Contents

1 Intro

Just use the following query on the HUJI DB:

Table 1: Query

```
select * from pg_stats where schemaname = 'danielkerbel'
                        and tablename='actors'
```

Replacae *schemaname* with your own CSE username, make sure you have the actors table in the DB.

And take the appropriate results:

tablename	attname	inherited	null_frac	avg_width	n_distinct	most_common_vals	most_common_freqs
actors	actorid	☐	0	4	-1	[NULL]	NULL
actors	name	☐	0	14	-0.99823588	{"Frank Clark","Frederick Lewis","Frederick Truesdell","G	{2.7140725E-4,2.7140725E-4,2.7140725E-4,2.7140725E-4,2
actors	byear	☐	0.34482291	4	92	{1884,1889,1885,1886,1882,1888,1887,1880,1878,1890,189	{0.022662506,0.02211969,0.02198398?,0.021576876,0.0207
actors	dyear	☐	0.36422852	4	101	{1941,1945,1943,1937,1940,1949,1933,1942,1953,1944,194	{0.014791695,0.0143845845,0.014248881,0.013570363,0.01

Figure 1: Query results

2 Size Estimation

2.1 q1

- All red parts
  - Num Rows: $10,000 \cdot \frac{1}{10} = 1,000$
  - Num Blocks: There are $\left\lfloor \frac{8192}{24} \right\rfloor = 341$ rows per block, thus $\left\lceil \frac{1,000}{341} \right\rceil = 3$ blocks
- All pipes
  - Num Rows: $10,000 \cdot \frac{1}{1000} = 10$
  - Num blocks: One block
- Red pipes
  - Num Rows: $10,000 \cdot \frac{1}{10 \cdot 1000} = 1$
  - One block
- Red items or pipes
  - Note, $P(\text{NotRed} \wedge \text{NotPipe}) \overset{\text{independence}}{=} \left(1 - \frac{1}{10}\right)\left(1 - \frac{1}{1000}\right)$
    Thus $P(\text{Red} \vee \text{Pipe}) = 1 - P(\text{NotRed} \wedge \text{NotPipe}) = 1 - \left(1 - \frac{1}{10}\right)\left(1 - \frac{1}{1000}\right)$
    Thus $T(\text{Red} \vee \text{Pipe}) = 10,000 \cdot \left(1 - \left(1 - \frac{1}{10}\right)\left(1 - \frac{1}{1000}\right)\right) = 1009$
  - Num blocks is $\left\lceil \frac{1009}{341} \right\rceil = 3$
- PIDs of red parts
  - Number of rows is identical, $1,000$
  - Since we're only taking PIDs, each block can contain $\left\lfloor \frac{8192}{4} \right\rfloor = 2048$ PIDs, thus
    the number of blocks needed is $\left\lceil \frac{1000}{2048} \right\rceil = 1$

2.2 q2

- All 5 properties, natural join
  - Since each catalog entry has a part(pid is a foreign key in Catalog, primary in Part), the
    number of rows is $T(Catalog) = 100,000$
  - Each row contains $4 + 10 + 10 + 4 + 4 = 32$ bytes, so a block contains $\frac{8192}{32} = 256$ rows
    Therefore we need $\left\lceil \frac{100,000}{256} \right\rceil = 391$ blocks
- Natural join, projecting pid, pname, price

- The number of rows is un-changed, $100,000$
- Each row contains $4 + 10 + 4 = 18$ bytes, a block contains $\lfloor \frac{8192}{18} \rfloor = 455$ rows,
  Therefore we need $\lceil \frac{100,000}{455} \rceil = 220$ blocks

- Natural join, filter red parts, all 5 properties
  - Denote $FilteredParts$ as the table we're joining $Catalog$ with, so we have the following number of rows:

$$\frac{T(Catalog) \cdot T(FilteredParts)}{\max\{V(Catalog, pid), V(FilteredParts, pid)\}}$$

    Note that $T(FilteredParts) = \frac{10,000}{10} = 1,000$ (by the given color distribution), and that

$$V(Catalog, pid) = T(Parts)$$
$$V(FilteredParts, pid) = T(FilteredParts)$$

    as pid is a primary key of $Catalog$, so plugging these into the formula we get:

$$\frac{100,000 \cdot 1,000}{\max\{10,000 \quad, 1,000\}} = 10,000$$

  *__Note:__ in the lecture there's a different formula that obtains the same result (first natural join, then filter)

  - There are 32 bytes per row, so a block contains 256 rows(see before), so we need $\lceil \frac{10,000}{256} \rceil = 40$ blocks

