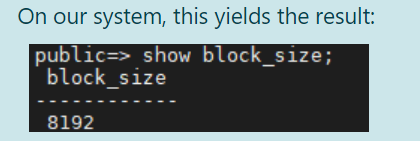
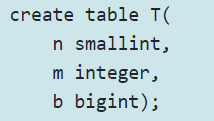
**בחני DB**

**שבוע 5:**

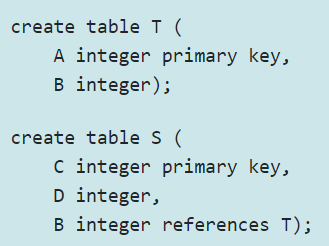
* **בוחן 1-**

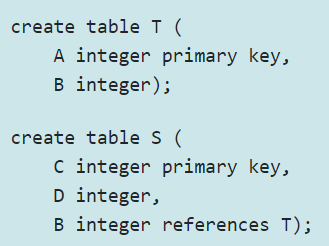


**2**

**4**

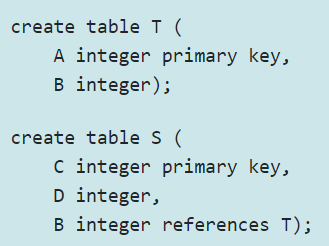
**8**

* Assuming that rows do not span blocks (and instead, are contained in a single block), how many rows of T can fit in a single block?
* How many bytes will be wasted in each block?
* Assuming that T contains 1200 rows, what is the I/O complexity of computing the query select \* from T ( I/O complexity is the number of blocks that must be read or written)
* **בוחן 2-**



Suppose we have 10,000 rows in table T and 100,000 rows in table S

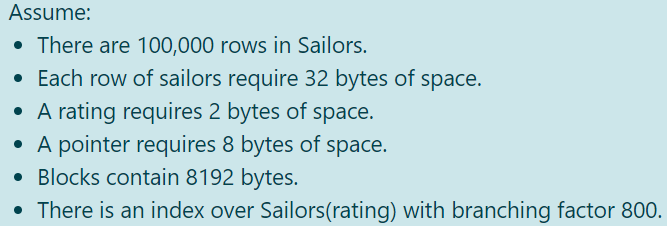
* What is the I/O complexity of adding a row to T?
* What is the I/O complexity of adding a row to S?
* What is the I/O complexity of deleting a row from T?
* What is the I/O complexity of deleting a row from S?
* **בוחן 3-**

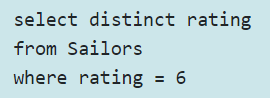


creating an index on B. block contains 8192 bytes. integer 4 bytes, each pointer 8 bytes.

* What will be the branching factor (d) chosen for the B+Tree index on T(B)?
* What is the **minimal** number of **children** that an internal node in the B+Tree has?
* What is the **maximal** number of **children** that an internal node in the B+Tree has?
* What is the **minimal** number of **values** that internal node in the B+Tree has?
* What is the **maximal** number of **values** that internal node in the B+Tree has?
* Assuming that T has 10,000 tuples, what is the **minimal** number of **leaf nodes** in the B+Tree?
* Assuming that T has 10,000 tuples, what is the **maximal** number of **leaf nodes** in the B+Tree?
* **בוחן 4-**

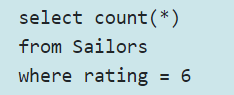
table Sailors(sid, sname, rating, age). Suppose that ratings are integer values between 1 & 10



* for curr SQL

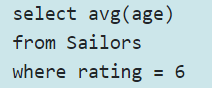
full table scan:

index scan:

* for curr SQL

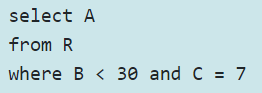
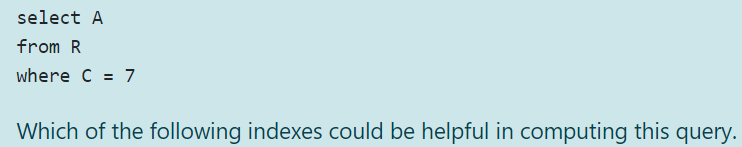
full table scan:

index scan:

* for curr SQL

full table scan:

