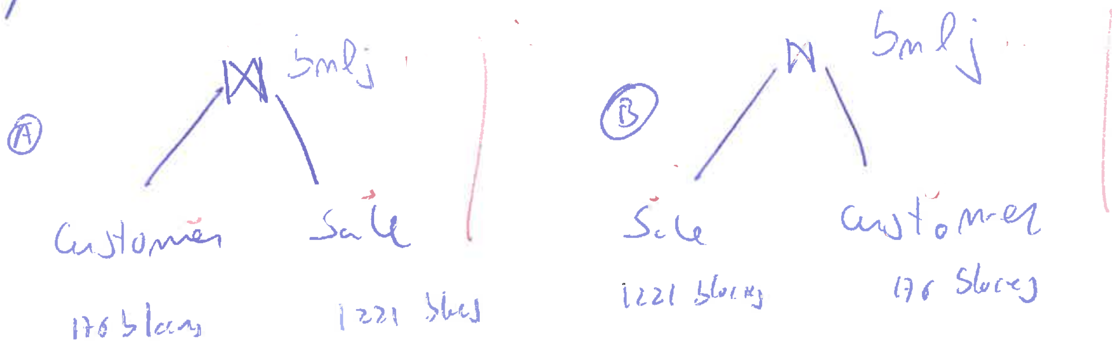
Soluções Teste 1 – SBD 2025/26

1a)



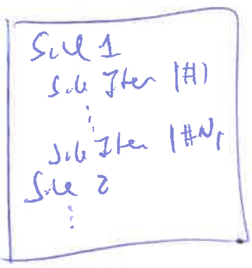
The cost for bnlj is $b_r * b_s + b_r$ block transfers + $2 * b_r$ seeks.

Thus in this case plan A is better since it will reduce the ~~number of seeks for~~ b_r blocks [illegible] fetch more tuples.

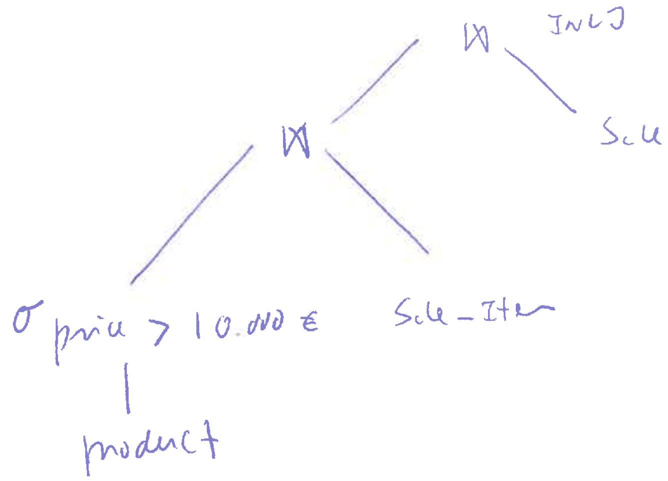
1b)

```
SELECT *
FROM Sale
     [illegible] SaleItem USING(Sale_id)
     INNER JOIN Product USING(Product_id)
WHERE Sale_id = [illegible];
```

I would cluster sales with SaleItem using Sale_id because this would make faster the retrieval of sale items for sales.



1c)



No, the indexes are not very useful except the one for obtaining the sale from the sale Item [illegible] (let sale Item)

Two indexes can be created:

- one for Sale_Item (product_id)
- an index for price could improve [illegible] but would be insignificant

1d) Supplier has 50 records

Each bitmap has 5000 bits ≃ 625 bytes

The index would occupy 4 blocks

The cost is simply to retrieve these 4 blocks at once and count the corresponding ones

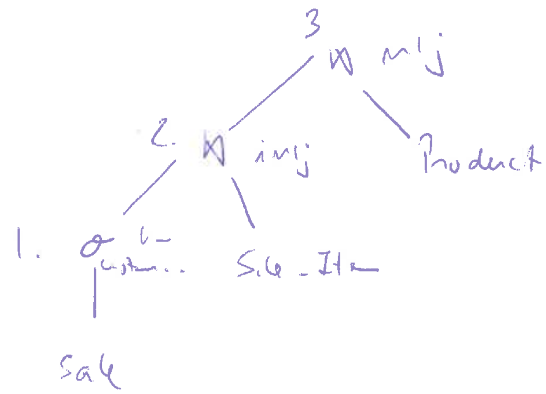
$4 t_T + 1 t_S = 13$ ms

if a seek would be needed = 40 ms | in any case it is [illegible]

Since the [illegible] is

$43 * t_T + 1 * t_S = 52$ ms

1e)



1. σ linear Sale will take 1 seek and $b_{Sale} * t_T$ transfer

$1 * t_S + \frac{b_{Sale}}{2} * t_T = 9\,ms + 1221 \times 1\,ms = 1230\,ms$

this returns 25 tuples

2. For each tuple returned in the previous query we do a index search, thus taking:

$n * (h_i + 1) * (t_S + t_T)$ since the tuples are pipelined and not need to be read from disk, obtaining:

$25 * (3 + 1) * (1 + 9\,ms) = 25 * 80\,ms = 2000\,ms$

3. Since products fits into memory we just need to read from disk Product once for each of the 100 sale items returned:

$1 * t_S + b_{Product} * t_T = 9\,ms + 43 * 1\,ms = 52\,ms$

2a) Surrogate keys are ~~per~~ identifiers generated by the DW and independent of the original keys of the information being inserted. The advantages are several

- Surrogate keys guarantee stable and permanent identifiers
- Enable SCD

⇒ they are smaller
- Need fewer columns
- Ensure integration across several multiple data sources

⋮

2b)

The use of random order insertion leads to problems due to

- Random I/O

~~TW... splits~~

- Poor

alternatives

1) - Sort data in order better I/O but will leave most blocks half full

2) Use bulk uploading tools

- Sort data
- Construct B+ tree level by level
- Better occupancy and I/O
- ~~...~~ - reducing size of B+ tree

2c)

The formula is

$b_r * t_T + b_r * t_s + n_r * C$

# of blocks to read from r

we need to do a seek for each block of the outer relation since we have just 1 block of memory

cost of the seen

number of tuples to search for in the inner

by reading b_b blocks at a time we have to do less seeks i.e.

$$\frac{b_r}{b_b}$$

since we read b_b blocks of r at a time