1 Cost Models

Assumptions

1. **Uniformity and independence assumption**
   All values of an attribute uniformly appear with the same probability (or even distribution). Values of different attributes are independent of each other.
   Simple, yet rarely realistic assumption
2. **Worst case assumption**
   No knowledge about relation contents available at all. In case of a selection σ_p, assume that all records will satisfy predicate p.
   Unrealistic assumption, can only be used for computing upper bounds
3. **Perfect knowledge assumption**
   Details about the exact distribution of values are known. Requires huge catalog or prior knowledge of incoming queries.
   Unrealistic assumption, can only be used for computing lower bounds

1.1 File Otganization

Cost Model :

Parameter	Description
b	number of pages per file
r	number of records per page
D	time to read a disk page
C	CPU time needed to process a record
H	CPU time taken to apply a function to a record e.g. comparison or hash

Cost of Scan :

File Org	Description	est. Cost
Heap	read all pages, process each of the records per page	$b \cdot (D + r \cdot C)$
Sorted	Same as for heap files	$b \cdot (D + r \cdot C)$
Hashed	additional free space due to overflow chain avoidance	$(100/80) \cdot b \cdot (D + r \cdot C)$

Cost of Search w. Equality :

File Org	Description	est. Cost
Heap	if equality test is on primary key, adds factor $\frac{1}{2}$	$b \cdot (D + r \cdot C)$ or $\frac{1}{2} b \cdot (D + r \cdot C)$
Sorted	Assuming equality test is on sort criterion, use bin search	$\log_2 b \cdot Dlog_2 r \cdot C$
Hashed	Assuming equality test on hash attribute. Directly leads to the page containing the hit	$H + D + r \cdot C$ or $H + D + \frac{1}{2} r \cdot C$

Cost of Search w. Range :

File Org	Description	est. Cost
Heap	Can appear everywhere =¿ full scan	$b \cdot (D + r \cdot C)$
Sorted	Search for equality=lower and scan sequentially until the first record with A ¿ upper	$\log_2 b \cdot D + \log_2 r \cdot C + \lfloor \frac{n}{r} \rfloor \cdot D + n \cdot C$
Hashed	Performs worst as additional space needs to be scaned	$(100/80) \cdot b \cdot (D + r \cdot C)$

Cost of Insert :

File Org	Description	est. Cost
Heap	Can be written to an arbitary page, involves reading and writing the page	$2D + C)$
Sorted	Insert into a sepcific place and shift all subsequent	$\log_2 b \cdot D + \log_2 r \cdot C + \frac{1}{2} \cdot b \cdot (2 \cdot D + r \cdot C)$
Hashed	Write to the page that the hash fn indicates	$H + D + C + D$

Cost of Delete :

File Org	Description	est. Cost
Heap	Read, delete and write	$2D + C)$
Sorted	delete and shift all subsequent	$D + \frac{1}{2} \cdot b \cdot (2 \cdot D + r \cdot C)$
Hashed	Access by rid is faster than hashing so same as heap	$D + C + D$

1.2 System Catalog

size of buffer pool, page size, information and statistics about tables, views, indexes

Information stored in the system catalog

- **Table metadata**
  - *table name*, *file name* (or some identifier), *file structure* (e.g., heap file)
  - *attribute name* and *type* of each attribute of the table
  - *index name* of each index on the table
  - *integrity constrains* (e.g., primary and foreign key constraints) on the table
- **Index metadata**
  - *index name* and *structure* (e.g., B+ tree)
  - *search key* attributes
- **View metadata**
  - *view name* and *definition*

Statistics

- **Table statistics**
  - *cardinality*: number of tuples $NTuples(R)$ for each table R
  - *size*: number of pages $NPages(R)$ for each table R
- **Index statistics**
  - *cardinality*: number of distinct key values $NKeys(I)$ for each index I
  - *size*: number of pages $INPages(I)$ for each index (for a tree index I, $INPages(I)$ denotes the number of leaf pages)
  - *height*: number of non-leaf levels for each tree index I
  - *range*: minimum present key value $ILow(I)$ and the maximum present key value $IHigh(I)$ for each index I

Example

Tables

name	file	#tuples	size
Tables	...	6	1
Attributes	...	23	1
Views	...	1	1
Indexes	...	0	1
Sailors	...	40,000	500
Reserves	...	100,000	1000

Views

name	text
Captains	SELECT * FROM Sailors WHERE...