- Natural join, filter red parts and prices below 25, all 5 properties
  - Using an idea similar to the formula in the lecture, denote $Temp$ as the natural join of $Catalog, Parts$ with red parts, this is what we have above, that is, $T(Temp) = 10,000$, now we need to compose another filter, note that $P(CatalogPriceBelow25) = \frac{1}{3}$ (remember we have a default of $\frac{1}{3}$ when the distribution isn't known, as told in the lecture), so the calculation is $T(Temp) \cdot P(CatalogPriceBelow25) = 10,000 \cdot \frac{1}{3} = 3333.33$
    (The final answer is **3334**, I don't know why they rounded the number up and not down).
  - Like before, a block contains 256 rows, so we need $\lceil \frac{3,334}{256} \rceil = 14$

3 Query Plans

Note, I number queries in q1, q2 by the order they appear. In case moodle randomizes the query order, I will attach a PDF with the scan of this quiz

3.1 Question 1

1. This works - similar to what we saw on the lecture at 05:22, simple BNL, **yes**
2. This works, so **yes** (note that 5 is better as we can push the filter on prices inside)
3. This doesn't work - we project $Parts.pname$ **before** doing a natural join via pid, impossible - so **no**
4. This works, we pushed all projections and filters inside as much as we can(while still retaining pid needed for the natural join), **yes**
5. This works and is similar to 2, but better in every way(pushed all filters inside), so **yes**
6. This is very similar to 4, but doesn't give us what is required - it returns tuples of $(pname, pid)$ and we are only interested in $pname$, so **no**

3.2 Question 2 - fully pushed query plans

1. This works, but is not fully pushed, so **no**
2. This works, but is not fully pushed, so **no**
3. This doesn't work, just like 3 in Question 1, so **no**
4. Just like 4, this works and is fully pushed, so **yes**
5. We can push the projection of $pname$ inside(turn it into $\sigma_{pname,pid}$, similar to 6) and then project $pname$ after the join,

   similar to 4, so **no** this isn't fully pushed.
6. Just like 6 from before, it doesn't give us what we want, so **no**

3.3 Question 3

1. No indices

   - We can use BNL, and we can switch between inner/outer relation(2 options), but for each filter there's only full table scan, so 2 options for this join
   - We can use SM, we don't need to switch betweem inner/outer as this is symmetric, and only full table scan each filter, so only 1 option
   - We can use HJ, like above, no need to switch between inner/outer, so only 1 option
   - We cannot use INL

   So the answer is $2 + 1 + 1 = \mathbf{4}$

   So the answer is **3**

2. Index on pid, so we can now do an INL(Index Nested Loop) join with $Parts$ as the inner relation, giving us another option so $4 + 1 = \mathbf{5}$

3. Now that there's only an index on $Color$, we only have the 3 joins BNL, SM, HJ, but now for each configuration(join type, join order) we can choose whether to use a full table scan or index scan on $\sigma_{color='red'}$, so the answer is $4 \cdot 2 = \mathbf{8}$

4. Answer is identical to (3) (we choose whether to full table scan or index scan on $\sigma_{price<20}$), so $4 \cdot 2 = \mathbf{8}$

5. Now that we have indices on both $Color, price$, we still have 3 types of joins, and each configuration has 4 options (1 - full table scan for both $color, price$,

   2 - full table scan for $color$, index scan for $price$, 3 - index scan for $color$, full table scan for $price$, 4- index scan for both $color, price$)

   So the answer is $4 \cdot 4 = \mathbf{16}$

6. Now that we have indices on both $Catalog.price$ and $Parts.pid$, we still have all options from (4), in addition:

   - We can use INL, and $Parts$ must be the inner relation in the join. Like in (4), we have 2 options for filtering $\sigma_{price<20}$ (full table scan or index scan) giving us a total of 2 options

   So the answer is $8 + 2 = \mathbf{10}$

7. Now that we have 2 separate indices on $Parts.pid$ and $Parts.color$, we still have all options from (3), in addition:

   - We can use INL, and $Parts$ must be the inner relation in the join. Note that we can't push $\sigma_{color='red'}$ before the join, nor do it during the join(since this is not a 2-column index),

     So we'll have $\sigma_{color='red'}$ after the INL join. One might think there are 2 options for said projection(full table scan vs index scan), but actually, that projection is done when emitting the results

     of the INL join(this is the option that is "most pushed"), so it doesn't matter that we have an index on $Parts.color$.

   So, the answer is $8 + 1 = \mathbf{9}$

4 Join Costs

Note that in all of the following queries, we don't need to calculate the price of projections and filterings done after the join,

because that can be done as part of the join, right before we output its results.

4.1 Query plan 1

The cost of BNL, where Parts is outer and Catalog is inner, price of join is

$$B(Parts) + B(Catalog) \left\lceil \frac{B(Parts)}{M-2} \right\rceil$$

Note that a block can contain $\left\lfloor \frac{8192}{4+10+10} \right\rfloor = 341$ entries of $Parts$, thus $B(Parts) = \lceil \frac{10{,}000}{341} \rceil = 30$

Likewise, a block can contain $\left\lfloor \frac{8192}{4+4+4} \right\rfloor = 682$ entries of $Catalog$, thus $B(Catalog) = \lceil \frac{100{,}000}{682} \rceil = 147$

So, the cost of the join is

$$30 + 147 \left\lceil \frac{30}{13} \right\rceil = 471$$

The rest of the query(filtering and projection) have a cost of 0, since they have been done right before writing the output of the BNL algorithm

Then, when filtering, denote $E = \pi_{pname}, \sigma_{color='red' \wedge price<20}$

Note that

$$Read(E) = B(JoinOutput)$$

How many blocks are outputted by the join? Each row contains $(pid, sid, pname, color, price)$, that is $4 + 4 + 10 + 10 + 4 = 32$ bytes,

so a block contains $\lfloor \frac{8192}{32} \rfloor = 256$ rows, there can be up to $100,000$ rows in the join output, so we'd have $\lceil \frac{100{,}000}{256} \rceil = 391$ blocks to read

So, $Read(E) = 391$

4.2 Query plan 2

Remember there are 2 forms of SMJ with different efficiency, lets see which is more efficient:

$$\begin{aligned} Read\left(\sigma_{color='red}(Parts)\right) &\overset{\text{full table scan}}{=} Read(Parts) = B(Parts) = 30 \\ Read(Catalog) &= B(Catalog) = 147 \\ T\left(\sigma_{color='red}(Parts)\right) &\overset{\text{given distribution}}{=} \frac{10,000}{10} = 1,000 \end{aligned}$$

A block of filtered parts contains $\left\lfloor \frac{8192}{4+10+10} \right\rfloor = 341$ rows, so we have $B\left(\sigma_{color='red}(Parts)\right) = \lceil \frac{1{,}000}{341} \rceil = 3$

Note that $\left\lceil \frac{B(FilteredParts)}{M} \right\rceil + \left\lceil \frac{B(Catalog)}{M} \right\rceil = \lceil \frac{3}{15} \rceil + \lceil \frac{147}{15} \rceil = 1 + 10 < 15$, so we can use the more efficient form,

thus the cost of the SM join is

$$\begin{aligned}
&Read\left(\sigma_{color='red}(Parts)\right) + Read(Catalog) \\
&+ 2\left(B\left(\sigma_{color='red}(Parts)\right) + B\left(Catalog\right)\right) \\
=&30 + 147 + 2\left(3 + 147\right) \\
=&477
\end{aligned}$$

4.3 Query plan 3

This query isn't possible, if we want to have $Parts$(in particular, filtered parts) as the outer relation and $Catalog$ as the inner one, we need an index on the primary key of the inner one, that is, an index on $Catalog.pid$, which does not exist. So the answer is $\mathbf{-1}$

4.4 Query plan 4

This query is possible(we have an index on the primary key of the inner table)

The cost of an INL join is

$$Read\left(\sigma_{price<20}(Catalog)\right) + T\left(\sigma_{price<20}(Catalog)\right) \cdot \text{selectCost}$$

- Note that $Read\left(\sigma_{price<20}(Catalog)\right) \overset{\text{full table scan}}{=} Read(Catalog) = 147$
- Also note that $T\left(\sigma_{price<20}(Catalog)\right) = \left\lceil 100,000 \cdot \frac{1}{3} \right\rceil = 33,334$ (we agreed in the lecture that if the distribution of some column is unknown, we use $\frac{1}{3}$, and we take a ceiling for some reason)
- As for the select cost, we will have an **index unique** scan:
  - we have a branching factor of 40 on $Parts.pid$, thus we have to traverse $\left\lceil \log_{\lceil 40/2 \rceil} 10,000 \right\rceil = 4$ levels
  - then we read a leaf
  - then we retrieve the row

  So $selectCost = 4 + 1 + 1 = 6$

Therefore, the answer is

$$147 + 33334 \cdot 6 = \mathbf{200,151}$$

4.5 Query plan 5

the cost of the BNL join is:

$$Read\left(\sigma_{color='red'}(Parts)\right) + Read\left(\sigma_{price<20}(Catalog)\right) \cdot \left\lceil \frac{B\left(\sigma_{color='red'}(Parts)\right)}{M} \right\rceil$$

- Regarding $\sigma_{price<20}(Catalog)$, we'll be performing an **index range** scan

- We have a branching factor of 20 on $Catalog.price$, so we have $\left\lceil \log_{20/2} 100,000 \right\rceil = 5$ levels
- By the assumption/like in previous questions, $T\left(\sigma_{price<20}(Catalog)\right) = \left\lceil 100,000 \cdot \frac{1}{3} \right\rceil = 33,334$,

  therefore we need to scan no more than $\left\lceil \frac{33,334}{20/2-1} \right\rceil = \left\lceil \frac{33,334}{9} \right\rceil = 3704$ leaves
- We also have to retrieve the rows themselves, that is, retrieve $33,334$ rows, however this is bigger than the cost of reading all blocks, therefore in this stage we do a

  full table scan, whose cost is $\overbrace{Read\left(\sigma_{price<20}(Catalog)\right)}^{\text{full table scan}} = B(Catalog) = 147$(as we calculated in a previous query plan)
- So, the answer is

$$\overbrace{Read\left(\sigma_{price<20}(Catalog)\right)}^{\text{index scan}} = 5 + 3704 + 147 = 3856$$

- Regarding $\sigma_{color='red'}(Parts)$, we'll be performing a **full table scan** - we already calculated this in query plan 2, as we've seen:

$$Read\left(\sigma_{color='red}(Parts)\right) = 30$$
$$B\left(\sigma_{color='red}(Parts)\right) = 3$$

To summarize, the cost of the join is $30 + 3856 \cdot \left\lceil \frac{3}{15-2} \right\rceil = \mathbf{3886}$

4.6 Query plan 6

The cost of the hash join is

$$Read\left(\sigma_{color='red'}(Parts)\right) + Read\left(\sigma_{price<20}(Catalog)\right)$$
$$+ 2\left(B\left(\sigma_{color='red'}(Parts)\right) + B\left(\sigma_{price<20}(Catalog)\right)\right)$$

But first we need to ensure it can work, that is:

$$\vee \begin{matrix} \left\lceil \frac{B\left(\sigma_{color='red'}(Parts)\right)}{M-1} \right\rceil < M-1 \\ \left\lceil \frac{B\left(\sigma_{price<20}(Catalog)\right)}{M-1} \right\rceil < M-1 \end{matrix}$$

As for the quantities:

- Regarding $\sigma_{color='red'}(Parts)$:

$$Read\left(\sigma_{color='red}(Parts)\right) \overset{\text{full table scan}}{=} Read(Parts) \overset{\text{seen before}}{=} 30$$

NOTE that we project $pname, pid$, so the number of blocks is smaller than seen in other queries. We have

$$T\left(\sigma_{color='red'}(Parts)\right) \overset{\text{given distribution}}{=} \left\lceil \frac{10,000}{10} \right\rceil = 1000$$

A block can contain $\left\lceil \frac{8192}{10+4} \right\rceil = 585$ entries of the form $\pi_{pname,pid}(something)$, so we have

$$B\left(\sigma_{color='red'}(Parts)\right) = \left\lceil \frac{1000}{585} \right\rceil = 2$$

- Regarding $\sigma_{price<20}(Catalog)$, we'll be performing an **index range scan** with row retrieval

  Therefore, the cost of this scan is identical to the one in query plan 5, $\overbrace{Read\left(\sigma_{price<20}(Catalog)\right)}^{\text{index scan}} \overset{\text{seen before}}{=} 3856$

  – One would be tempted to omit the step of retrieving rows, but we don't have an index over $Catalog.pid$, so we need to retrieve the entire rows(after matching by $Catalog.price$)

  As for the number of blocks, first, note that $T\left(\sigma_{price<20}(Catalog)\right) = \left\lceil \frac{100,000}{3} \right\rceil = 33,334$.

  Since we're projecting pid(4 bytes), note that a block can contain $\left\lceil \frac{8192}{4} \right\rceil = 2048$ entries(output of index range scan), so we have

$$B\left(\sigma_{price<20}(Catalog)\right) = \left\lceil \frac{33,334}{2048} \right\rceil = 17$$

Note that we **can** use hash join, e.g,

$$\frac{2}{14} < 15$$

So, plugging into the formula we get that its cost is

$$30 + 3856 + 2\left(2 + 17\right) = \mathbf{3924}$$

5 Optimal Plan

1. Which plan is the most optimal? Essentially we need to go over all possible query plans and take the one(s) with the lowest cost.

