Query Processing 5: Query Optimization

Yufei Tao

https://www.cse.cuhk.edu.hk/~taoyf

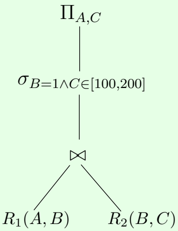
This lecture will provide an overview of **query optimization**, which is a process carried out by a database to select a query execution strategy.

Example

Relations $R_1(A, B)$ and $R_2(B, C)$.
SQL query:

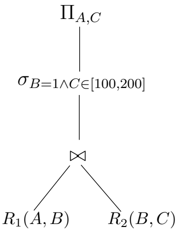
SELECT A, C FROM R, S
WHERE $R_1.B = R_2.B$ AND $S.B = 1$ AND $S.C$ BETWEEN 100 AND 200

A query plan:



Before running a query plan, the database always estimates its cost.

Assume:



- $B(R_1) = B(R_2) = 10000$ (i.e., each relation has 10000 blocks).
- $R_1 \bowtie R_2$ has B_1 blocks.
- $\sigma_{B=1\wedge C\in[100,200]} R_1 \bowtie R_2$ has B_2 blocks.
- $M = 101$ (the memory has 101 blocks).

Strategy 1:

- Compute $R_3 = R_1 \bowtie R_2$ with BNL and **materialize** R_3 (i.e., write R_3 to the disk) $\Rightarrow 10^4 + 10^6$ I/Os (BNL) + B_1 (mat.).
- Compute $R_4 = \sigma_{B=1\wedge C\in[100,200](R_3)}$ by reading R_3 and materializing $R_4 \Rightarrow B_1$ (reading) + B_2 (materialization) I/Os.
- Compute $\Pi_{A,C}(R_4)$ by reading $R_4 \Rightarrow B_2$ I/Os.

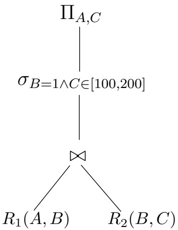
In total: $1010000 + 2(B_1 + B_2)$ I/Os.

Next, we will see several generic methods for improving the strategy.

Method 1: Algorithm Selection

Assume:

- $B(R_1) = B(R_2) = 10000$
- $R_1 \bowtie R_2$ has B_1 blocks.
- $\sigma_{B=1 \wedge C \in [100,200]} R_1 \bowtie R_2$ has B_2 blocks.
- $M = 101$.



Strategy 2:

- Compute $R_3 = R_1 \bowtie R_2$ with **sort join** and materialize $R_3 \Rightarrow 5(10^4 + 10^4)$ I/Os (**under no-skew assumption**) $+ B_1$ (mat.).
- Compute $R_4 = \sigma_{B=1 \wedge C \in [100,200](R_3)}$ by reading R_3 and materializing $R_4 \Rightarrow B_1$ (reading) $+ B_2$ (materialization) I/Os.
- Compute $\Pi_{A,C}(R_4)$ by reading $R_4 \Rightarrow B_2$ I/Os.

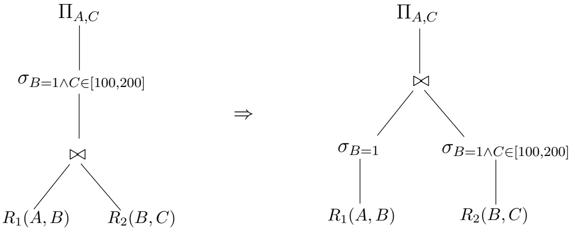
In total: $100000 + 2(B_1 + B_2)$ I/Os.

Method 2: Query Rewriting

Observe:

$$\Pi_{A,C}(\sigma_{B=1 \land C \in [100,200]}(R_1 \bowtie R_2)$$
$$= \Pi_{A,C}(\sigma_{B=1}(R_1) \bowtie \sigma_{B=1 \land C \in [100,200]}(R_2))$$

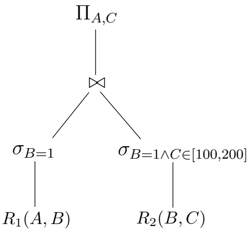
Query rewriting converts the original query to an equivalent query using laws of relational algebra.



Rule of thumb: In practice, selections are almost always pushed down as much as possible.

Assume:

- $B(R_1) = B(R_2) = 10000$
- $\sigma_{B=1} R_1$ has 1000 blocks
- $\sigma_{B=1 \wedge C \in [100,200]} R_2$ has 800 blocks
- $\sigma_{B=1 \wedge C \in [100,200]} R_1 \bowtie R_2$ has B_2 blocks.
- $M = 101$.



Strategy 3:

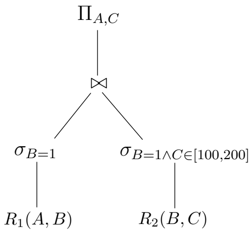
- Compute $R_3 = \sigma_{B=1} R_1$ by reading R_1 and materializing R_3 $\Rightarrow$ 10000 (reading) + 1000 (mat.) I/Os
- Compute $R_4 = \sigma_{B=1 \wedge C \in [100,200]} R_2$ by reading R_2 and materializing R_4 $\Rightarrow$ 10000 (reading) + 800 (mat.) I/Os
- Compute $R_5 = R_3 \bowtie R_4$ with sort join and materialize R_5 $\Rightarrow$ $5(1000 + 800)$ I/Os (under no-skew assumption) + B_2 (mat.).
- Compute $\Pi_{A,C}(R_5)$ by reading R_5 $\Rightarrow$ B_2 I/Os.

