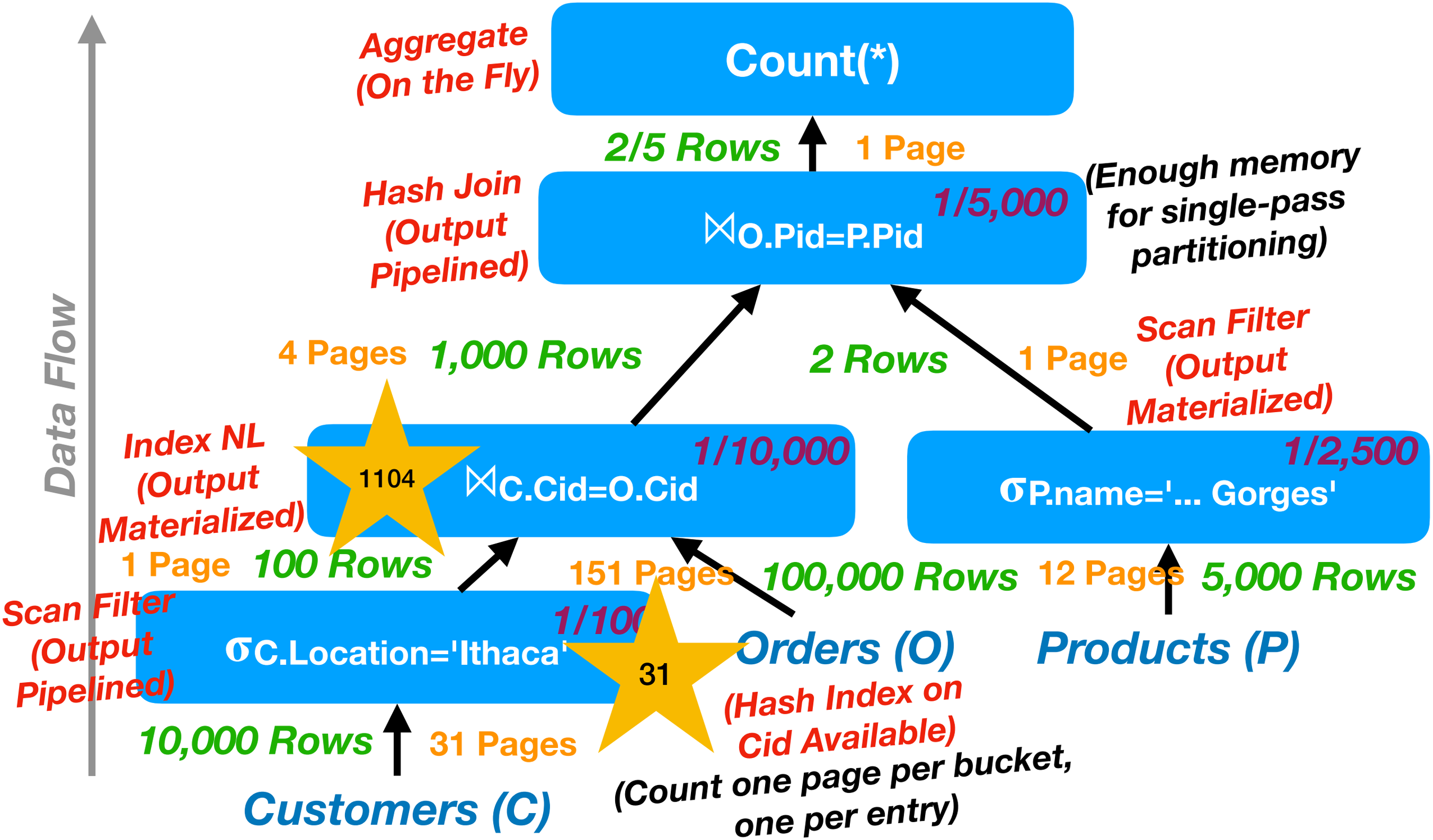
Cost Estimation



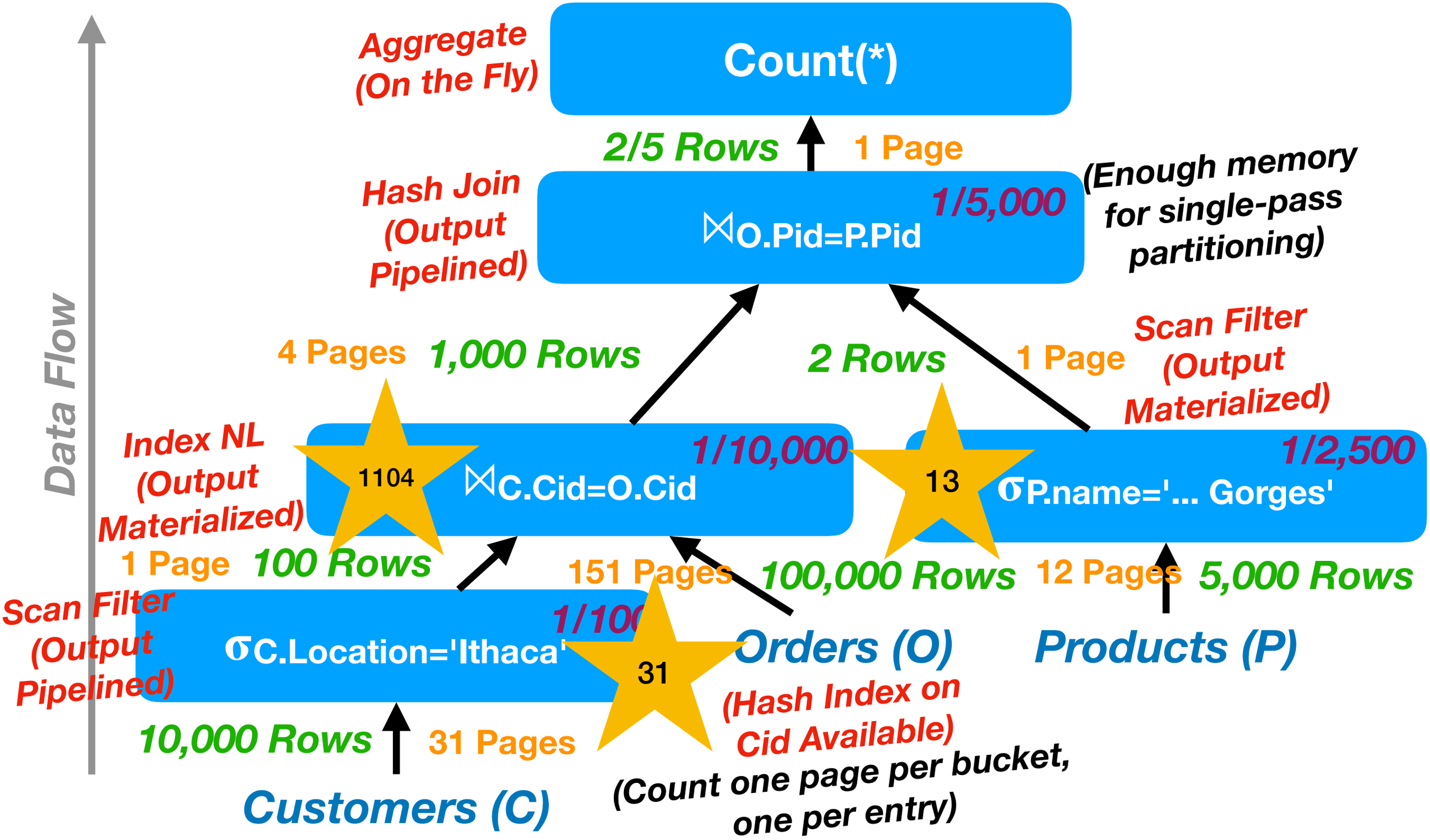
Cost Estimation



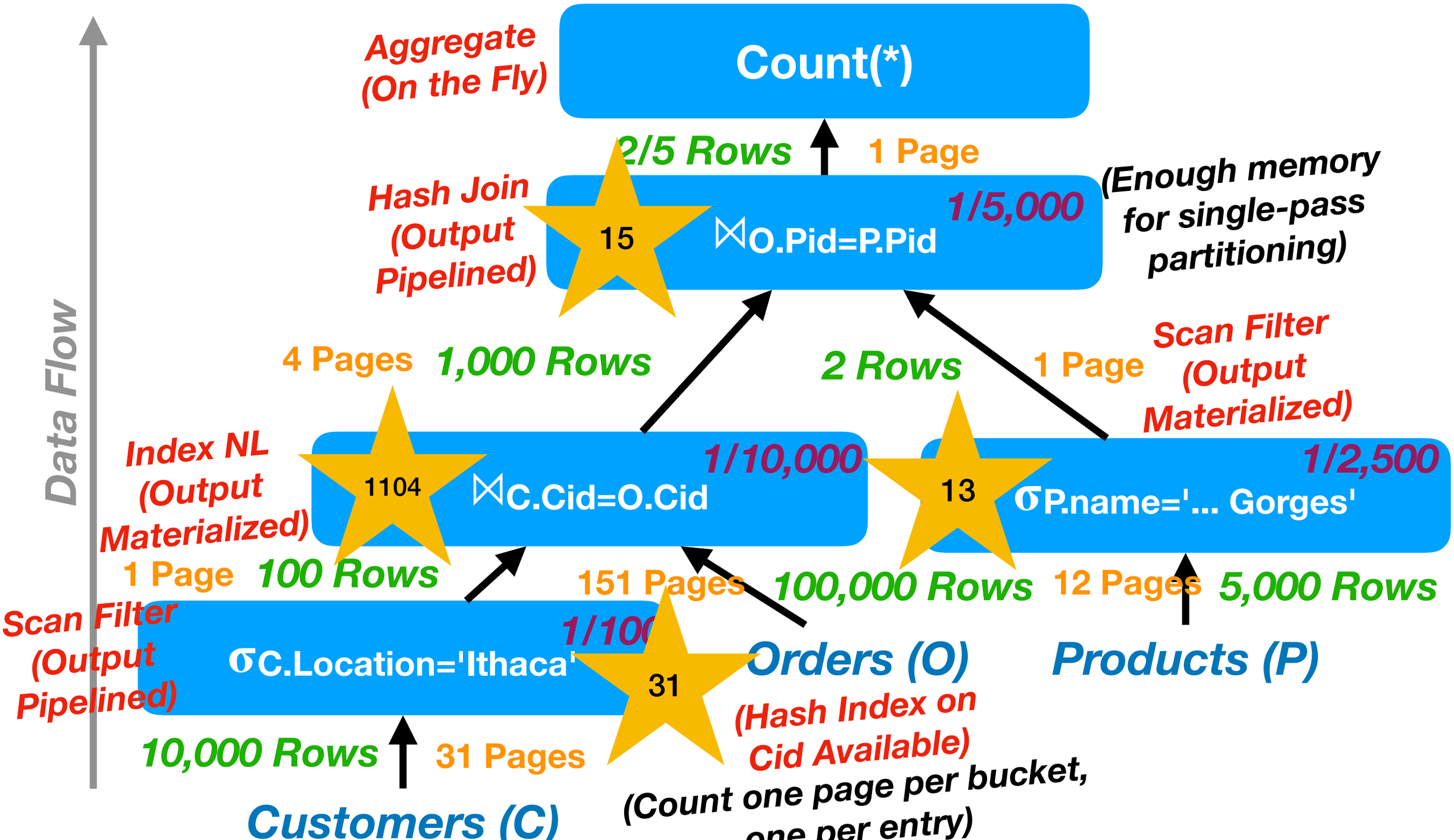
Cost Estimation



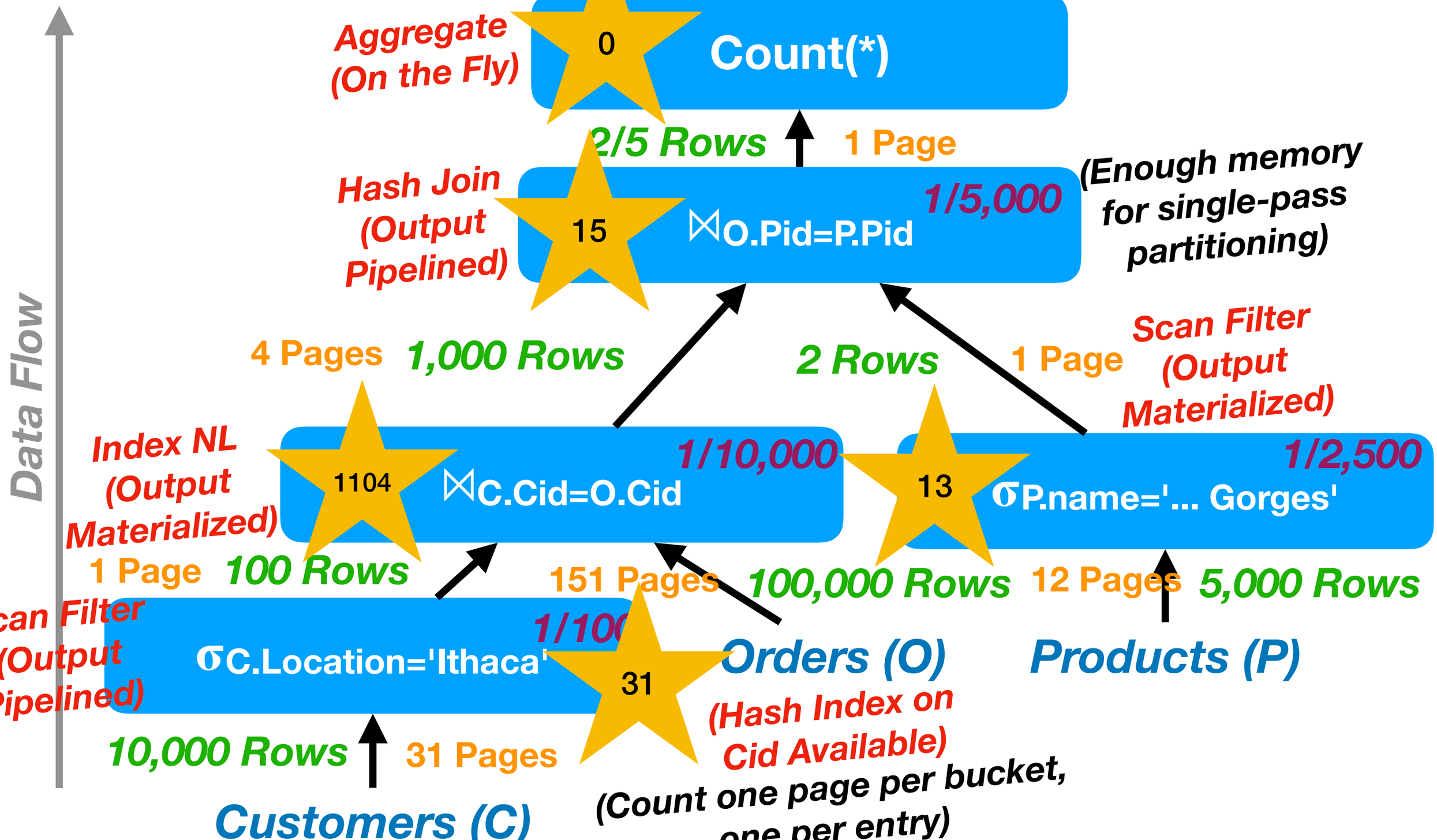
Cost Estimation