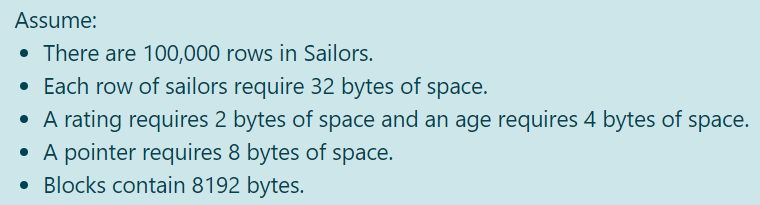
index scan:

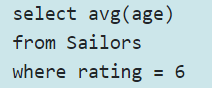
* **בוחן 6-**

b a

* for the first query, the most efficient index is because we match on c, and it saves us from having to retrieve a row.
* for the second query, the most efficient index is - once we reach leaf with , we need to scan right until we find and since is part of the index, this saves us from having to retrieve a row.

table Sailors(sid, sname, rating, age). Suppose that ratings are integer values between 1 & 10



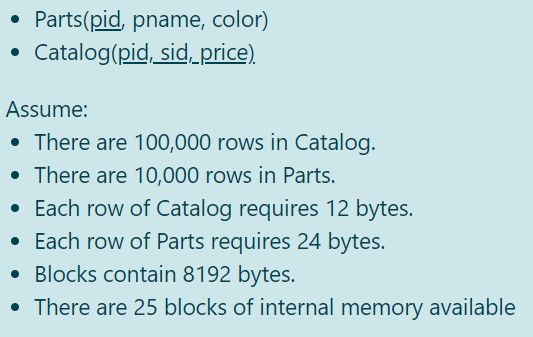
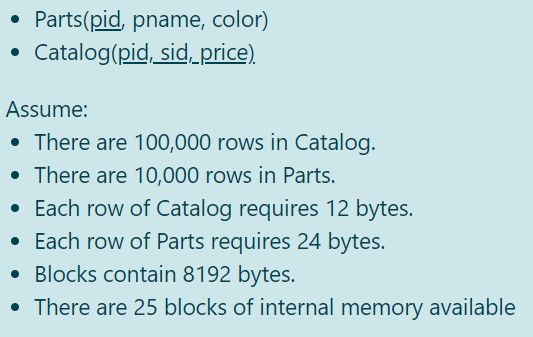
* We create an index over **Sailors(rating,age).** What is the optimal (=maximal) branching factor that can be used for such an index?
* for curr SQL

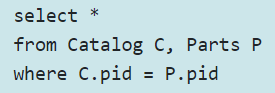
full table scan:

index scan:

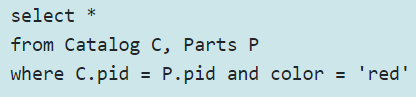
*\* Note:*we must access the blocks of the Sailors table, in order to read the age values. Normally, to compute the cost, we assume one disk access for each matching row, Sometimes it may occur, that the number of matching rows is larger than the total number of blocks in the table Sailor. In this case, you should compute the cost of accessing the rows as the min(number of matching rows, number of blocks in the table)

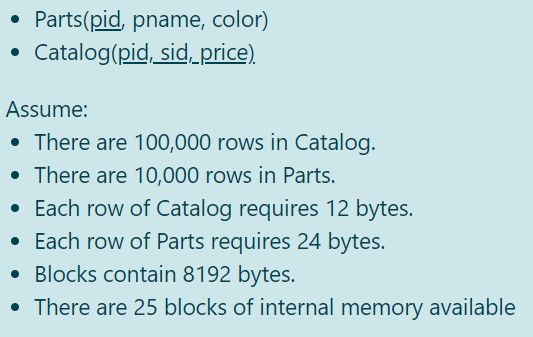
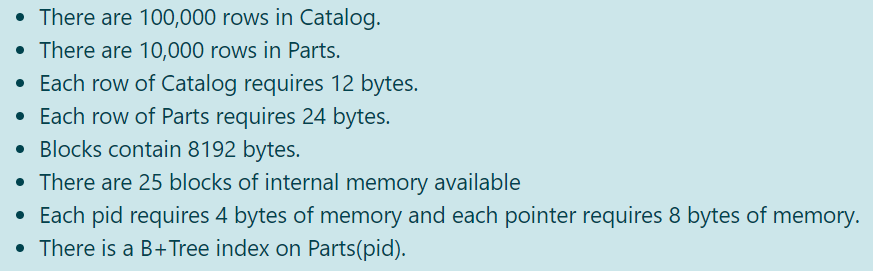
**שבוע 6:**

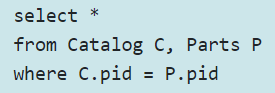
* **בוחן 1-**



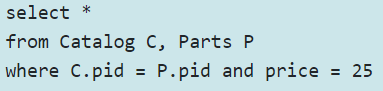
* What is B(Catalog)? (i.e., how many blocks is Catalog stored in?)
* What is B(Parts)?
* What is the cost of computing this query using **block nested loop**s join?