Indexes

name	file	type	#keys	size
Boats	...	B+Tree	100	1

Attributes

name	table	type	pos
name	Tables	string	1
file	Tables	string	2
#tuples	Tables	integer	3
size	Tables	integer	4
name	Attributes	string	1
table	Attributes	string	2
type	Attributes	string	3
pos	Attributes	integer	4
⋮	⋮	⋮	⋮
sid	Sailors	integer	1
sname	Sailors	string	2
rating	Sailors	integer	3
age	Sailors	real	4
sid	Reserves	integer	1
bid	Reserves	integer	2
day	Reserves	date	3
rname	Reserves	string	4

Typical database profile for relation *R*

$NTuples(\mathbf{R})$	number of tuples in relation **R**
$NPages(\mathbf{R})$	number of disk pages allocated for relation **R**
$s(\mathbf{R})$	average record size (width) of relation **R**
b	block size, alternative to $s(\mathbf{R})$
	$NPages(N) = NTuples(R) / \lfloor {}^{b}/_{s(R)} \rfloor$
$V(\mathbf{A}, \mathbf{R})$	number of distinct values of attribute **A** in relation **R**
$High(\mathbf{A}, \mathbf{R})/Low(\mathbf{A}, \mathbf{R})$	maximum and minimum value of attribute **A** in relation **R**
$MCV(\mathbf{A}, \mathbf{R})$	most common value(s) of attribute **A** in relation **R**
$MVF(\mathbf{A}, \mathbf{R})$	frequency of most common value(s) of attribute **A** in relation **R**
⋮	*possibly many more*

1.3 Access Paths

Cost model for access methods on relation *R*

Access method	Cost
access primary index **I**	$\begin{cases} height(\mathbf{I}) + 1 & \text{if } \mathbf{I} \text{ is B+ tree} \\ 1.2 + 1 & \text{if } \mathbf{I} \text{ is hash index} \end{cases}$
clustered index **I** matching predicate p	$(NPages(\mathbf{I}) + NPages(\mathbf{R})) \cdot sel(p)$
unclustered index **I** matching predicate p	$(NPages(\mathbf{I}) + NTuples(\mathbf{R})) \cdot sel(p)$
sequential scan	$NPages(\mathbf{R})$

If less than 5% are retrieved, a table scan is cheaper.

Hash vs. B+Tree Index: Hash Indexes match if selection contains equality on indexed attribute; B+Trees match if selection contains any condition on an attribute in the trees search prefix. If matches with an index were found in a CNF, those conjuncts are called **primary conjuncts**

1.4 Operators

In Total Two-way Merge Sort costs

$$2N(1 + log_2 N)\text{I/O ops}$$

In Total External Merge Sort costs

$$2N(1 + \lceil \log_{B-1} \lceil \frac{N}{B} \rceil \rceil)\text{I/O ops}$$

Selection query $Q := \sigma_{A=c}(R)$

Selectivity $sel(\mathbf{A} = c)$	$\begin{cases} MCF(\mathbf{A}, \mathbf{R})[c] & \text{if } c \in MCF(\mathbf{A}, \mathbf{R}) \\ 1/V(\mathbf{A}, \mathbf{R}) & \textbf{(uniformity assumption)} \end{cases}$	
Cardinality $\lvert Q \rvert$	$sel(\mathbf{A} = c) \cdot NTuples(\mathbf{R})$	
Record size $s(Q)$	$s(\mathbf{R})$	
Number of attribute values $V(\mathbf{A}', Q)$	$\begin{cases} 1, & \text{for } \mathbf{A}' = \mathbf{A} \\ c(\lvert\mathbf{R}\rvert, V(\mathbf{A}, \mathbf{R}), \lvert Q \rvert) & \text{otherwise} \end{cases}$	

Definition

The **selectivity** (or **reduction factor**) or a predicate p, denoted by $sel(p)$, is the faction of records in a relation R that satisfy the predicate p.

$$0 \leq sel(p) = \frac{\lvert \sigma_p(R) \rvert}{\lvert R \rvert} \leq 1$$

Cost of $\sigma_p^{scan}(R_{in})$ using a sequential scan

access path	file scan (**openScan**) of R_{in}
prerequisites	none (p arbitrary, R_{in} may be a heap file)
I/O cost	$\underbrace{\lVert R_{in}\rVert}_{\text{input cost}} + \underbrace{sel(p)\cdot\lVert R_{in}\rVert}_{\text{output cost}}$