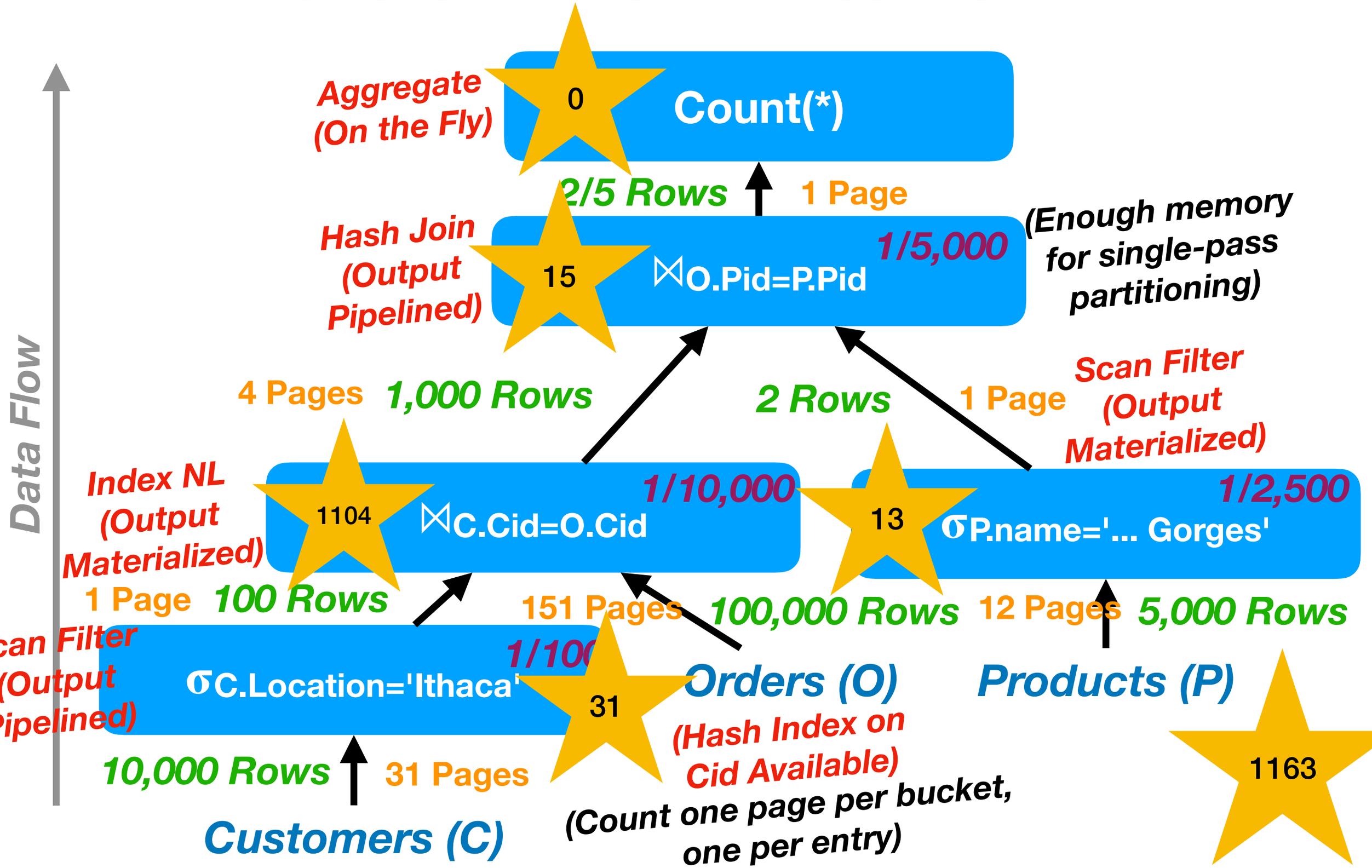
Cost Estimation



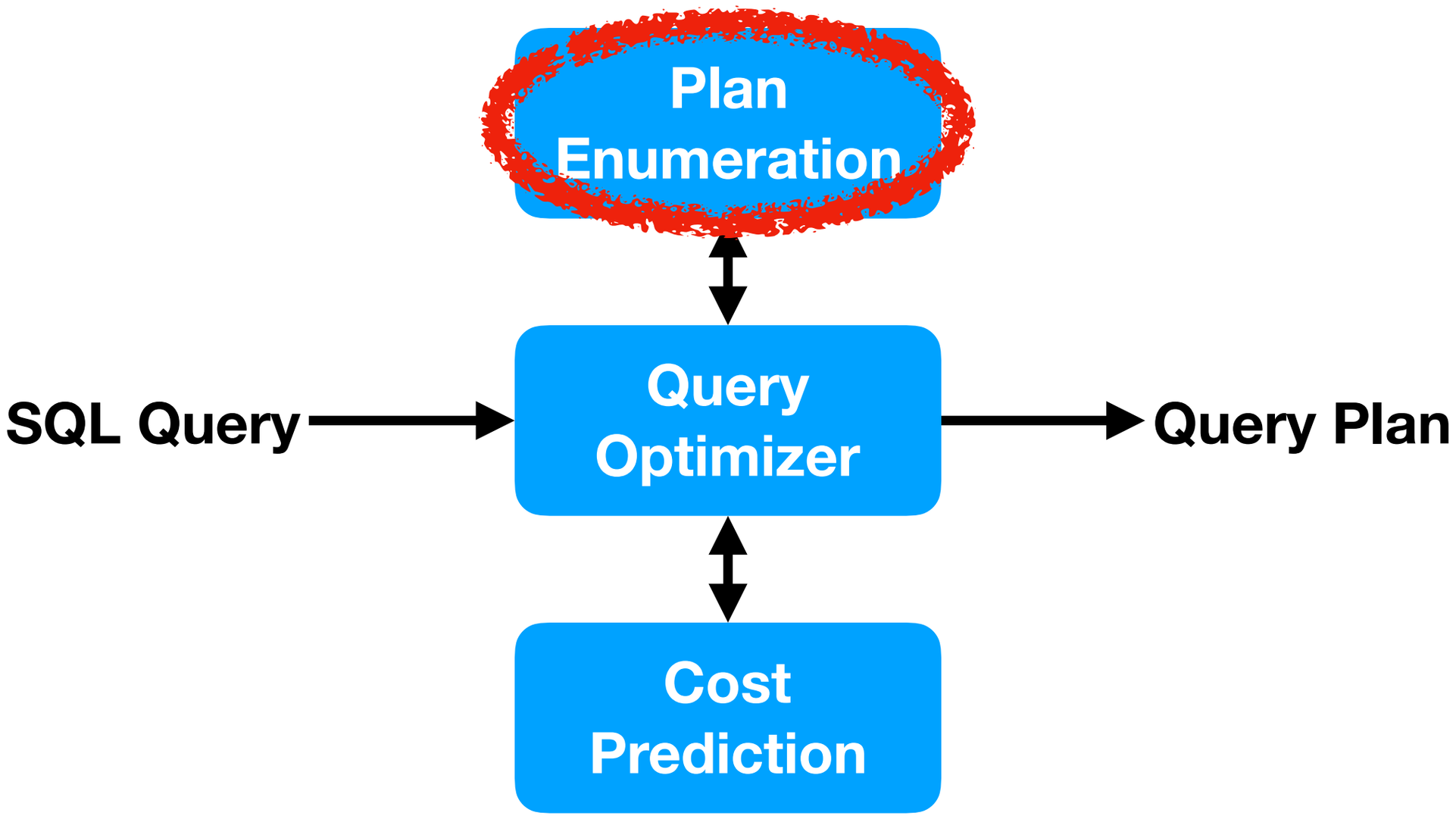
Cost Estimation



Cost Estimation



Optimizer Overview



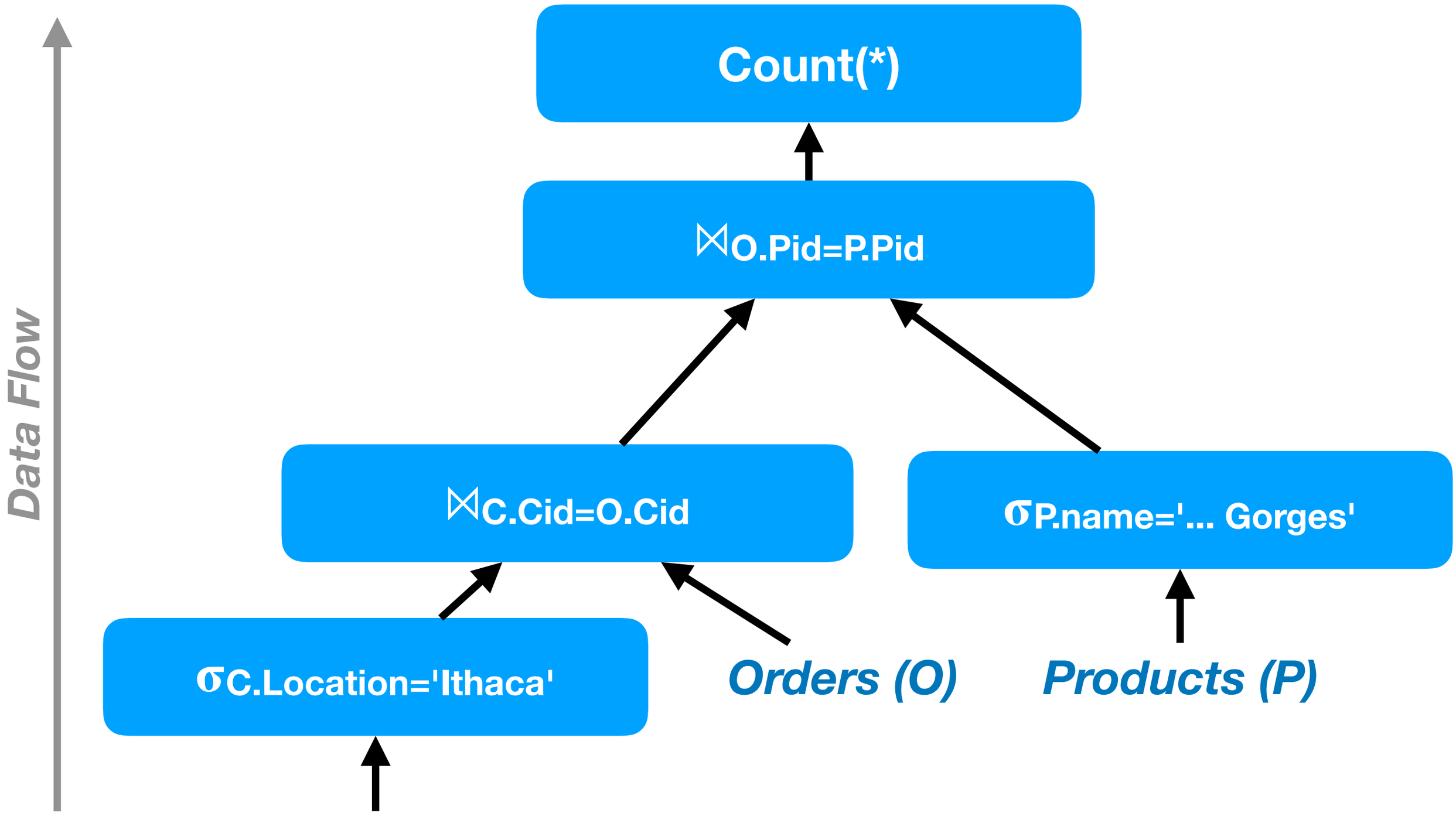
Query Plan Space

- Decide **order** of operations and **implementation**
- Apply **heuristic** restrictions
  - H1: Apply predicates/projections **early**
  - H2: Avoid **predicate-less** joins
  - H3: Focus on **left-deep plans**

H1: Early Predicates and Projections

- Processing **more** data is more expensive
- Want to **reduce** data size as quickly as possible
- Can do that by adding **predicates** (discarding rows)
- Can also do that by adding **projections** (discard columns)

