CMPE321

Project 4

Ahmet Fırat Gamsız – 2020400180

Yunus Emre Özdemir - 2020400153

**Design**

* Page size: 10
  + This was already given in the project description. Pages are json files. They are formatted as:

{<primary\_key\_val\_1>: {<field\_name\_1>: <field\_val\_1>, <field\_name\_1>: <field\_val\_1>},

<primary\_key\_val\_2>: {<field\_name\_1>: <field\_val\_1>, <field\_name\_1>: <field\_val\_1>}, …}

* + As record header we included primary key value of that record. This is done to increase the speed of searching.
  + We didn’t include any page header. We thought it would be unnecesseray and repetitive in our design. Instead we used a catalogue (called catalogue.json). This file act as a general page header. It keeps type name, primary key, fields and their types (str, int) for each type.
* File size: 100
  + Since file is an abstract concept, we used folders (named as type name) as file. Each file has 100 pages in it as json files.
  + Our decision of using folders has multiple reasoning:
    - Keeping every page in a single file (file as in OS notation) is more prone to corruptions. Our structure is safer, when a single page is corrupted other pages are not affected. (Consider an incorrectly inserted curly brace in the file.)
    - Having a folder structure facilitates reading and editing text files. This structure makes the code less error prone because we didn’t need to use byte locations or pad entries to ensure length constraints on records.
    - This design enables us to use lookout tables for quickly searching records with primary keys.
  + Since we are not allowed to load whole files to RAM, this implementation have the same disk access problems (loading and unloading each page is an I/O operation and they are slow) as keeping all pages in a single file (OS notation).
  + We also keep metadata (metadata.json) and lookup table (lookup.json) in files. Metadata file keeps the number of records in each page and used to find empty slots when inserting new records. Lookup table is used for searching.
* Maximum number of fields type can have: ???
* Maximum length of a type name: ???
* Maximum length of a field name: ???

**Search Design**

Lookup tables, dictionaries, fast but using duplicate information etc.

**Brief Code Explanation**

How type/record created, how searched, how deleted

**Notes:**

* We assumed maximum 1000 entries can be stored for a type. A new file won’t be created when 1001th record inserted. We didn’t include this because it has been said that it won’t be tested. Implementation doesn’t introduce new logic but just requires extra checks on files.
* We made extra checks for given assumptions in the description. They were used for our testing purposes.

**Further Improvements and Setbacks**

Using hashing, probing, keeping bits for deleted slots, adding extra files etc.