In total: $30800 + 2B_2$ I/Os.

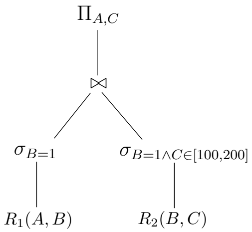
Method 3: Applying Indexes

Assume:

- $B(R_1) = B(R_2) = 10000$
- $\sigma_{B=1} R_1$ has 1000 blocks
- $\sigma_{C \in [100,200]} R_2$ has 3000 blocks
- R_2 is **clustered** on C.
  There is a **B-tree** on $R_2.C$ with 3 levels.
- $\sigma_{B=1 \wedge C \in [100,200]} R_2$ has 800 blocks
- $\sigma_{B=1 \wedge C \in [100,200]} R_1 \bowtie R_2$ has B_2 blocks.
- $M = 101$.



Strategy 4:



- Compute $R_3 = \sigma_{B=1} R_1$ by reading R_1 and materializing R_3 $\Rightarrow$ 10000 (reading) + 1000 (mat.) I/Os
- Compute $R_4 = \sigma_{B=1 \wedge C \in [100,200]} R_2$ by using the B-tree on $R_2.C$ and materialize $R_4 \Rightarrow 3$ (B-tree) + 3000 (retrieving $\sigma_{C \in [100,200]} R_2$) + 800 (mat.) I/Os
- Compute $R_5 = R_3 \bowtie R_4$ with sort join and materialize $R_5 \Rightarrow$ $5(1000 + 800)$ I/Os (under no-skew assumption) + B_2 (mat.).
- Compute $\Pi_{A,C}(R_5)$ by reading $R_5 \Rightarrow B_2$ I/Os.

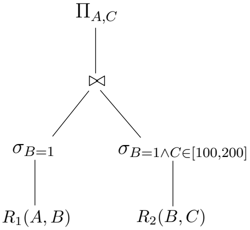
In total: $23803 + 2B_2$ I/Os.

Method 4: Pipelining

Pipelining processes multiple operations in a query plan **together**.

Assume:

- $B(R_1) = B(R_2) = 10000$
- $\sigma_{B=1} R_1$ has 1000 blocks
- $\sigma_{B=1 \wedge C \in [100,200]} R_2$ has 800 blocks
- $M = 101$.



Strategy 5:

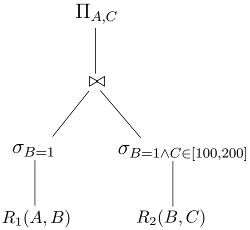
- Compute $R_3 = \sigma_{B=1} R_1$ by reading R_1 and materializing R_3 $\Rightarrow$ 10000 (reading) + 1000 (mat.) I/Os
- Compute $R_4 = \sigma_{B=1 \wedge C \in [100,200]} R_2$ by reading R_2 and materializing $R_4 \Rightarrow$ 10000 (reading) + 800 (mat.) I/Os
- Compute $R_5 = R_3 \bowtie R_4$ with sort join.
  For each tuple $t \in R_5$ found in memory, output directly $t.A$.
  $\Rightarrow 5(1000 + 800)$ I/Os (under no-skew assumption).

In total: 30800 I/Os.

Pipelining may avoid materializing intermediate results altogether.

Assume:

- $B(R_1) = B(R_2) = 10000$
- $\sigma_{B=1} R_1$ has 1000 blocks. R_1 is **clustered** on B and there is a **hash index** on $R_1.B$.
- $\sigma_{B=1 \wedge C \in [100,200]} R_2$ has 800 blocks
- $M = 101$.



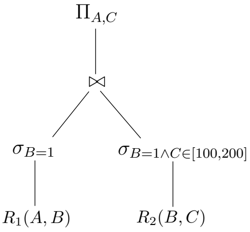
Strategy 6:

- Compute $R_3 = \sigma_{B=1 \wedge C \in [100,200]} R_2$ by reading $R_2 \Rightarrow 10000$ I/Os.
- Using the hash index on R_1, we can find $R_4 = \sigma_{B=1} R_1$ in $1 + 1000 = 1001$ I/Os.
- **As soon as getting a tuple $t \in R_3$ in memory**: probe the hash index on $R_1.B$ to find all tuples $s \in \sigma_{B=1} R_1$; output $(s.A, t.C)$.
  $\Rightarrow |R_3| \cdot 1001$ I/Os.

In total: $10000 + 1001|R_3|$ I/Os.

Assume:

- $B(R_1) = B(R_2) = 10000$
- $\sigma_{B=1} R_1$ has 1000 blocks. R_1 is **clustered** on B and there is a **hash index** on $R_1.B$.
- $\sigma_{B=1 \wedge C \in [100,200]} R_2$ has 800 blocks
- $M = 101$.



Strategy 7:

- Compute $R_3 = \sigma_{B=1 \wedge C \in [100,200]} R_2$ by reading $R_2 \Rightarrow 10000$ I/Os.
- **Every time we have accumulated $M - 1 = 100$ blocks of new tuples of R_3 in memory**, probe the hash index on $R_1.B$ to find $R_4 = \sigma_{B=1} R_1$; output $(s.A, t.C)$ for each $t \in R_3$ in memory and every $s \in R_4$
  $\Rightarrow \lceil 800/100 \rceil \cdot 1001 = 8008$ I/Os.

In total: 18008 I/Os.

Query optimization is an art. It is an active research area.

The methods we discussed are representative but far from being comprehensive. The query optimization module of a database system is a highly sophisticated and often kept as a commercial secret.