資料庫管理 HW05

B12508026 戴偉璿

November 30, 2025

1. If we read the outer loop into $B-2$ blocks, the total cost is M(number of outer blocks)+$\lceil \frac{M}{B-2} \rceil \times N$(number of inner loop read times). Thus, the total cost is $\lceil \frac{M}{B-2} \rceil \times N + M$ I/Os. In the other way, if we read the inner loop into $B-2$ blocks, the total cost is $N + \lceil \frac{N}{B-2} \rceil \times M$ I/Os. Therefore, the total cost is $\min(\lceil \frac{M}{B-2} \rceil \times N + M, N + \lceil \frac{N}{B-2} \rceil \times M)$ I/Os. Thus, these two methods are roughly equal. However, in practical, we usually put smaller relation in the outer loop to reduce the number of passes of the inner loop. In this case, place the smaller relation into $B-2$ blocks is better.

2.

   (a) I don't think this plan is more efficient. There are no join keys between `PRODUCT` and `SALES`. If we join these two relations first, it would cause cartesian product, which would produce a very large intermediate relation.

   (b) To achieve projection pushdown, I project the needed attributes at the first step. Next step would be handle the predicate pushdown. I apply the selection `s.store_id=1` and `sd,unit_price>=20` at this phase. Now I need to decide which join should be performed first. I have two joins: `PRODUCT` $\bowtie$ `SALES_DETAIL` and `SALES` $\bowtie$ `SALES_DETAIL`. Any way, `SALES_DETAIL` must be joined first. Thus, I need to choose between `PRODUCT` $\bowtie$ `SALES_DETAIL` and `SALES` $\bowtie$ `SALES_DETAIL`. Finally I decide to join `SALES` and `SALES_DETAIL` first because there is a selection `s.store_id=1` on `SALES`, which would reduce the number of tuples in the intermediate relation. Besides, the join key `receipt_no` may be more selective than `product_id`. The overall plan is shown as Fig .1.

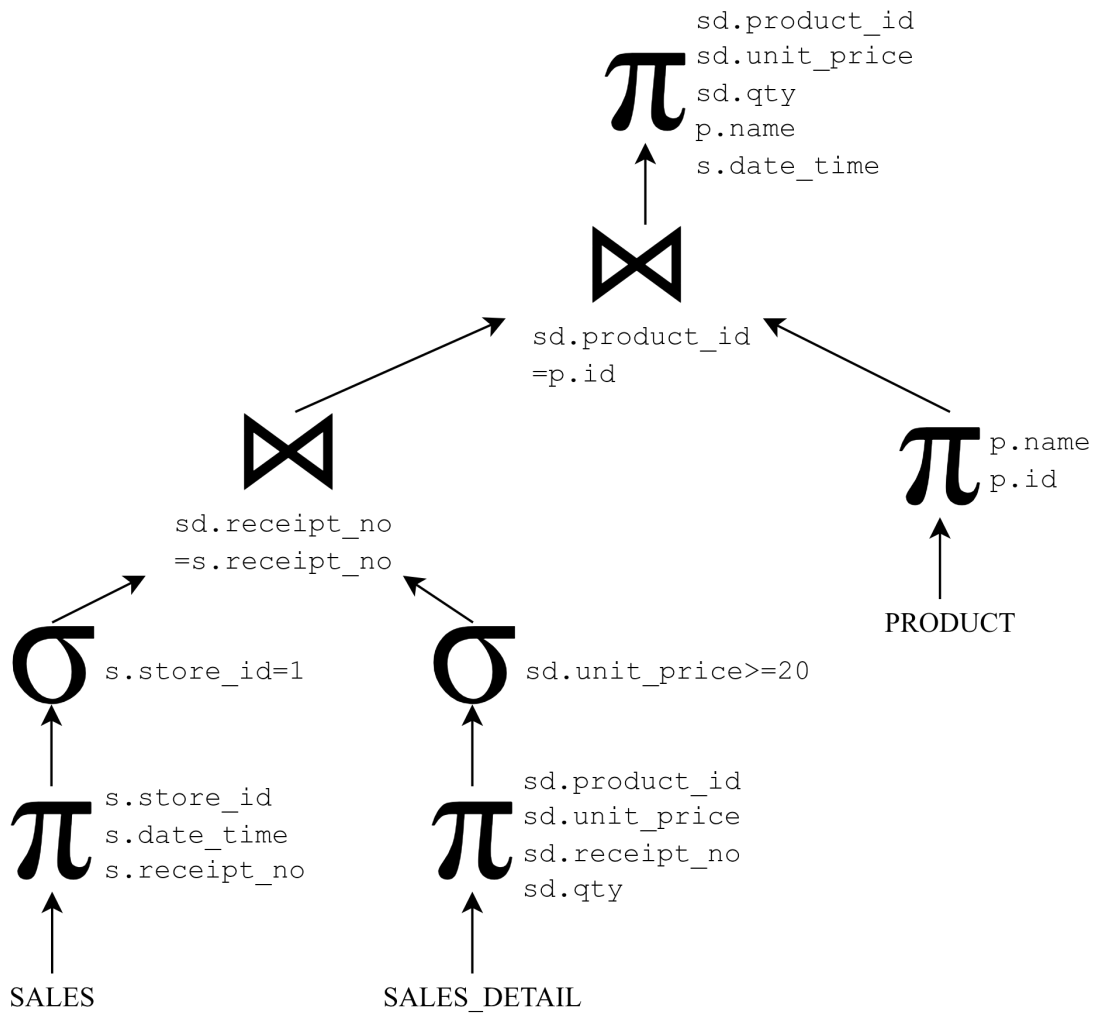

Figure 1: Query Plan with Projection and Predicate Pushdown

(c) The query plan generated by postgreSQL is shown as Fig .2.

```
                                  QUERY PLAN
------------------------------------------------------------------------------
 Nested Loop  (cost=21.71..24.47 rows=1 width=138)
   ->  Hash Join  (cost=21.56..23.61 rows=1 width=20)
         Hash Cond: ((sd.receipt_no)::text = (s.receipt_no)::text)
         ->  Seq Scan on sales_detail sd  (cost=0.00..1.86 rows=69 width=23)
               Filter: (unit_price >= 20)
         ->  Hash  (cost=21.50..21.50 rows=5 width=56)
               ->  Seq Scan on sales s  (cost=0.00..21.50 rows=5 width=56)
                     Filter: (store_id = 1)
   ->  Index Scan using product_pkey on product p  (cost=0.15..0.86 rows=1 width=122)
         Index Cond: (id = sd.product_id)
(10 rows)
```

Figure 2: Query Plan Generated by PostgreSQL

The DBMS first pushes down the filters so that the tables become smaller before

any joins. Then it estimates how many rows remain after each step and chooses the join order that keeps the intermediate results as small as possible. This is why it joins `SALES_DETAIL` and `SALES` first using a Hash Join, and only later joins `PRODUCT` using an index.