

# mydb 数据库设计报告

朱倬民 2012011894

温和 2013011407

2016 年 1 月 3 日

## 1 功能介绍

mydb 是一个简单的面向单用户、单线程的关系数据库管理系统，支持增、查、删、改四项基本数据库查询功能，通过 SQL 的一个子集实现数据库与用户的交互。除了支持基本的增、查、删、改功能外，mydb 具有如下扩展功能：

1. 索引：使用 `std::multimap` 实现，支持对任意列建立索引加速约束检查；
2. 域完整性约束：支持 `CHECK` 语句的创建，并在更新数据库时进行约束检查；
3. 外键约束：支持 `FOREIGN KEY` 的创建，并在更新数据库时进行约束检查；
4. 模糊查询：支持使用 `LIKE` 语句进行模糊匹配；
5. 支持多表连接查询，包括三个表以上的连接；
6. 聚集查询：支持将 `SUM`, `AVG`, `MAX`, `MIN` 这四个函数应用于表中的某些列；
7. 分组聚集查询：支持查询时的 `GROUP BY` 语句，可以处理聚集查询与分组聚集查询并存的查询；
8. 完整的建表属性约束：建表时除了允许 `NOT NULL`, `PRIMARY KEY`, `FOREIGN KEY` 与 `CHECK` 约束外，还可以设置 `UNIQUE`, `AUTO_INCREMENT` 与 `DEFAULT` 约束；
9. 在查询时，支持 `BETWEEN` 和 `IN` 操作符；
10. 远程连接：允许远程开启 mydb 服务端后，本地用 mydb 客户端进行连接，通过 SQL 语句交互的方式进行数据库操作；
11. 采用 MySQL 风格打印查询结果，易于用户查看。

