Project 4: Design & Implementation of the Dune Archive System

CMPE 321, Introduction to Database Systems, Spring 2025

Due: 10.06.2025 23:59 o'clock

1 Introduction

The planet Arrakis, also known as Dune, is the only known source of the valuable spice "Melange," a substance critical to space travel, life extension, and prescient abilities. After decades of intense political intrigue and battles among noble Houses vying for control of Arrakis, much of the detailed records about these Houses, the Fremen tribes, sandworms, spice fields, and weapons have been fragmented or lost.

To preserve the fragile history and essential knowledge of Arrakis, the Imperial Archivists and Bene Gesserit Sisters have commissioned the creation of the "Dune Archive" — a robust information system that will securely store vital data regarding noble Houses, tribes, spice production, and other critical elements of the planet.

As the appointed scholars of the Imperial Archive, your task is to design and implement this system, ensuring that records can be created, updated, searched, and deleted efficiently and reliably. Your work will safeguard the legacy of Arrakis for generations to come.

2 First Phase: Design

In this phase, you are expected to design the **Dune Archive** system. Your design should specify how data will be stored in files, pages, and records, following the principles of a simple DBMS.

The system must support the following operations:

Dune Definition Language Operations

- Create a type (i.e., a relation or table)

Dune Manipulation Language Operations

- Create a record
- Delete a record

- Search for a record by primary key

Your design should include, but is not limited to, the following aspects:

- Definition of page size, i.e., the maximum number of records per page (up to 10)
- Definition of file size, i.e., the maximum number of pages per file (decide and clearly specify)
- Page organization should follow the unpacked slotted page format, where each record occupies a slot, and the page maintains a bitmap indicating occupied slots.
- Records must have fixed length, with field lengths defined and stored in a system catalog file.
- Specification of information stored in page headers (e.g., page number, number of records, free space indicators) and record headers (e.g., validity flags)
- Maximum number of fields per type (the maximum should be at least 6)
- Maximum length of type names (at least 12 characters)
- Maximum length of field names (at least 20 characters)

Assumptions

- All type names, field names, and string field values consist of alphanumeric characters.
- Integer fields contain integer values (positive or negative).
- Only `int` and `string` field types are supported.
- Each page may contain up to 10 records.
- The term “type” refers to a database relation (table).
- You may make additional assumptions if necessary, but these must not contradict the base assumptions and must be clearly documented and justified in your report.

Constraints

- Each type (relation) must be stored in a separate file, which contains multiple pages.
- Data must be organized in pages, and pages must contain records. You must clearly explain your page and record structures in your report.
- Each page can hold a maximum of 10 records.
- The primary key field must be used as the search key for all search and delete operations.

- Every create and delete record operation must update the corresponding page and file accordingly.
- Although a file contains multiple pages, you must load pages individually as needed (e.g., for searching). Loading the entire file into memory is not allowed.

3 Second Phase: Implementation

In this phase, you are expected to implement the **Dune Archive** software that you have designed in Phase One. The implementation must support the Dune Definition Language Operations and the Dune Manipulation Language Operations.

4 Input/Output

Submissions will be graded automatically for various test cases. You will write your code in Python 3. The input file's full path will be given as an argument to the executable program. The command that will be executed after unzipping your submission at the same location is as follows:

```
python3 2019400XXX/archive.py fullinputfilepath
```

The input file contains a list of operations. Each line corresponds to an operation. The format for each operation is as follows:

4.1 Dune Definition Language Operations

Table 1: Input and Output Formats of Create Type Operation

Operation	Input Format	Output Format
Create	create type <type-name><number-of-fields><primary-key-order><field1-name><field1-type><field2-name>..	.None

4.2 Dune Manipulation Language Operations

Table 2: Input and Output Formats of Create, Delete and Search Operations For Records

Operation	Input Format	Output Format
Create	create record <type-name><field1-value><field2-value>...	None
Delete	delete record <type-name><primary-key>	None
Search	search record <type-name><primary-key>	<field1-value><field2-value>...