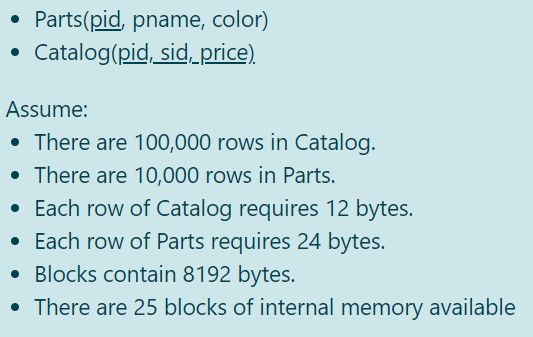
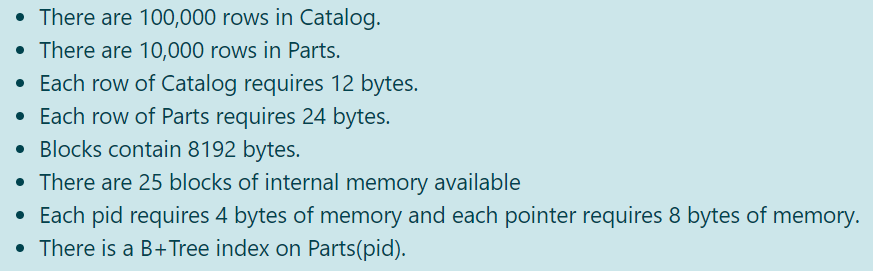


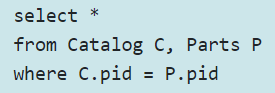
* What is the cost of this query using **block nested loops join**, the selection is computed **after** the join?
* What is the cost of this query using **block nested loops join**, the selection is computed **before** the join
* What is the smallest buffer size for which your answer to (2) does not change?
* **בוחן 2-**



* What is the cost of computing this query using index nested loops join?

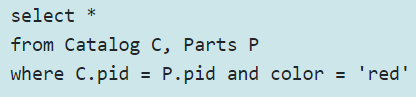


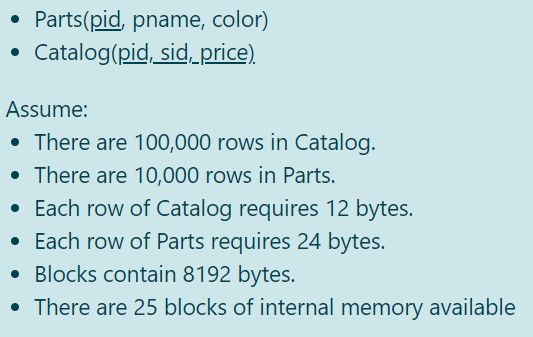
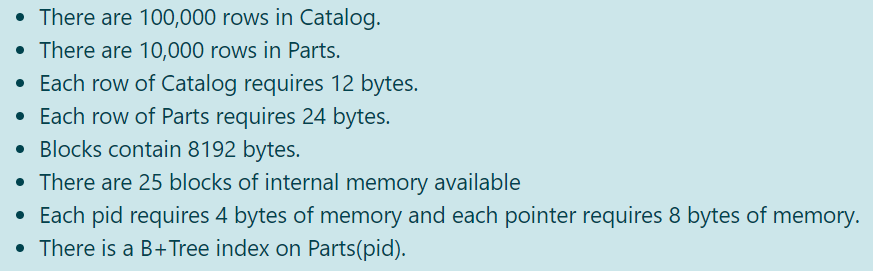
* What is the cost of this query using **INL**, the selection is computed **after** the join?
* What is the cost of this query using **INL**, the selection is computed **before** the join?
* How would your answer to (2) change, if the selection condition was price > 25?
* **בוחן 3-**



