Lemondb Database Service

Company Lemonion Inc. Team Team 1

- Zhang Yichi
- Zhao Wenhui
- Zhou Runqin
- Zhu Yuankun

Introduction

Lemondb is a relational database management system produced by Lemonion Inc. It serves to opearates on databases based on queries, i.e. some specific instructions. It also features on the multithreading operation. This documentation will help you get more familiar with the product and could be a reference for you in using.

Install

Together with this documentation, the source code of the software is given. Please just use Cmake to build this project based on the given CMakeLists. Then the binary **lemondb** will be generated and you could use it.

Data

Lemondb serves for relational database system, which means the data are stored as set of **tables**. The database can store multiple tables in the same time, each table with a unique name and fields. Every tuple (row) in the table has a key value, which is the main of the tuple in query operation. And each tuple will have values for every field.

The raw source of data are stored in a .tbl file in the following form:

> Tablename NumberofFields KEY FieldName1 FiledName2 ... k1 v11 v21 ... k2 v21 v22 ... ...

Queries

Queries are the basic instructions that users can use for the system. Following is the manifest of queries that user can use to operate the software.

LOAD

> LOAD tableFilePath;

Load a table from a file located at tableFilePath.

COPYTABLE

> COPYTABLE Table NewTable;

Creates a copy of the existing table table, and named it NewTable. The new copy will contain all the records and fields from the original table.

DUMP

> DUMP table filePath;

Dump a table Table into a file located at filePath.

DROP

> DROP Table;

Delete the table Table along with all its content.

TRUNCATE

> TRUNCATE Table;

Delete all the content of the table Table, i.e. removes all the records from this table.

DELETE

> DELETE ( ) FROM table; DELETE ( ) FROM table WHERE (cond) ...;

Delete rows from the table table. This query is a conditional query.
It will print **Affected <n> rows**. to standard output, where <n> is the number of rows that were deleted

INSERT

> INSERT (key value1 value2 ...) FROM Table;

Insert the row (key value1 value2 ...) into the table Table

UPDATE

> UPDATE (Field Value) FROM Table WHERE (cond) ...;

Update the field Field of the rows satisfying the conditions with new value Value in Table. This query is a conditional query.
It will print **Affected <n> rows**. to standard output, where <n> is the number of updated rows.

SELECT

> SELECT (KEY field ...) FROM table WHERE (cond) ...;

For each record satisfying the condition print the fields specified in the SELECT clause. Each field that is not a KEY may appear at most once. The KEY field will always appear at the beginning. This is a conditional query.
It will print each record in the format (KEY field1 field2 field3 ...). The records will be displayed in ascending lexical order, sorted by KEY. If no record satisfies the condition nothing will be printed.

SWAP

> SWAP (field1 field2) FROM table WHERE (cond) ...;

This query swaps the values of field1 and field2 for the records of table that satisfy the given condition. There is no restriction on the fields. When they are the same the query does nothing. It will print **Affected <n> rows.** to standard output, where <n> is the number of rows updated.

DUPLICATE

> DUPLICATE ( ) FROM table WHERE (cond) ...;

This query copies the records satisfying the condition in table table. This query is a conditional query. The affected records are inserted into the table, with key originalKey_copy. If a copy of a record already exists the copy is not overwritten, however the copy can be duplicated into originalKey_copy_copy.
On success print **Affected <n> rows.** to standard output, where <n> is the number of rows updated.

SUM

> SUM (fields ...) FROM table WHERE (cond) ...;

This query aggregates records that satisfies the given conditions. This is a conditional query. The SUM clause sums the values of one or more fields given in fields over all the affected records. The KEY field cannot be summed over.
It will print **ANSWER = (<sumFields> ...)** to standard output, where <sumFields> represents the sum of the fields, in the order specified in the query.

COUNT

> COUNT ( ) FROM table WHERE (cond) ...;

This query counts the number of records that satisfies the conditions. This is a conditional query.
It will print **ANSWER = <numRecords>** to standard output, where <numRecords> represents the desired count. If no record is affected, then the count is zero.

MIN/MAX

> MIN (fields ...) FROM table WHERE (cond) ...; MAX (fields ...) FROM table WHERE (cond) ...;

MIN clause finds the minimum value among all affected records for the values of one or more fields given in fields. The KEY field is considered not comparable thus will not appear in the MIN clause.
It will print **ANSWER = (<minValues> ...)** to the standard output, where <minValues> represents the minimum for each of the fields, in the order specified in the query.
The MAX query works in similar way, but finds the maximum instead of the minimum.

ADD/SUB

> ADD (fields ... destField) FROM table WHERE (cond) ...; SUB (fieldSrc fields ... destField) FROM table WHERE (cond) ...;

These two queries perform arithmetic operations on records that satisfy the conditions. Both are conditional queries. The ADD clause sums up one or more fields given in fields and store the result in the destField. The SUB clause subtracts the zero or more values of fields field, from the fieldSrc field, and stores the result in destField.
It will print **Affected <n> rows.** to standard output, where <n> is the number of rows affected.

LISTEN

> LISTEN (path_of_file);

Read queries from the file pointed to by path_of_file. Either an abosulte or a relative path can be provided.
On success, the program will print **ANSWER = (listening from <file>).**
On an error, it will print **Error: could not open path_of_file**, and carry on the work without this file.

QUIT

> QUIT;

Quit the database. Wait for running queries to complete.

Read Queries from File

If you have large transaction of queries to do, you can save it in a file and use the following opetion when starting the program

> --listen <filename>

Then the program will directly use the queries stored in the file.

Multithread Working

If you want to use multiple threads to read the file, please use the following command option

> --threads <int>

Working Progress

- 2018.11.16: LISTEN queries implemented, all multithread queries implemented. The speed can be further improved..
- 2018.11.09: Multithread mechanism designed.
- 2018.11.02: Queries Implemented.

Credits

We want to thank those experienced company members who offers much help to us on the implementation of this program:

- Prof. Manuel Charlemagne
- TA Liu Yihao
- TA Su Hang