Cost of $\sigma_p(R_{in})$ using binary search

access path	binary search, then sorted file scan of R_{in}
prerequisites	R_{in} sorted on sort key k that matches p
I/O cost	$\underbrace{\log_2\lVert R_{in}\rVert}_{\text{input cost}} + \underbrace{sel(p)\cdot\lVert R_{in}\rVert}_{\text{sorted scan}} + \underbrace{sel(p)\cdot\lVert R_{in}\rVert}_{\text{output cost}}$

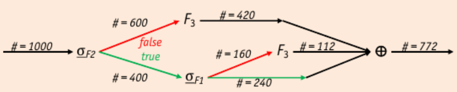
Cost of $\sigma_p(R_{in})$ using a clustered B+ tree index

access path	access of B+ tree on R_{in}, then sequence set scan
prerequisites	clustered B+ tree on R_{in} with key k that matches p
I/O cost	$\approx \underbrace{3}_{\text{B+ tree access}} + \underbrace{sel(p)\cdot\lVert R_{in}\rVert}_{\text{sorted scan}} + \underbrace{sel(p)\cdot\lVert R_{in}\rVert}_{\text{output cost}}$

Cost of $\sigma_p(R_{in})$ using a hash index

access path	hash index on R_{in}
prerequisites	R_{in} hashed on key k, p has a term $k = c$
I/O cost	$\approx \underbrace{1.2}_{\text{B+ tree access}} + \underbrace{sel(p)\cdot\lVert R_{in}\rVert}_{\text{output cost}}$

Example



Mean cost per tuple (⊕ disjoint union): $C_2 + (1 - s_2)\cdot C_3 + s_2\cdot(C_1 + (1 - s_1)\cdot C_3) = 40.6$

Note that many variations are possible, e.g., for tuning in parallel environments

$$c(n,m,r) = \begin{cases} r, & \text{for } r < \frac{m}{2} \\ \frac{r+m}{3}, & \text{for } \frac{m}{2} \le r < 2m \\ m, & \text{for } r \ge 2m \end{cases}$$

Selection query $Q := \sigma_{A=B}(R)$

Equality between attributes, e.g., $\sigma_{A=B}(\mathbf{R})$ can be approximated by

$$sel(\mathbf{A} = \mathbf{B}) = 1/\max(V(\mathbf{A}, \mathbf{R}), V(\mathbf{B}, \mathbf{R}))$$

This formula assumes that each value of the attribute with fewer distinct values has a corresponding match in the other attribute (**independence assumption**).

Selection query $Q := \sigma_{A>c}(R)$

If $Low(\mathbf{A}, \mathbf{R}) \le c \le High(\mathbf{A}, \mathbf{R})$, range selections, e.g., $\sigma_{A>c}(\mathbf{R})$ can be approximated by

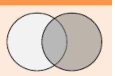
$$sel(\mathbf{A} > c) = \frac{High(\mathbf{A}, \mathbf{R}) - c}{High(\mathbf{A}, \mathbf{R}) - Low(\mathbf{A}, \mathbf{R})}$$

This formula uses the **uniformity assumption**.

Selection query $Q := \sigma_{A\ \mathtt{IN}\ (\ldots)}(R)$

Element tests, e.g., $\sigma_{\mathbf{A}\ \mathtt{IN}\ (\ldots)}(\mathbf{R})$ can be approximated by multiplying the selectivity for an equality selection $sel(\mathbf{A} = c)$ with the number of elements in the list of values.

Selections with composite predicates



- **conjunctive** predicates, e.g., $Q := \sigma_{\mathbf{A}=c_1 \land \mathbf{B}=c_2}(\mathbf{R})$

$$sel(\mathbf{A} = c_1 \land \mathbf{B} = c_2) = sel(\mathbf{A} = c_1)\cdot sel(\mathbf{B} = c_2)$$

which gives $|Q| = \dfrac{|\mathbf{R}|}{V(\mathbf{A}, \mathbf{R})\cdot V(\mathbf{B}, \mathbf{R})}$

