CmpE 321 Group 20 Project 4 Report

Dağhan Erdönmez 2021400093 & Mehmet Emin Atak 2021400282

June 2025

1 Design

We tried to design the system with a clear structure as much as possible. In order to do that, we used `.txt` files and object oriented programming, in addition to `log.csv` file which stores the log records, as the name suggests. Below you will find the core elements of our design and how we handled them.

1.1 Catalog

Just like the real database management systems, we used a system catalog to keep track of type definitions, relations in other words. We implemented this system catalog via `catalog.txt`. Here is the structure of the system catalog for a type:

```
<type-name>|<num-fields>|<primary-key-index>|
<field1-name>:<field1-type>:<size>|
<field2-name>:<field2-type>:<size>|...
```

We used this catalog as a look-up table.

1.2 File&Page

We implemented the file concept for each type, using a different `<type-name>.txt`. Files consist of pages. The maximum number of pages per file is set to 100, while each page can store up to 10 records.
A page corresponds to a row in our `file.txt`. Below you may find how a page looks like.

```
<page-number>|<num-records>|bitmap|<record-1> <record-2> ...
```

A bitmap is a list consisting of 10 bits, each of them indicating whether that slot is occupied by a real record or not. An example is shown below:

```
1011000000 => slots 0, 2, 3 occupied
```

1.3 Record

Records are stored in the pages, one after another. We are using a validity bit at the beginning of each record, indicating whether that record really exists. Here is the structure of a record in our design:

```
<valid-bit><field-value-1><field-value-2>...
```

One of the most important decisions we made during our design is to assume that each field can have 25 characters at most. The reason we put this constraint is to ensure that we can separate these field values from each other. There is no special character between two fields. Therefore, in order to parse the records, we are just reading 25 characters for each field and stripping the whitespaces in case there are any.

1.4 Operations

The design level handling of the operations is like the following:

- While creating a new type, we first check if there is a type of the same name exists. If not, then we add the new type definition and create a new empty file named `<type-name>.txt`
- While creating a new record, we first check if the type specified exists. If everything is correct, we parse the fields and check if the primary key already exists in the already existing records by reading all of the pages one by one and comparing the primary keys. If there is no duplicate, we find a page with an empty slot and insert the new record there. If all of the pages are already full, then we add a new page and insert the record there.
- While deleting a record we search the page containing that primary key one by one and find that record in there. We set the validity flag of that record to 0, update the bitmap of the page and reduce the number of counts in that page.
- While searching a record, we again look for the record page by page using the primary key. If the record is found, we return the requested fields.

1.5 Logging&Output

In case of an output other than `None`, we write the result into `output.txt`. Please note that even if the operations do not give results -an input consisting of only create operations can be an example-, an `output.txt` is created. Additionally, we refresh the output file for each run, which means one can not see the outputs of the previous runs. However that does not mean the effects of those runs do not persist. One can create a type or record in a run and can achieve those in the upcoming runs.

For the log records, we are using a `log.csv` file. Unlike the `output.txt`, this file is not refreshed between the runs. In each run, new log records are appended. The structure of a log record is shown below:

```
<timestamp>,<operation>,<success/failure>
```

We treat a run as a batch of operations, meaning that the timestamps of each operation in a single input file are the same. Despite, the order of the log records is preserved. Also, the timestamps for operations in different runs are not the same. Timestamps are extracted via `time.time()` method in Python.

2 Assumptions & Decisions

- A field value can have at most 25 characters.
- A file can have at most 100 pages.
- A page can have at most 10 records.
- Operations in a single run have the same timestamp.
- For each type, a txt file corresponding to a File structure is created.
- The lines (corresponding to a page) of each file is read one by one, representing the idea of ”one page at memory at a single time”.