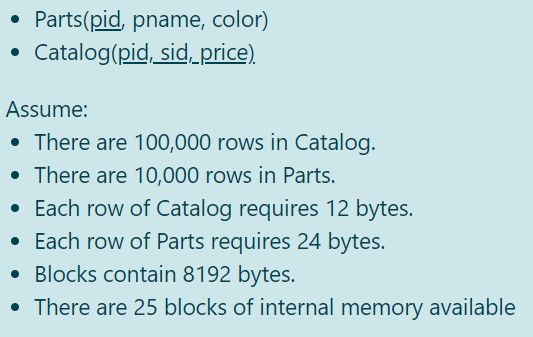
* Can we use *hash join* to compute this query if there are 25 blocks of internal memory available? **YES**
* Can we use *hash join* to compute this query if there are 4 blocks of internal memory available? **NO**
* What is the **smallest size of internal memory** for which it is still possible to compute the query using hash join?
* What is the cost of computing the query with hash join?

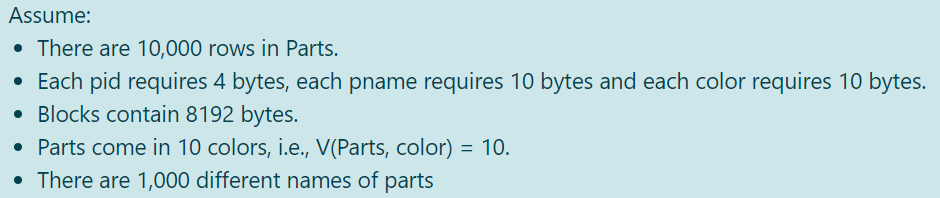


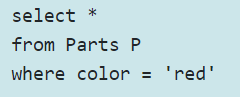


* Can we use *hash join* to compute this query if there are 25 blocks of internal memory available? **YES**
* Can we use *hash join* to compute this query if there are 4 blocks of internal memory available? **YES**
* What is the **smallest size of internal memory** for which it is still possible to compute the query using hash join?
* What is the cost of computing the query with hash join?
* **בוחן 4-**
* Can we sort **Catalog** with 2 phase external merge sort if there are 25 blocks of internal memory available? **YES**
* Can we sort **Catalog** with 2 phase external merge sort if there are 10 blocks of internal memory available? **NO**
* What is the **min number of blocks** needed to sort Catalog with 2 phase external **merge sort**?
* What is the cost of sorting with merge sort?
* Can we sort **parts** with 2 phase external merge sort if there are 25 blocks of internal memory available? **YES**
* Can we sort **parts** with 2 phase external merge sort if there are 10 blocks of internal memory available? **YES**
* What is the **min number of blocks** needed to sort parts with 2 phase external **merge sort**?
* What is the cost of sorting parts with merge sort?

**שבוע 7:**

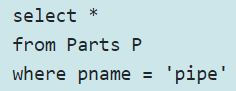
* **בוחן 1-**





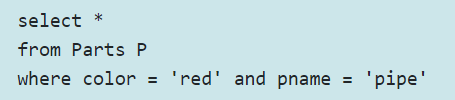
* Number of rows:

Number of blocks:



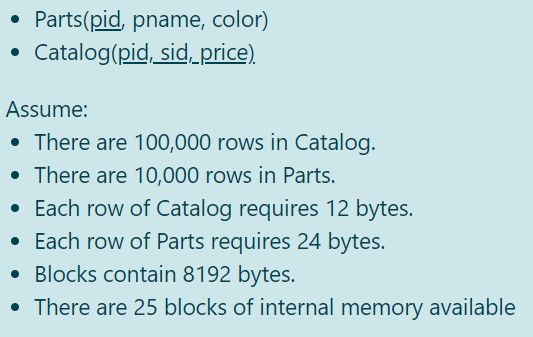
* Number of rows:

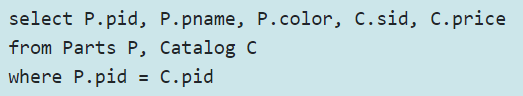
Number of blocks:



* Number of rows:

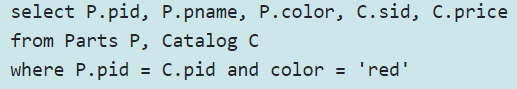
Number of blocks:





* Number of rows:

Number of blocks:



* Number of rows:

Number of blocks: