Experiments

2.3. Experiment

- Our implementation of fixed_len_sizeof returns the number of bytes in an specific record/tuple. Because of that, if a tuple with 10 characters is supplied to the program, it will compute 10 bytes as the return value for it.
  - 100 attributes * 10 bytes = 1.000 bytes

3.2. Experiment

- The (record / second) performance decreases more significantly for the reading process (since we are writing sequentially in the end of file). The reading performance decreases according to the number of new tuples added (Page x Number of records).
- Both methodologies determine an specific number of bytes to be store. Having said that, as bigger as the block_size gets the I/O cost goes expressively down for the reading process since the record is written in an linear way in the disk (good for sequential access).
- The page based method stores the records in a more compressed way, removing separators and break lines and taking more advantage of the file space (page is a raw file). Also, together with the heap file abstraction and indexes, pages can be easily maintained and records can also be found with less overhead.
- Removing, searching or updating an record can be a really cost full task without a RID (record id), since we need to do a full scan over the entire file to find an entry. If pages were organized using one, or more indexes structure it would be probably more efficient.

4.3. Experiment

- Since select uses a filter attribute (the same as the fetched attribute), we will need to full scan the entire universe of pages (for this attribute), bring it to buffer and do extra computing to check our WHERE statement. As page size increases, the full scan will take more time (Pages x Number of records) since we need to bring each page to memory and do the comparison.
- As the range gets wider and page size grows, the bigger will be the page that we will need to output to the users. Having said that, we will fetch more results and it takes more processing time and memory (buffer memory). Also, as output grows, we have less space to fetch the remaining pages located on the disk (the ones we still need to search).
- In conclusion, as page grows more we need to scan, and as filter gets wider, more we need to fetch to memory.

5.2. Experiment

- For a single attribute selection the col store is quite efficient, but for a complete tuple return, or multiple filters it can get inefficient. Having said that, since every attribute is stored in a different file, we have various pages with the same page size. As the pages grow, the performance drops considerably deal to the size factor.

- The csv2heapfile creates a single heap file containing all the tuples. In terms of space and organization it is more efficient than the csv2colstore. The writing process is also faster in the csv2heapfile since we are dealing with a single file. However, for selecting a single attribute you need to pull the entire file to memory (the whole tuple) which is quite inefficient. On the other hand, the csv2colstore is faster in terms of single attribute selections

.Concluding, you have a downside for csv2colstore during the writing process, but it has an upside during the single read process.

- The select2 over the column store method is a lot more efficient compared to the select over the standard heap file method. The reason for this is that we are selecting a single attribute (not the entire tuple) which gives column store a considerable advantage over the standard heap file. In the column store we are dealing with simple Tuple ID / Attribute files that makes the search faster and simple. However, the standard heap file carries Tuple ID / N Attributes that can lead to extra I/O to return certain attributes that are irrelevant for this specific query. Concluding, the select 2 has a better performance.

- Compared with the other selects, select 3 takes more time dealt to extra I/O for the filtering parameter (supposing a different attribute from the one being fetched). It needs to return the file that contains the attribute to be fetched, and nest loop it with another file that contain the filter attribute. Having said that, as wide as the range is we almost do a full scan on the "filter attribute file" for each entry on the "main attribute file". So not only the select 3 is more time consuming in terms of I/O, but also in terms of processing for each tuple.