4.3 Sample Input, Output and Log Files

```
create type house 6 1 name str origin str leader str military_strength int wealth int
    spice_production int
create record house Atreides Caladan Duke 8000 5000 150
create type fremen 5 1 name str tribe str skill_level int allegiance str age int
create record fremen Stilgar SietchTabr 9 Atreides 45
create record house Harkonnen GiediPrime Baron 12000 3000 200
```

```
delete record house Corrino
search record fremen Stilgar
search record house Atreides
```

Listing 1: Sample Input (input.txt)

```
Stilgar SietchTabr 9 Atreides 45
Atreides Caladan Duke 8000 5000 150
```

Listing 2: Sample Output (output.txt)

```
1635018009, create type house 6 1 name str origin str leader str military_strength int wealth int spice_production int, success
1635018014, create record house Atreides Caladan Duke 8000 5000 150, success
1635018016, create type fremen 5 1 name str tribe str skill_level int allegiance str age int, success
1635018018, create record fremen Stilgar SietchTabr 9 Atreides 45, success
1635018020, create record house Harkonnen GiediPrime Baron 12000 3000 200, success
1635018022, delete record house Corrino, failure
1635018025, search record fremen Stilgar, success
1635018027, search record house Atreides, success
```

Listing 3: Sample Log File (log.csv)

5 Implementation Instructions

- Any field of a type can be the **primary key**, and it is defined with *primary-key-order* in the create type operation (see Table 1).
- Search and deletion of records shall always be done by primary key.
- Test cases are not necessarily independent of each other. For example, if the type *house* is created in test case 1, it should be accessible in test case N.
- The system should log all definition and manipulation operations into a CSV file (named `log.csv`), which includes the UNIX time of occurrence, the operation string, and the status of the operation (either `success` or `failure`). The file should **NOT** be present in the submission and **WILL NOT** be present at the first runtime.
  - The log file must be persistent and never deleted when the Dune Archive software is stopped or restarted.
  - The UNIX timestamps in the sample test cases are deliberately set for you to understand the time flow and operation order. In real life, user inputs may have varying intervals, and your system may record identical timestamps for rapid consecutive operations; this is acceptable.
  - Use `import time` and `int(time.time())` in Python to generate UNIX timestamps.
- The system should correctly handle `create type`, `create record`, `delete record`, and `search record` operations and log the success or failure of each operation.
- The status of an operation is decided based on whether the result is empty or not. For search operations, if the output is empty (i.e., the record is not found),

the operation must be considered a failure and logged accordingly. The invalid operations resulting in failure are:

  - Creating a type with an existing name.
  - Creating a record with a primary key that already exists.
  - Deleting a type that does not exist.
  - Deleting a record with a primary key that does not exist in that type.
  - Searching for a non-existing record.

- All outputs must be printed to `output.txt` file **NOT** to the screen (standard output) or the log file.

6 Submission

- This project can be implemented either individually or as a team of two people.
- The submission must include your report (`report.pdf`), source code (`archive.py`), and a README (`README.txt` or `README.md`) file that describes how to run your code.
- You **MUST** include all your design decisions and detail your approach in the PDF report.
- Include comments in your code file as it will be graded.
- Your directory must have the structure as shown below:

```
2019400XXX_2019400XXX.zip
└── 2019400XXX_2019400XXX
    ├── archive.py
    ├── README.md
    ├── report.pdf
    └── StudentID_Contribution.pdf
```

- Name the folder as **StudentID1_StudentID2** if you are working as a group.
- Name the folder as **StudentID1** if you are working individually.
- **If working as a group, each member must submit the ZIP file separately on Moodle.**
- Zip the folder for submission and name the `.zip` file with the same name as the folder.
- Submit the `.zip` file through **Moodle** before the deadline.
- **No late submissions will be accepted.**
- Each group should submit **one** `.zip` file per member.

- Do not include any other files in your `.zip` file.
- **Any other submission method is not allowed.**
- 20 points will be deducted for non-compliance with **ANY** of the naming and folder conventions explained above.
- Note that submissions will be inspected for plagiarism with previous years' submissions as well as this year's. Any sign of plagiarism will be penalized.

Individual Contribution Report Guidelines

Each student must write their own **Individual Contribution Report** in PDF format and include it in the ZIP file. The report should be named as:

`StudentID_Contribution.pdf`
(e.g., `2023789_Contribution.pdf`)

What to include:

- **Personal Info:** Name, Student ID, Group Number
- **Contributions:** Tasks you worked on (e.g., trigger logic, match constraints, UI forms, login system)
- **Teamwork:** How the team collaborated and divided responsibilities
- **Self-Reflection:** Your learning experience and skills improved

Formatting:

- **Length:** 1 page (max 2 pages)
- **Font & Size:** Times New Roman, 12pt, 1.5 spacing
- **Format:** PDF

Note: If one team member fails to submit their contribution report, they will not be credited for the project submission.

7 Late Submission Policy

We will not accept late submissions for this project. Deadline is strict. We will not accept any submission other than Moodle.

8 Academic Honesty

Please read carefully the academic honesty part of the syllabus as we give utmost importance to academic honesty.