Database Files and Indexing

Database Engine

Database Design Document

CS 6360.5U1 Team project

Team Red

Team Members,

Sangeetha Tatineni

Ben Forrest Monical

Andrew David Byers

Ameya Nandakumar Potey

Aarthi Gunasekaran

Objective

Objective of this project is to implement a rudimentary database engine that is based on a simplified file-per-table variation on the SQLite file format.

Design Approach

A simple modular design approach was adopted for this project to enable the team to work individually on each data base operation. The source code for the database engine was developed in a collaborative fashion with non-linear workflow. Multiple design meeting, status meetings and collective testing were the key for the successful completion of the project.

Design Decisions

- Each DB operation was developed in a separate java class and all methods within these files are made static which could be called without taking an instance of a class.
- A decision was made to develop an 'utils class' with general methods that could be used by all other java files. The general methods include, but not limited to,
    - Code to check if the table exists
    - Extract cell offsets of a page
    - Get last cell index of a page
    - Increment record count of a page
    - Get row id of most recent record
    - Append Bytes to an array list
    - Insert cell bytes into the current page
    - Calculate the cell position in a page for an insertion
    - Check if page overflows
    - Extract columns names
    - Trim Array fields
- GIT software tool was used for tracking the changes made by the team.
- Datatypes, their sizes and the serial type codes in this DB engine were adhered to the "DavisBase Nano File Format Guide".
- Page Header format and each cell/record format was also derived from "DavisBase Nano File Format Guide".
- Not support [UNIQUE] and [PRIMARY KEY] constraints in the create command due to the limited time.

- Due to time constrains, decision was made to adopt serial search approach to find a record from multiple pages of a file instead of B+ tree.
- Database catalog (meta-data information) is basically contained in two base or mandatory tables called “davisbase_tables.tbl” and “davisbase_columns.tbl”.
- The team has made a design decision on not to add the two meta-data catalog files namely “davisbase_tables.tbl” and “davisbase_columns.tbl” into catalog itself.
- Decision was made to display the row id to the user, along with other information in the record. This is to help used to perform future operations.

Design Assumptions

- Row id is always the primary key for all the tables.
- User will not provide the [UNIQUE] and [PRIMARY KEY] constraints in the CREATE command.
- User will not provide any empty spaces between the column names and their values of INSERT command. The example command format is as below,
    - INSERT INTO TABLE (Contact_id,Fname,Mname,Lname) CONTACT VALUES ('1','Cindra','Del','Philipeaux');
- SELECT, DELETE and UPDATE only work for the WHERE condition with row id.
- UPDATE command deletes the existing record and creates a new record with new row id.

Programming language

Java

Framework

Eclipse IDE
Windows command prompt

Functionalities

DDL, DML, and DQL commands are supported in the Davisbase engine. Below are the supported syntax for the commands.

Create Table:

CREATE TABLE Task (

```
        Name TEXT NOT NULL,
        Description TEXT NOT NULL,
        DueDatetime DATETIME NOT NULL
);
```

Insert:

```
INSERT INTO TABLE (Name, Description, DueDatetime)
Task VALUES (
        'Name of the task',
        'Task description',
        '2021-05-21'
);
```

Update:

```
UPDATE Task
SET Name = 'New task name'
WHERE rowid < 2;
```

Delete:

```
DELETE FROM TABLE Task
WHERE rowid = 1;
```

Select:

```
SELECT
FROM Task
WHERE rowid=1;
```

Drop:

```
DROP TABLE Task;
```

Conclusion,

With the above-mentioned design decisions and design assumption, a rudimentary database engine that is based on a simplified file-per-table was successfully accomplished by this team. The database engine supports creation/deletion of tables, insertion of records, updating the record and querying for the data. However, due to the time constrain, some of the features like creating index, B+ tree approach for quick search of data were not implemented.