   Luckily, we've already calculated some relevant query plans in in the previous quiz. Out of them, BNL had the lowest cost, so I'm **guessing** the answer is **Block Nested Loops**

   (This does NOT necessarily rule out other forms, since we didn't see them in all configurations, e.g, query plans 6 and 7 could've used both full table scans rather than index scans for both tables. this is a GUESS)

2. One might be tempted to pick 471(the answer to BNL, that is, query plan 1 from the previous quiz), however, this isn't fully pushed(even though it's better than other, fully pushed queries)

   This is the query in the most pushed form:

   $$(\pi_{pid,pname}\sigma_{color='red'}(Parts)) \bowtie (\pi_{pid}\sigma_{price<20}(Catalog))$$

   We still have 8 options (whether to calculate these selections via full table scan or index scan, and which will serve as outer/inner term)

   Recall the cost of BNL is $Read(Outer) + Read(Inner)\left\lceil \frac{B(Inner)}{M} \right\rceil$

   Let's calculate some quantities(most were already calculated in the previous quiz)

   - Read costs (only dependant on selections)
     - Full table scan:

       $$Read\left(\pi_{pid,pname}\sigma_{color='red'}(Parts)\right) = Read(Parts) = 30$$
       $$Read\left(\pi_{pid}\sigma_{price<20}(Catalog)\right) = Read(Catalog) = 147$$

     - Index scan:

       $$Read\left(\pi_{pid,pname}\sigma_{color='red'}(Parts)\right) = ?$$
       $$Read\left(\pi_{pid}\sigma_{price<20}(Catalog)\right) = 3856$$

       One can see that the read cost of the projected+filtered catalog is much bigger than full table scan.
       As for 'parts', we have no index on the color, so I'm guessing this will also be bigger than a full table scan.

   - Block counts (only dependant on projections)
     - For $\pi_{pid,pname}\sigma_{color='red'}(Parts)$, every block can have $\left\lfloor \frac{8192}{10+4} \right\rfloor = 585$ tuples.
       By the given distribution, $T\left(\pi_{pid,pname}\sigma_{color='red'}(Parts)\right) = T\left(\sigma_{color='red'}(Parts)\right) = \left\lceil \frac{10{,}000}{10} \right\rceil = 1,000$
       So $B\left(\pi_{pid,pname}\sigma_{color='red'}(Parts)\right) = \left\lceil \frac{1000}{585} \right\rceil = 2$
     - For $\pi_{pid}\sigma_{price<20}(Catalog)$, every block can have $\left\lfloor \frac{8192}{4} \right\rfloor = 2048$ tuples
       By convention, $T\left(\pi_{pid}\sigma_{price<20}(Catalog)\right) = T\left(\sigma_{price<20}(Catalog)\right) = \left\lceil \frac{100{,}000}{3} \right\rceil = 33,334$
       So $B\left(\pi_{pid}\sigma_{price<20}(Catalog)\right) = \left\lceil \frac{33{,}334}{2048} \right\rceil = 17$

   To conclude, we will use full table scans on both inputs, so the question is which ordering(2 options):

   (a) Parts outer, catalog inner: $30 + 147\left\lceil \frac{17}{15} \right\rceil = 324$

(b) Catalog outer, parts inner: $147 + 30\left\lceil \frac{2}{15} \right\rceil = 177$

So, the answer is **177**

3. This answer isn't related to the query plan, but rather, estimating query sizes.

   - First, regarding the size of the join(without filters), we have(as per slide 13 in query estimation) the following amount of rows

     $$T(Catalog \bowtie Parts) = T(Catalog) = 100,000$$

     (one part may be in many catalogs)

     Now, composing the filter $\sigma_{color='red'}$, we have $V(color, Parts) = 10$ by the given distribution, so we have

     $$T\left(\sigma_{color='red'}(Catalog \bowtie Parts)\right) = \frac{100,000}{10} = 10,000$$

     Now, composing $\sigma_{price<20}$, we don't have a known distribution on price, so by our convention, we have $T\left(\sigma_{price<20}\sigma_{color='red'}(Catalog \bowtie Parts)\right) = \left\lceil \frac{10,000}{3} \right\rceil = 3,334$

   So the answer is $\mathbf{3,334}$

4. Note that an output entry has $pname$, taking 10 bytes, so a block can contain $\left\lfloor \frac{8192}{10} \right\rfloor = 819$ entries, therefore we'd need $\left\lceil \frac{3,334}{819} \right\rceil = \mathbf{5}$ blocks.