Chapter 3.6 Cost Functions

1. Restrictions (see also chapt. 3.4)

λ: Bytes/sec when reading from HD

ν: Bytes/sec when writing to HD

1.1 Relationscan: θ(n)

Estimate or determine constant experimentally:

n Tuple * b Bytes/Tuple * 1/λ sec/Byte

1.2 Access via Index: measure depending on B-tree height and cache-size

- random : Tuple/sec
- sequential

2. Projections:

2.1 θ**(n)**, read once all of R, like 1.1.

2.2 θ**(n log n)** : cost of sorting, determine experimentally, see also course on Data Structures

2.3 $\theta(n^2)$: for nested loop, see also nested-loop-join

3.Join – cost (consider only I/0)

3.1 Nested Loop: $\underbrace{R1 \bowtie R2}_{T1} \bowtie \underbrace{R3}_{T2}$

Variant 1: R1, T1 fit in AS

R1;	R2;	T1;	R3;	T2;
read	read	write	read	write
		read		

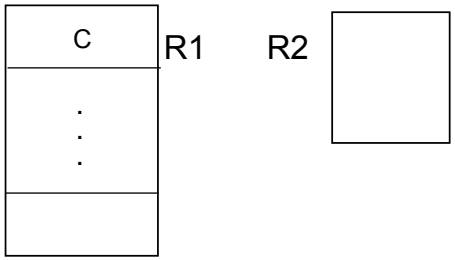
|R1|: number of tuples, ||R1|| size in bytes

$$\frac{1}{\lambda}||R1|| + \frac{1}{\lambda}||R2|| + \left(\frac{1}{\lambda} + \frac{1}{\nu}\right)||T1|| + \frac{1}{\lambda}||R3|| + \frac{1}{\nu}||T2||$$

$\Rightarrow$ small intermediate result is good!

Variant 2: R1, T1 do not fit in AS

let c = cache – size in bytes



$$\frac{1}{\lambda}\|R1\|+\frac{\|R1\|}{c}\cdot\frac{1}{\lambda}\cdot\|R2\|+k_1\cdot sel(R1\bowtie R2)\cdot|R1|\cdot|R2|$$

$k_1=\frac{1}{\nu}\cdot$ tuple size of T1

cost of the 2. join: T1 ⋈ R3

$$\frac{1}{\lambda}\|T1\|+\frac{\|T1\|}{C}\cdot\frac{1}{\lambda}\cdot\|R3\|+k_2\cdot sel(T1\bowtie R3)\cdot|T1|\cdot|R3|$$

$k_2=\frac{1}{\nu}\cdot$ tuple size of T2

$\Rightarrow 0$ (|R1|*|R2|*|R3|)
constants are influenced by C and Join-Sequence

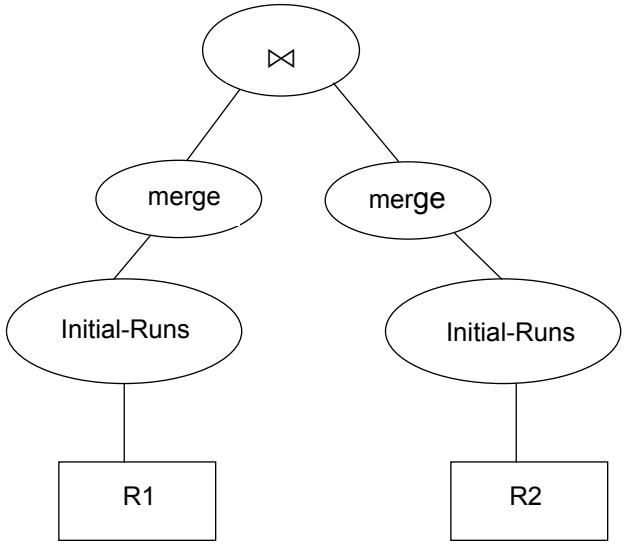
3.2 Merge-Join:

$$\frac{1}{\lambda}\|R1\|+\frac{1}{\lambda}\|R2\|+\frac{1}{\nu}\|T1\|$$

with ||T1|| = sel (R1 ⋈ R2) * |R1| * |R2| * tuple size
etc. for T2

3.3 Sort-Merge-Join:

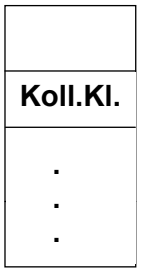
trivial idea: sort first, then Merge Join
better:



Comparison of cost?

3.4 Hash-Join:

R1 ⋈ R2



$$\frac{1}{\lambda}\|R1\|+\frac{1}{\lambda}\|R2\|+\frac{1}{\nu}\|T1\|$$

in AS

3.5 Double-Hash-Join:

k_1 should be chosen such that hash-classes of size ≅ c arise

1. R1 hashing:

$$\left(\frac{1}{\lambda}+\frac{1}{\nu}\right)\cdot\|R1\|$$

2. R2 hashing:

$$\left(\frac{1}{\lambda}+\frac{1}{\nu}\right)\cdot\|R2\|$$

3. read every collision class of R1 and corresponding collision class of R2

$$\frac{1}{\lambda}(\|R1\|+\|R2\|)$$

4. Write T1:

$$\frac{1}{\nu}\|T1\|$$

$$\Rightarrow\left(\frac{2}{\lambda}+\frac{1}{\nu}\right)(\|R1\|+\|R2\|)+\frac{1}{\nu}\|T1\|$$

3.6 Index Join:

R1 ⋈ R2 $\quad\theta$**(|R1| * log |R2|)**

read R1, for every tuple of R1 $j_{PR1,\ R2}$ index accesses to R2

$$\frac{1}{\lambda}\cdot\|R1\|+|R1|\cdot jp_{R1,\ R2}\cdot\frac{1}{\beta}+\frac{1}{\lambda}\|T1\|$$

where β = number of tuples per second, which can be delivered via random Index-accesses β ≈ 50 – 100/sec

3.7 Restrictions and Joins:

additive computation of costs, i.e. split up operations

⇒ **Consequence:** cost model and cost estimation are very complex and uncertain and imprecise

Secret of DBS providers!!