- **disjunctive** predicates, e.g., $Q := \sigma_{\mathbf{A}=c_1 \lor \mathbf{B}=c_2}(\mathbf{R})$

$$sel(\mathbf{A} = c_1 \lor \mathbf{B} = c_2) = sel(\mathbf{A} = c_1) + sel(\mathbf{B} = c_2) - sel(\mathbf{A} = c_1)\cdot sel(\mathbf{B} = c_2)$$

which gives $|Q| = \dfrac{|\mathbf{R}|}{V(\mathbf{A}, \mathbf{R}) + V(\mathbf{B}, \mathbf{R}) - V(\mathbf{A}, \mathbf{R})\cdot V(\mathbf{B}, \mathbf{R})}$

Projection query $Q := \pi_L(R)$

Cardinality $|Q|$

$$\begin{cases} V(\mathbf{A}, \mathbf{R}), & \text{for } L = \{\mathbf{A}\} \\ |\mathbf{R}| & \text{if keys of } \mathbf{R} \in L \\ |\mathbf{R}| & \text{no duplicate elimination} \\ \min(|\mathbf{R}|, \Pi_{\mathbf{A}_i \in L} V(\mathbf{A}_i, \mathbf{R})) & \text{otherwise} \end{cases}$$

Record size $s(Q)$ $\quad \Sigma_{\mathbf{A}_i \in L}\, s(\mathbf{A}_i)$

Number of attribute values $V(\mathbf{A}_i, Q)$ $\quad V(\mathbf{A}_i, \mathbf{R})$ $\quad$ for $\mathbf{A}_i \in L$

Cost of $\pi_l^{sort}(R_{in})$ using sorting for duplicate elimination

access path	file scan (**openScan**) of R_{in}
prerequisites	none (B available buffer pages)
I/O cost	$\underbrace{\lVert R_{in}\rVert + \lVert R_{tmp}\rVert}_{\text{projection}} + \underbrace{2\cdot\lVert R_{tmp}\rVert\cdot\left(\left\lceil\log_{B-1}\left\lceil{}^{\lVert R_{tmp}\rVert}/_{B}\right\rceil\right\rceil\right)}_{\text{duplicate elimination}}$

Cost of $\pi_l^{hash}(R_{in})$ using hashing for duplicate elimination

access path	file scan (**openScan**) of R_{in}
prerequisites	none (B available buffer pages)
I/O cost	$\underbrace{\lVert R_{in}\rVert + \lVert R_{tmp}\rVert}_{\text{projection}} + \underbrace{\lVert R_{tmp}\rVert + \lVert R_{tmp}\rVert}_{\text{duplicate elimination}}$

Union query $Q := R \cup S$

$$\begin{aligned} |Q| &\le |\mathbf{R}| + |\mathbf{S}| \\ s(Q) &= s(\mathbf{R}) = s(\mathbf{S}) \\ V(\mathbf{A}, Q) &\le V(\mathbf{A}, \mathbf{R}) + V(\mathbf{A}, \mathbf{S}) \end{aligned}$$

$sch(\mathbf{R}) = sch(\mathbf{S})$

Difference query $Q := R - S$

$$\begin{aligned} \max(0, |\mathbf{R}| - |\mathbf{S}|) \le |Q| &\le |\mathbf{R}| \\ s(Q) &= s(\mathbf{R}) = s(\mathbf{S}) \\ V(\mathbf{A}, Q) &\le V(\mathbf{A}, \mathbf{R}) \end{aligned}$$

Cross-product query $Q := R \times S$

$$\begin{aligned} |Q| &= |\mathbf{R}|\cdot|\mathbf{S}| \\ s(Q) &= s(\mathbf{R}) + s(\mathbf{S}) \\ V(\mathbf{A}, Q) &= \begin{cases} V(\mathbf{A}, \mathbf{R}), & \text{if } \mathbf{A} \in sch(\mathbf{R}) \\ V(\mathbf{A}, \mathbf{S}), & \text{if } \mathbf{A} \in sch(\mathbf{S}) \end{cases} \end{aligned}$$

Special cases of join queries $Q := R \bowtie_p S$

- no common attributes ($sch(\mathbf{R}) \cap sch(\mathbf{S}) = \emptyset$) or join predicate p = true

$$\mathbf{R} \bowtie_p \mathbf{S} = \mathbf{R} \times \mathbf{S}$$

- join attribute, say $\mathbf{A}$, is key in one of the relations, e.g., in $\mathbf{R}$, and assuming the inclusion dependency $\pi_{\mathbf{A}}(\mathbf{S}) \subseteq \pi_{\mathbf{A}}(\mathbf{R})$

$$|Q| = |\mathbf{R}|$$

↬ this inclusion dependency is guaranteed by a foreign key relationship between $\mathbf{R.A}$ and $\mathbf{S.A}$ in $\mathbf{R} \bowtie_{\mathbf{R.A}=\mathbf{S.A}} \mathbf{S}$.

General join queries $Q := R \bowtie_{R.A=S.B} S$

Assuming inclusion dependencies, the cardinality of a general join query Q can be estimated as

Cardinality $|Q|$

$$\begin{cases} \dfrac{|\mathbf{R}|\cdot|\mathbf{S}|}{V(\mathbf{A}, \mathbf{R})}, & \text{for } \pi_{\mathbf{B}}(\mathbf{S}) \subseteq \pi_{\mathbf{A}}(\mathbf{R}) \\ \dfrac{|\mathbf{R}|\cdot|\mathbf{S}|}{V(\mathbf{B}, \mathbf{S})}, & \text{for } \pi_{\mathbf{A}}(\mathbf{R}) \subseteq \pi_{\mathbf{B}}(\mathbf{S}) \end{cases}$$

Typically, the smaller of these two estimates is used

Cardinality $|Q|$ $\quad \dfrac{|\mathbf{R}|\cdot|\mathbf{S}|}{\max(V(\mathbf{A}, \mathbf{R}), V(\mathbf{B}, \mathbf{S}))}$

Record size $s(Q)$ $\quad s(\mathbf{R}) + s(\mathbf{S}) - \Sigma\, s(\mathbf{A}_i)$ $\quad$ for all common $\mathbf{A}_i$ of a **natural join**

Number of attribute values $V(\mathbf{A}', Q) \le \begin{cases} \min(V(\mathbf{A}', \mathbf{R}), V(\mathbf{A}', \mathbf{S})), & \mathbf{A}' \in sch(\mathbf{R}) \cap sch(\mathbf{S}) \\ V(\mathbf{A}', X), & \mathbf{A}' \in sch(X) \end{cases}$

Cost of $R_1 \bowtie_p^{nl} R_2$

access path	file scan (**openScan**) of R_1 and R_2
prerequisites	none (p arbitrary, R_1 and R_2 may be heap files)
I/O cost	$\underbrace{\lVert R_1\rVert}_{\text{outer loop}} + \underbrace{\lvert R_1\rvert\cdot\lVert R_2\rVert}_{\text{inner loop}}$

Block Nested Loops Join Costs: $\lceil ||R_1||/b_1\rceil \cdot \lceil ||R2||/b_2\rceil$

Cost of $R_1 \bowtie_p^{index\text{-}nl} R_2$

access path	file scan (**openScan**) of R_1, index access to R_2
prerequisites	index on R_2 that matches join predicate p
I/O cost	$\underbrace{\lVert R_1\rVert}_{\text{outer loop}} + \underbrace{\lvert R_1\rvert\cdot(\text{cost of } \textbf{one} \text{ index access to } R_2)}_{\text{inner loop}}$

Cost of $R_1 \bowtie_{A=B}^{sort\text{-}merge} R_2$

access path	sorted file scan of R_1 and R_2
prerequisites	p equality predicate $R_1.A = R_2.B$
I/O cost	cost of sorting R_1 and/or R_2, if not sorted already, plus
	best case: $\lVert R_1\rVert + \lVert R_2\rVert$
	worst case: $\lVert R_1\rVert\cdot\lVert R_2\rVert$

Cost of $R_1 \bowtie_{A=B}^{hash\text{-}join} R_2$

access path	file scan (**openScan**) of R_1 and R_2
prerequisites	equi-join, i.e., p equality predicate $R_1.A = R_2.B$
I/O cost	$\underbrace{\underbrace{\lVert R_1\rVert + \lVert R_2\rVert}_{\text{read}} + \underbrace{\lVert R_1\rVert + \lVert R_2\rVert}_{\text{write}}}_{\text{partitioning phase}} + \underbrace{\lVert R_1\rVert + \lVert R_2\rVert}_{\text{probing phase}} = 3\cdot(\lVert R_1\rVert + \lVert R_2\rVert)$