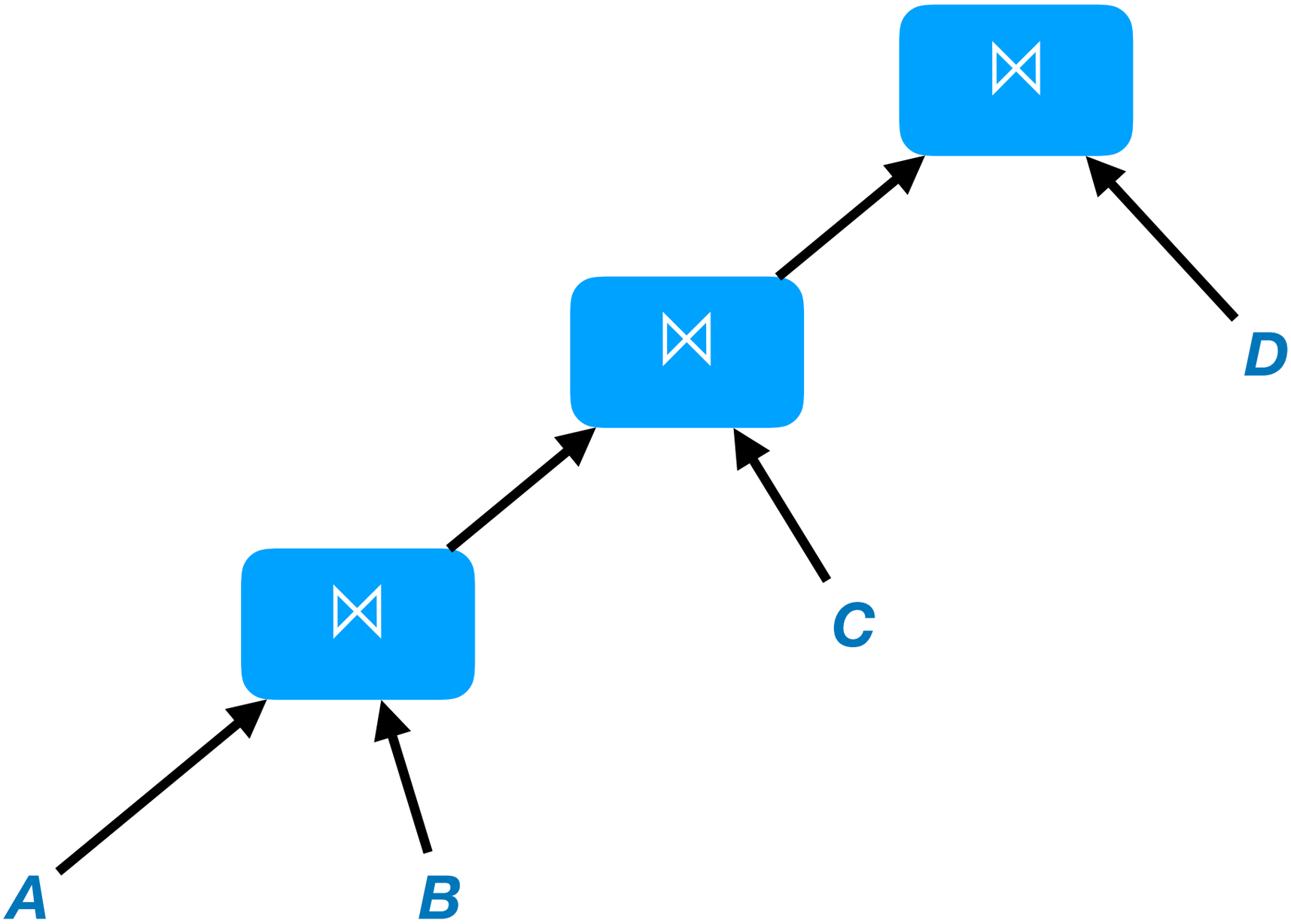
Can We Improve This?



H2: Avoid Joins without Predicates

- Join result **size**: product of input cardinality * selectivity
- Selectivity is **one** when joining tables without predicates
- Often means very **large** join results, probably sub-optimal
- (Heuristic may **discard** optimal order in special cases)

H3: Use "Left-Deep Plans"

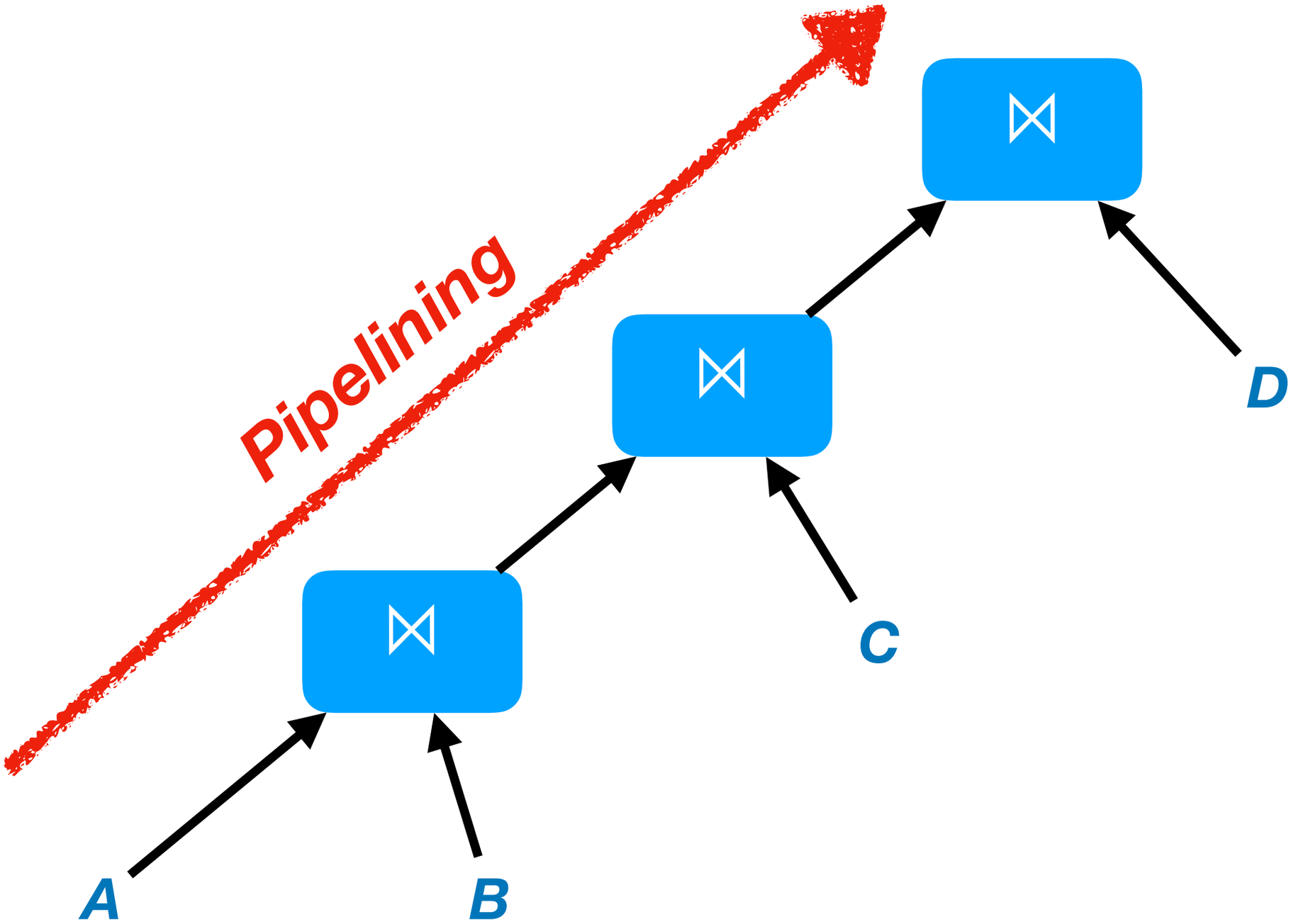


Why Left-Deep Plans?

- Allows **pipelining**: joins pass on result parts in-memory

- Allows to use **indices** on join columns (of base tables)

Focus on "Left-Deep Plans"



Naive Plan Enumeration

- **Generate** every possible plan
- **Estimate** cost for each plan
- **Select** plan with minimal cost

Asymptotic Analysis

- Number of join orders can grow quickly **O(nrTables!)**.
- Number of join orders **lower-bounds** number of plans
- Enumerating plans is considered **impractical**

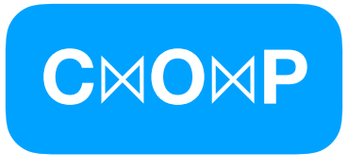
Principle of Optimality

- Want **cheapest** plan to join set of tables
- This plan joins **table subsets** "on the way"
- Assume we use **sub-optimal** plan for joining table subset
- Replacing by a **better** plan can only improve overall cost
- This is generally called the **principle of optimality**

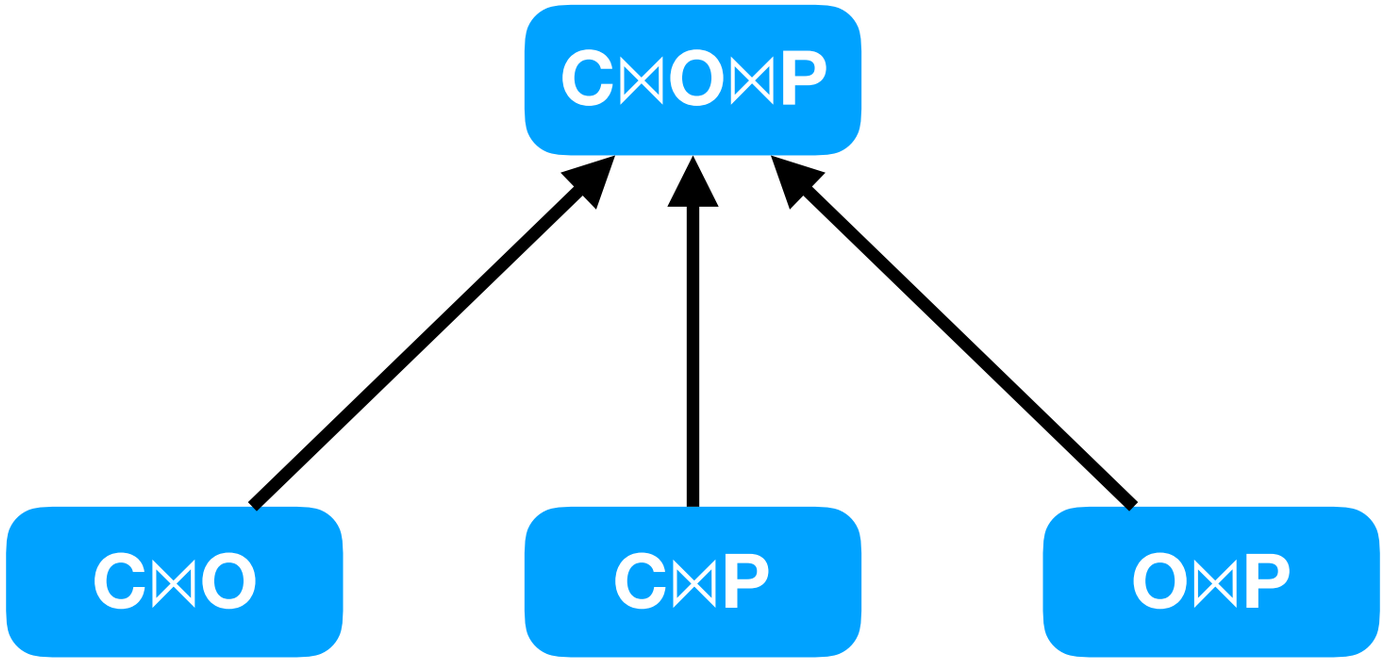
Efficient Optimization

- Find optimal plans for (smaller) sub-queries **first**
  - **Sub-query**: joins subsets of tables
- Compose optimal plans **from optimal sub-query plans**

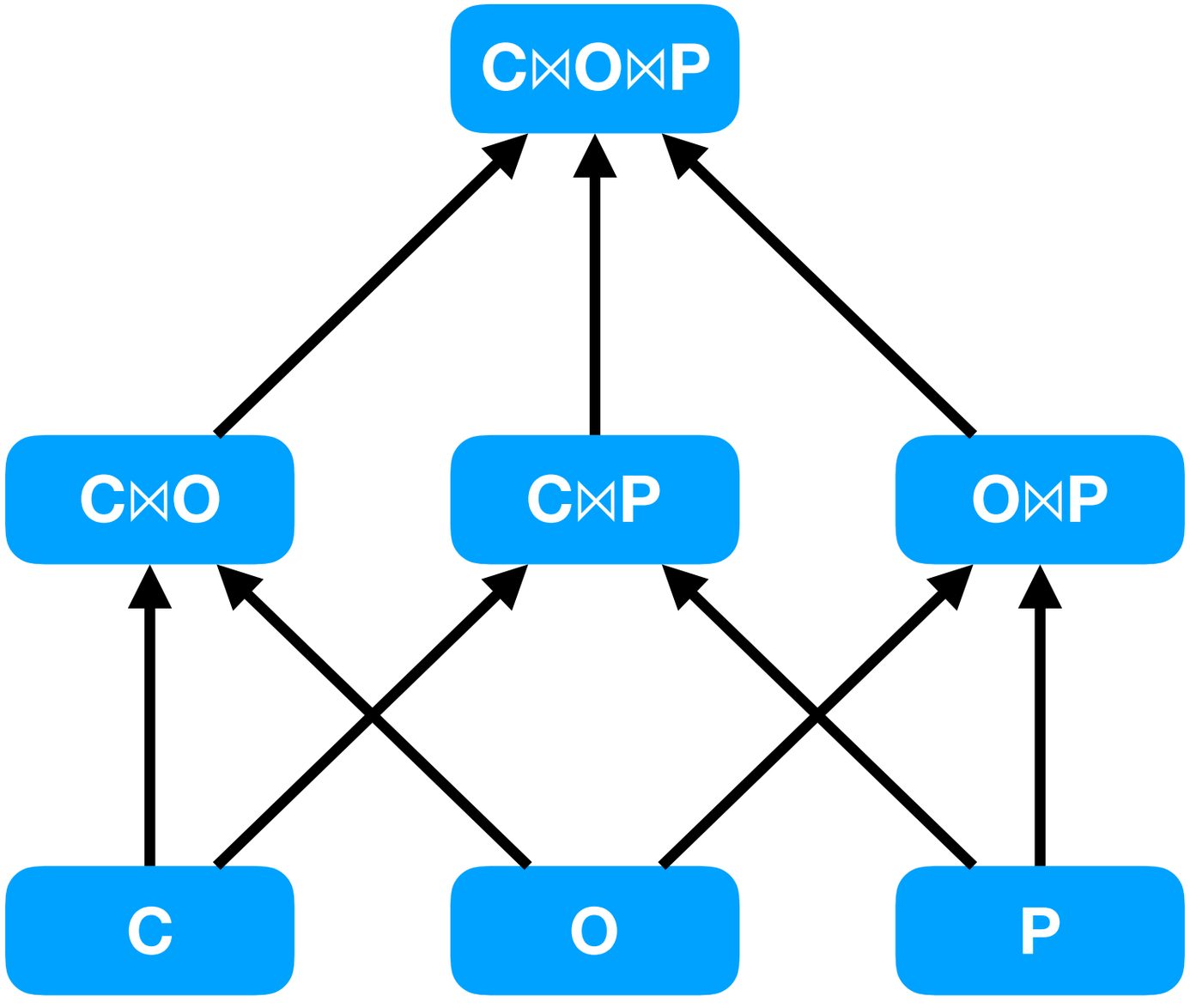
Dynamic Programming



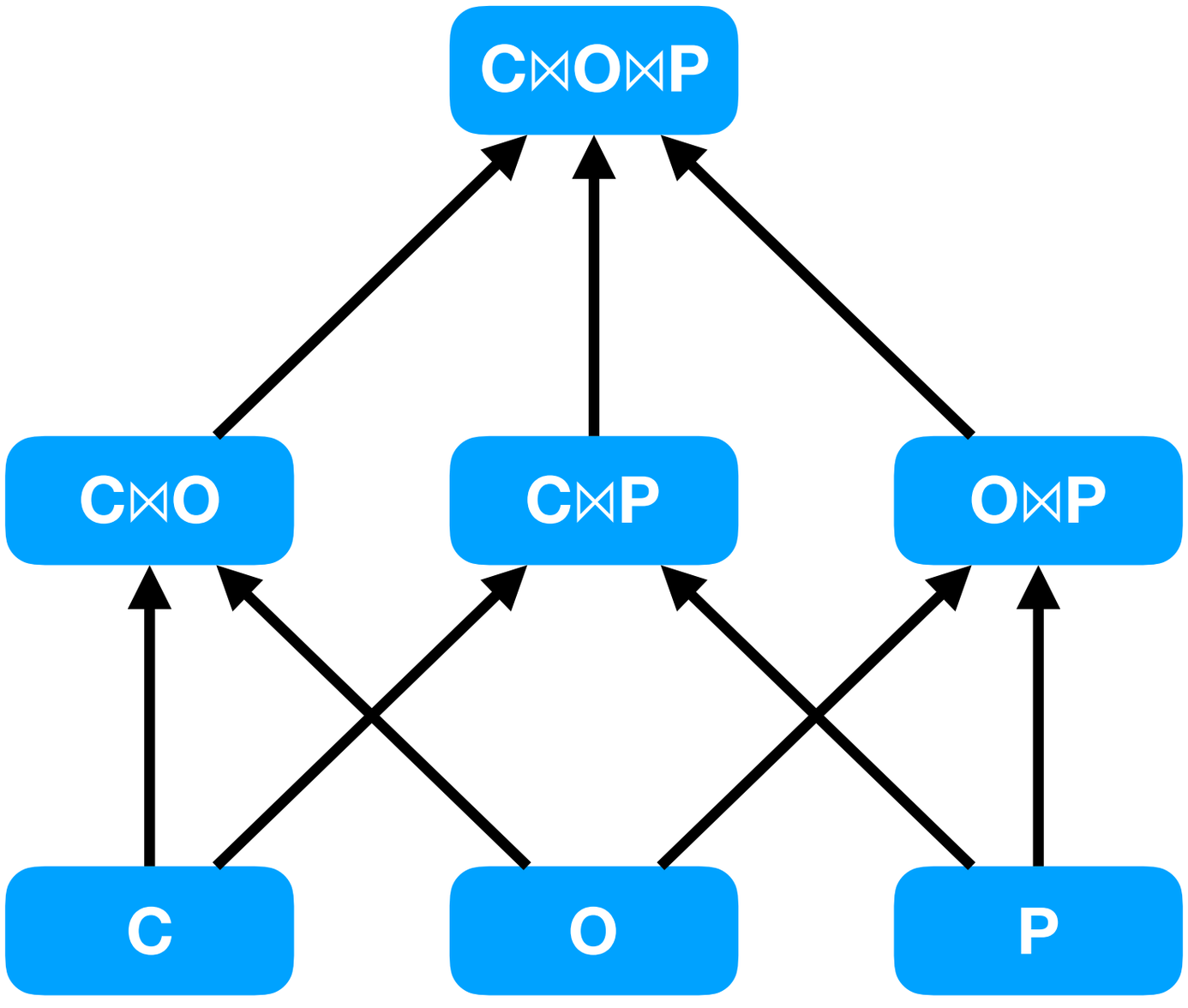
Dynamic Programming



Dynamic Programming

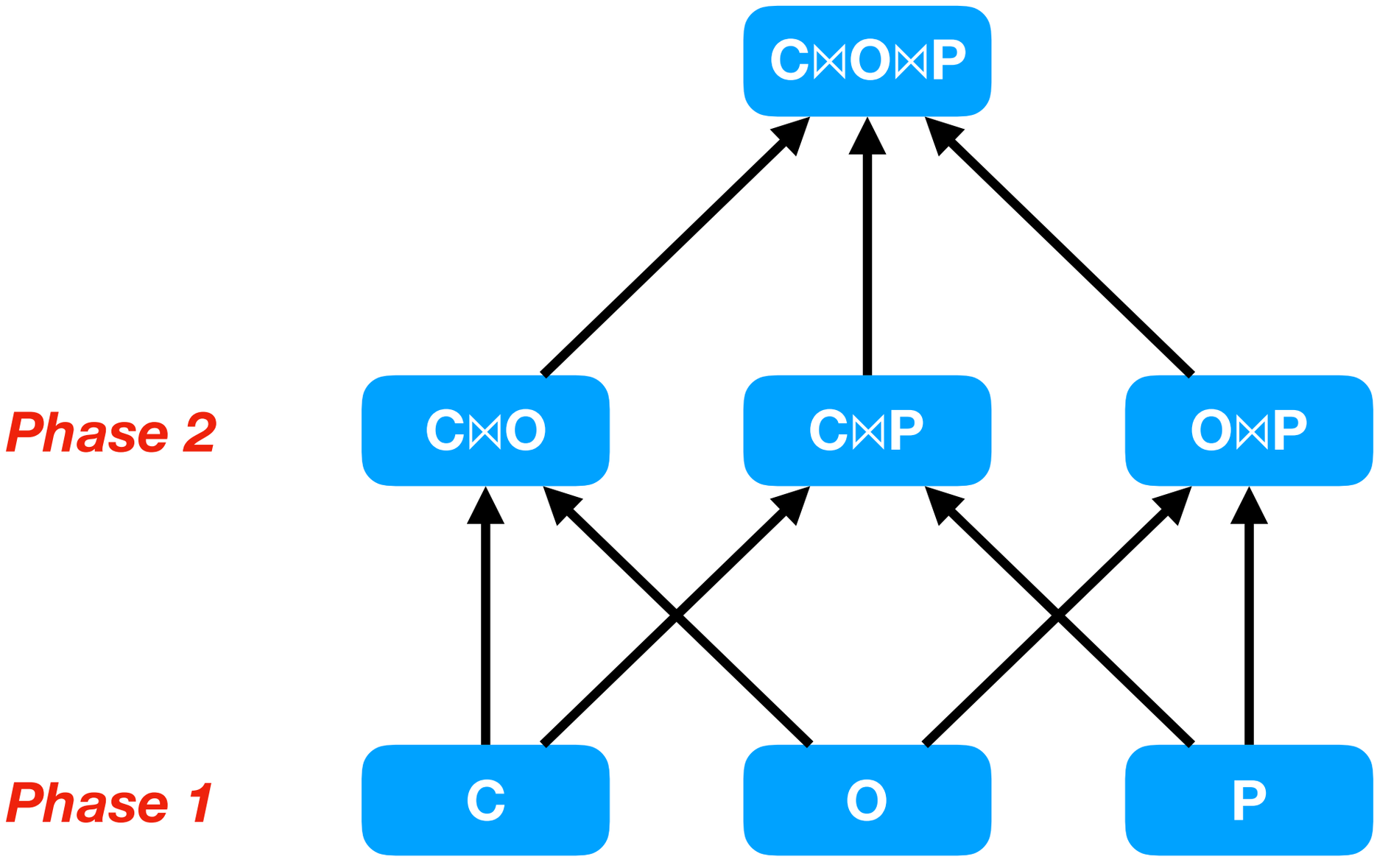


Dynamic Programming



Phase 1

Dynamic Programming

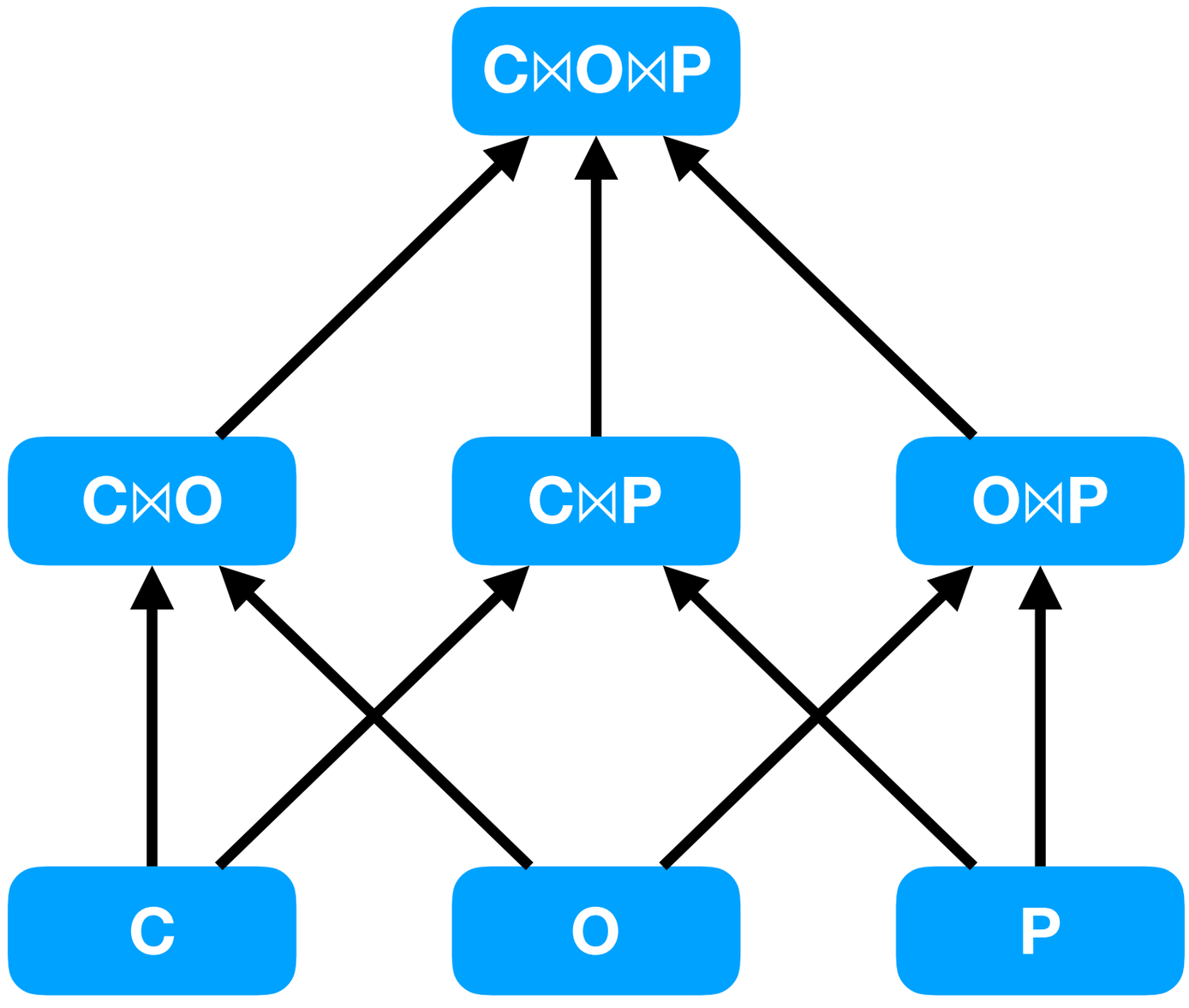


Dynamic Programming

Phase 3

Phase 2

Phase 1



Dynamic Programming