>>

Query Cost Estimation

∧ >>

❖ Query Cost Estimation

Without executing a plan, cannot always know its precise cost.

Thus, query optimisers estimate costs via:

- cost of performing operation (dealt with in earlier lectures)
- size of result (which affects cost of performing next operation)

Result size estimated by statistical measures on relations, e.g.

r_S	cardinality of relation S
R_S	avg size of tuple in relation S
$V(A,S)$	# distinct values of attribute A
$min(A,S)$	min value of attribute A
$max(A,S)$	max value of attribute A

<< ^ >>

❖ Estimating Projection Result Size

Straightforward, since we know:

- number of tuples in output

  $r_{out} = |\,\pi_{a,b,..}(T)\,| = |\,T\,| = r_T$ (in SQL, because of bag semantics)

- size of tuples in output

  R_{out} = sizeof(a) + sizeof(b) + ... + tuple-overhead

Assume page size B, $b_{out} = ceil(r_T / c_{out})$, where $c_{out} = floor(B/R_{out})$

If using **`select distinct`** ...

- $|\,\pi_{a,b,..}(T)\,|$ depends on proportion of duplicates produced

<< ^ >>

❖ Estimating Selection Result Size

Selectivity = fraction of tuples expected to satisfy a condition.

Common assumption: attribute values uniformly distributed.

Example: Consider the query

```
select * from Parts where colour='Red'
```

If $V(colour,Parts)=4, \; r=1000 \Rightarrow |\sigma_{colour=red}(Parts)|=250$

In general, $|\sigma_{A=c}(R)| \cong r_R / V(A,R)$

Heuristic used by PostgreSQL: $|\sigma_{A=c}(R)| \cong r/10$

<< ^ >>

❖ Estimating Selection Result Size (cont)

Estimating size of result for e.g.

```
select * from Enrolment where year > 2015;
```

Could estimate by using:

- uniform distribution assumption, *r*, min/max years

Assume: min(year)=2010, max(year)=2019, $|Enrolment| = 10^5$

- 10^5 from 2010-2019 means approx 10000 enrolments/year
- this suggests 40000 enrolments since 2016

Heuristic used by some systems: $|\sigma_{A>c}(R)| \cong r/3$

<< ^ >>

❖ Estimating Selection Result Size (cont)

Estimating size of result for e.g.

```
select * from Enrolment where course <> 'COMP9315';
```

Could estimate by using:

- uniform distribution assumption, *r*, domain size

e.g. $|V(course,Enrolment)| = 2000$, $|\sigma_{A<>c}(E)| = r * 1999/2000$

Heuristic used by some systems: $|\sigma_{A<>c}(R)| \cong r$

<< ^ >>

❖ Estimating Selection Result Size (cont)

How to handle non-uniform attribute value distributions?

- collect statistics about the values stored in the attribute/relation
- store these as e.g. a histogram in the meta-data for the relation

So, for part colour example, might have distribution like:

`White`: 35% **`Red`**: 30% **`Blue`**: 25% **`Silver`**: 10%

Use histogram as basis for determining # selected tuples.

Disadvantage: cost of storing/maintaining histograms.

<< ^ >>

❖ Estimating Selection Result Size (cont)

Summary: analysis relies on operation and data distribution:

E.g. `select * from R where a = k;`

Case 1: *uniq(R.a)* ⇒ 0 or 1 result

Case 2: r_R tuples && *size(dom(R.a))* = *n* ⇒ r_R / n results

E.g. `select * from R where a < k;`

Case 1: $k \le min(R.a)$ ⇒ 0 results

Case 2: $k > max(R.a)$ ⇒ $\cong r_R$ results

Case 3: *size(dom(R.a))* = *n* ⇒ ? *min(R.a) ... k ... max(R.a)* ?

<< ^ >>

❖ Estimating Join Result Size

Analysis relies on semantic knowledge about data/relations.

Consider equijoin on common attr: $R \bowtie_a S$

Case 1: $values(R.a) \cap values(S.a) = \{\} \Rightarrow size(R \bowtie_a S) = 0$

Case 2: $uniq(R.a)$ and $uniq(S.a) \Rightarrow size(R \bowtie_a S) \le min(|R|, |S|)$

Case 3: $pkey(R.a)$ and $fkey(S.a) \Rightarrow size(R \bowtie_a S) \le |S|$

❖ Cost Estimation: Postscript

Inaccurate cost estimation can lead to poor evaluation plans.

Above methods can (sometimes) give inaccurate estimates.

To get more accurate cost estimates:

- more time ... complex computation of selectivity
- more space ... storage for histograms of data values

Either way, optimisation process costs more (more than query?)

Trade-off between optimiser performance and query performance.

Produced: 5 Apr 2021