## 2 系统架构

mydb 的顶层设计架构如图 1所示。

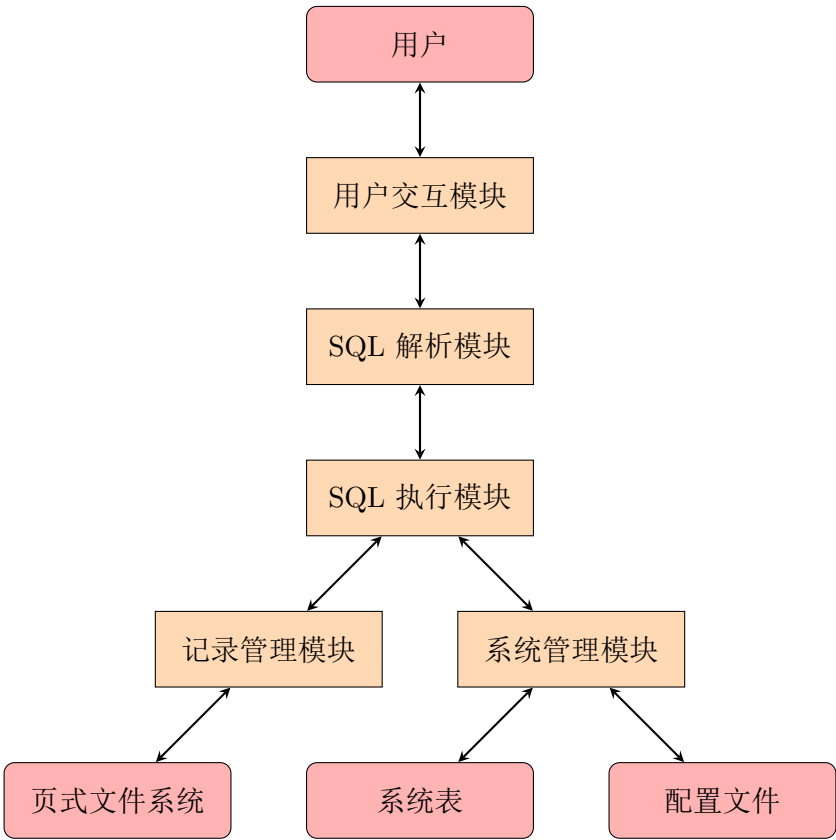


图 1: 系统架构

## 3 模块设计

接下来我们详细介绍各模块的设计与实现方法。

### 3.1 记录管理模块

在数据存储上，某个数据库 (database) 的某一张表 (table) 对应于与数据库同名的目录下与表同名的一个文件。在访问该表时，调用提供的页式文件系统进行文件读写，其中，每页大小均为 8KB。我们定义，该文件对应的各页中，第 1 页 (即编号为 0 的页) 存储与记录 (record) 相关的元数据，其数据格式如表 1所示。

表 1: 元数据记录格式

偏移量 (字节)	记录信息
0	每条记录长度
4	记录总个数
8	最后一条记录的 ID (rid)
12	最后一页的编号 (pid)
1024	空闲页位图

其中，空闲页位图大小总计 7K，用 1 表示空闲，0 表示占用。之后的每一页，均用于存储各条记录的数据。这些页的前 32 字节为空闲位图，记录该页哪些位置处可以用来存放新的记录，共计 256 位，分别对应该页的每 32 个字节 (称为一个字) 是否可用。因此，所有的记录必须按一个字对齐。对于具有  $n$  列的一条记录，其数据格式如表 2 所示。

表 2: 记录数据格式

偏移量 (字节)	记录信息
0	记录 ID (rid)
4	各字段是否为空的位图
$4 + n$	记录的数据

对于记录的数据，我们按照列的顺序依次存储，我们支持如表 3 所示的数据类型。

表 3: 支持的数据类型

类型	映射的 C++ 类型	长度 (字节)
BOOL	bool	1
SHORT	int8_t	2
INT	int16_t	4
LONG	int32_t	8
FLOAT	float	4
DOUBLE	double	8
CHAR	char	1
STRING	char*	最大 65535

每条记录的数据部分最长支持 65535 字节。

对于索引，我们将每个表的索引单独存放在相应文件中，存放格式为将 `multimap` 中的 key-value 对的二进制表示直接保存。

### 3.2 系统管理模块

我们将每个数据库中包含的所有结构信息存放在该数据库目录下的一个文件，`tablelist.dat` 中，存储方式为序列化后得到的 JSON 字符串。

- 对于每个数据库，我们记录它的名字，有几个表，以及每个表的结构信息。
- 对于每个表，我们记录表名，每条记录的长度，以及表中的列与约束。
- 对于每个列，我们记录它的编号、列名、数据类型、列宽以及在记录中的偏移量。
- 对于每个约束，我们记录它对应的列编号（若跨多列则为任意非法值）、约束类型以及约束相关的数据。

在执行 `USE` 操作时，系统管理模块会将有关这个数据库的所有结构信息全部读取出来，以供查询执行模块使用。对数据库结构进行的所有更改，包括新建、删除数据库，都会在系统退出时记录到磁盘中。

本模块同时提供对数据库结构更改的支持，如 `CREATE / DROP` 等。

### 3.3 SQL 解析模块

我们利用了 `flex` 和 `bison` 工具构建 SQL 语句解析前端，将 SQL 语句转换为自定义的语法树结构。对于语法正确的语句，我们采用 Visitor 设计模式，对其进行遍历，检查可能出现的类型不一致、数量不一致等错误。然后将其送往 SQL 执行模块执行，并获取执行后返回的结果。若执行成功，利用 MySQL 风格 (如图 2) 打印出查询结果；否则，打印出错误信息。

```
mydb> select * from customer;
```

customer.gender	customer.name	customer.id
'F'	'CHAD CABELLO'	300001
'F'	'FAUSTO VANNORMAN'	300002
'M'	'JO CANNADY'	300003
'F'	'LAWERENCE MOTE'	300004
'F'	'RODERICK NEVES'	300005
'M'	'JACOB LEDGER'	300006

图 2: MySQL 风格打印

接下来我们列出 `mydb` 支持的 SQL。

### 3.3.1 mydb 关键字

(case not sensitive)

DATABASE	DATABASES	TABLE	TABLES	SHOW	CREATE
DROP	USE	CHECK	PRIMARY	KEY	UNIQUE
NOT	NULL	AUTO_INCREMENT	INSERT	INTO	VALUES
DELETE	FROM	WHERE	UPDATE	SET	SELECT
GROUP	BY	IS	IN	BETWEEN	LIKE
AND	OR	SUM	AVG	MAX	MIN
INT	SMALLINT	BIGINT	FLOAT	REAL	DOUBLE
VARCHAR	STRING	CHAR	BOOLEAN	DATETIME	DESC
FOREIGN	REFERENCES	INDEX	DEFAULT	TRUE	FALSE

### 3.3.2 mydb 文法

$\langle \text{program} \rangle$	::= $\langle \text{stmt} \rangle [\langle \text{stmt} \rangle]^*$
$\langle \text{ident} \rangle$	::= $[A-Za-z\_][A-Za-z0-9\_]^*$
$\langle \text{integer} \rangle$	::= $(+ -)?[0-9]^+$
$\langle \text{length} \rangle$	::= $[0-9]^+$
$\langle \text{real} \rangle$	::= $(+ -)?[0-9]^+.[0-9]^+$
$\langle \text{string} \rangle$	::= $'.*'$
$\langle \text{bool} \rangle$	::= <b>TRUE</b>   <b>FALSE</b>
$\langle \text{value} \rangle$	::= $\langle \text{integer} \rangle$   $\langle \text{real} \rangle$   $\langle \text{string} \rangle$   $\langle \text{bool} \rangle$   <b>NULL</b>
$\langle \text{stmt} \rangle$	::= $\langle \text{sysStmt} \rangle$ ;   $\langle \text{dbStmt} \rangle$ ;   $\langle \text{tbStmt} \rangle$ ;   $\langle \text{idxStmt} \rangle$ ;
$\langle \text{sysStmt} \rangle$	::= <b>SHOW DATABASES</b>
$\langle \text{dbStmt} \rangle$	::= <b>CREATE DATABASE</b> $\langle \text{dbName} \rangle$   <b>DROP DATABASE</b> $\langle \text{dbName} \rangle$   <b>USE</b> $\langle \text{dbName} \rangle$   <b>SHOW TABLES</b>

$\langle tbStmt \rangle ::= \text{CREATE TABLE } \langle tbName \rangle ( \langle field \rangle [, \langle field \rangle]^* )$   
 $\quad | \text{ DROP TABLE } \langle tbName \rangle$   
 $\quad | \text{ DESC } \langle tbName \rangle | \text{ SHOW TABLE } \langle tbName \rangle$   
 $\quad | \text{ INSERT INTO } \langle tbName \rangle [ ( \langle colName \rangle [, \langle colName \rangle]^* ) ] \text{ VALUES } ($   
 $\quad \quad \langle values \rangle ) [, ( \langle values \rangle )]^*$   
 $\quad | \text{ DELETE FROM } \langle tbName \rangle \text{ WHERE } \langle boolExpr \rangle$   
 $\quad | \text{ UPDATE } \langle tbName \rangle \text{ SET } \langle colName \rangle = \langle expr \rangle [, \langle colName \rangle = \langle expr \rangle]^*$   
 $\quad \quad \text{WHERE } \langle boolExpr \rangle$   
 $\quad | \text{ SELECT } \langle selectors \rangle \text{ FROM } \langle tbName \rangle [, \langle tbName \rangle]^* [\text{WHERE } \langle boolExpr \rangle]$   
 $\quad \quad [\text{GROUP BY } \langle colName \rangle]$

$\langle tbName \rangle ::= \langle ident \rangle$

$\langle colName \rangle ::= \langle ident \rangle$

$\langle field \rangle ::= \langle colName \rangle \langle type \rangle [ \langle attr \rangle ]^*$   
 $\quad | \text{ CHECK } ( \langle boolExpr \rangle )$   
 $\quad | \text{ PRIMARY KEY } ( \langle colName \rangle )$   
 $\quad | \text{ FOREIGN KEY } ( \langle colName \rangle ) \text{ REFERENCES } \langle tbName \rangle ( \langle colName \rangle$   
 $\quad \quad ) )$

$\langle type \rangle ::= \text{INT } [ ( \langle length \rangle ) ]$   
 $\quad | \text{ SMALLINT}$   
 $\quad | \text{ BIGINT}$   
 $\quad | \text{ FLOAT } | \text{ REAL}$   
 $\quad | \text{ DOUBLE}$   
 $\quad | \text{ VARCHAR } ( \langle length \rangle )$   
 $\quad | \text{ STRING}$   
 $\quad | \text{ CHAR}$   
 $\quad | \text{ BOOLEAN}$   
 $\quad | \text{ DATETIME}$

$\langle attr \rangle ::= \text{NOT NULL } | \text{ UNIQUE } | \text{ AUTO\_INCREMENT } | \text{ DEFAULT } \langle value \rangle$

$\langle values \rangle ::= \langle value \rangle [, \langle value \rangle]^*$

$\langle boolExpr \rangle ::= \langle col \rangle \langle boolOp \rangle \langle expr \rangle$   
 $\quad | \langle col \rangle \text{ IS } [\text{NOT}] \text{ NULL}$   
 $\quad | \langle col \rangle [\text{NOT}] \text{ IN } ( \langle value \rangle [, \langle value \rangle]^* )$

		$\langle col \rangle$ [NOT] BETWEEN $\langle value \rangle$ AND $\langle value \rangle$
		$\langle col \rangle$ [NOT] LIKE $\langle string \rangle$
		( $\langle boolExpr \rangle$ )
		$\langle boolExpr \rangle$ AND $\langle boolExpr \rangle$
		$\langle boolExpr \rangle$ OR $\langle boolExpr \rangle$
$\langle col \rangle$	::=	[ $\langle tbName \rangle$ .] $\langle colName \rangle$
$\langle boolOp \rangle$	::=	'='   '<>'   '<='   '>='   '<'   '>'
$\langle expr \rangle$	::=	$\langle value \rangle$
		$\langle col \rangle$
		$\langle expr \rangle$ $\langle op \rangle$ $\langle expr \rangle$
		( $\langle expr \rangle$ )
$\langle op \rangle$	::=	'+'   '-'   '*'   '/'   '%'
$\langle selectors \rangle$	::=	*   $\langle selector \rangle$ [, $\langle selector \rangle$ ]*
$\langle selector \rangle$	::=	$\langle col \rangle$
		$\langle func \rangle$ ( $\langle col \rangle$ )
$\langle func \rangle$	::=	SUM   AVG   MAX   MIN
$\langle idxStmt \rangle$	::=	CREATE INDEX $\langle tbName \rangle$ ( $\langle colName \rangle$ )
		DROP INDEX $\langle tbName \rangle$ ( $\langle colName \rangle$ )

### 3.3.3 mydb 运算符优先级

(from lowest to highest)

left **or**

left **and**

left **+ -**

left **\* / %**

## 3.4 SQL 执行模块

在本模块中，我们将在数据库上实际执行上一模块解析出的 SQL 命令。

- 对于 USE, CREATE, DROP 等这种对表结构进行操作的命令，送往系统管理模块执行。
- 对于 INSERT, UPDATE, DELETE, SELECT 等这种涉及记录的命令，将在本模块中通过调用系统管理模块和记录管理模块提供的功能来完成。

我们定义了一个 `Evaluator` 类，可以对包含列名的表达式求值。

下面我们将详细列举各命令、功能的实现原理及方法。

#### 3.4.1 INSERT

我们支持两种格式的 `INSERT`。对于不指定列名的格式，我们根据创建表时给出的列顺序将表数据填入这条记录的相应位置，否则按照给出的列名填入。在填入用户给定的数据前，我们需要初始化 `NULL` 列的内容，以及在记录中填入默认值和自增字段的值。在填入数据时，我们会对列数据格式作兼容性检查。在填入数据后，我们会检查该记录是否满足所有的约束。最后，若一切条件均满足，便调用记录管理模块将该条记录写入数据库，同时更新有关索引。

#### 3.4.2 SELECT

由于 `SELECT` 语句格式比较复杂，我们还支持聚集函数、`GROUP BY` 和多表连接，因此在我们的实现中分四个步骤进行。

**Query** 在此阶段，我们会根据 `WHERE` 子句的语法树生成一个 `filter` 函数对象，并将 `filter` 函数传入记录管理模块，在相应表的记录中将所有可能符合条件的记录找出。此处的可能是指在多表连接查询中，忽略了其余表的条件时的情况。这一阶段也就是 `WHERE` 子句的处理过程。

**Join** 在此阶段，我们会使用递归将上一阶段在所有表中查出的记录做笛卡尔积，同时再次应用 `filter` 函数找出所有符合条件的记录。

**Group By** 在此阶段，我们会根据 `GROUP BY` 子句将上一阶段得到的结果分组。若 `GROUP BY` 为空，则分为 1 组。

**Aggregate** 在此阶段，我们会对上一阶段中得到的所有组中的记录分别应用聚集函数，得到每组中的结果。若聚集函数为空，则返回所有结果。

经过以上四个步骤后，我们便得到了 `SELECT` 语句的结果，最后将其打印出来即可。

#### 3.4.3 UPDATE

首先，我们会根据 `WHERE` 语句查出所有符合条件的需要更新的记录（查询的详细方法见 3.4.2）。在查出需要更新的记录后，由原记录计算 `SET` 子句中表达式的值，然后的处理流程与 3.4.1 `INSERT` 完全类似。



#### 3.4.4 DELETE

对于 DELETE 命令, 同样根据 WHERE 语句查出符合条件的记录, 然后就可以调用记录管理模块删除这些记录以及对应的索引了。

#### 3.4.5 USE, CREATE / DROP DATABASE / TABLE

对于这几个命令, 直接调用系统管理模块完成相应功能。

注意对于 CREATE TABLE 命令, 我们会增加一个隐藏的列 NULL, 以记录每列是否为空。

#### 3.4.6 CREATE / DROP INDEX

对于建立、删除索引命令, 由记录管理模块完成相应操作。

建立索引时, 在记录管理模块中新建一个 multimap, 并将所有记录的相应列值加入 multimap 中。

删除索引时, 直接删除 multimap。

#### 3.4.7 CHECK

对于 CHECK 约束, 在建表时将 BoolExpr 序列化保存下来, 每当插入或更新记录时, 对该表达式求值并判断真值即可。

#### 3.4.8 PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL

PRIMARY KEY 相当于 UNIQUE + NOT NULL。

这四个约束在插入或更新记录时检查。

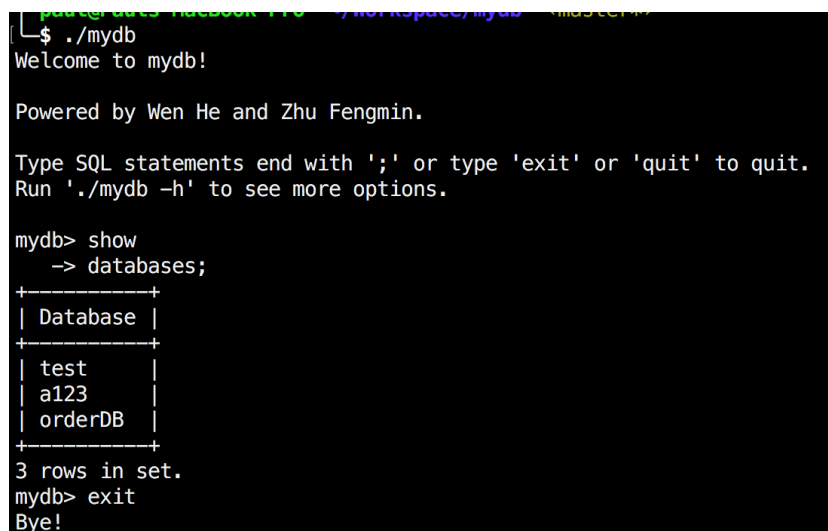
- 对于 NOT NULL, 直接检查 NULL 列中对应位即可。
- 对于 PRIMARY KEY 和 FOREIGN KEY, 使用索引在记录管理模块中寻找该列是否存在相应的值。

#### 3.4.9 WHERE 子句的扩展运算

- 对于 BETWEEN 运算, 我们只支持数值类型 BETWEEN, 直接将其转化为 >= 和 <= 运算执行。
- 对于 IN 运算, 我们直接循环每个值, 判断是否相等。
- 对于 LIKE 运算, 我们将查询字符串转换为正则表达式, 然后调用正则表达式匹配。具体来说, 将 % 转换为 .\*, 将 \_ 转换为 ., 将 [!set] 中的 ! 转换为 ^。

### 3.5 用户交互模块

在顶层，我们设计了一个交互式终端，方便用户键入 SQL 语句并执行数据库操作。输入 `exit` 或 `quit` 可以退出终端。我们还支持多行输入 SQL 语句 (如图 3)，以 `;` 作为语句的结束。



```
└─$ ./mydb
Welcome to mydb!

Powered by Wen He and Zhu Fengmin.

Type SQL statements end with ';' or type 'exit' or 'quit' to quit.
Run './mydb -h' to see more options.

mydb> show
      -> databases;
+-----+
| Database |
+-----+
| test     |
| a123     |
| orderDB  |
+-----+
3 rows in set.
mydb> exit
Bye!
```

图 3: 交互式终端允许多行输入

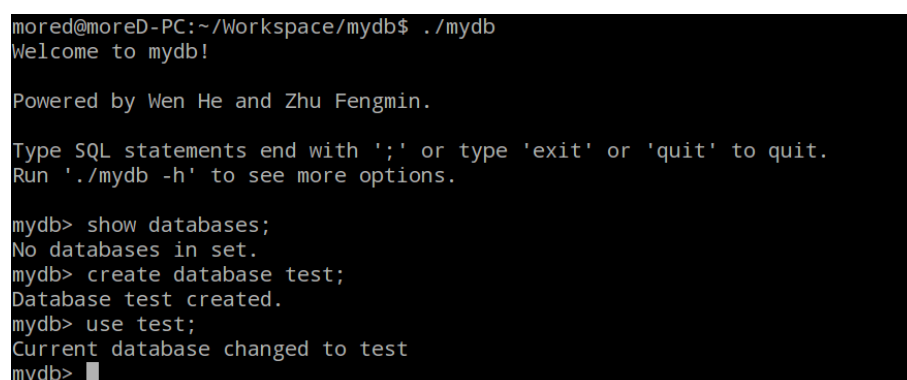
此外，我们还利用 Linux 的 `socket` API 实现了远程数据库连接。具体来说，就是通过 `socket` 把客户端的 SQL 语句发送给服务端，服务端接收到命令后调用 SQL 解析模块，操作完成后，将返回的结果字符串通过 `socket` 回送给客户端，客户端将其显示出来。

## 4 测试结果

接下来我们对 `mydb` 实现的功能一一进行测试。

### 4.1 数据库管理及查询功能

首先，我们打开 `mydb`，新建一个数据库 `test` 并切换到它。



```
mored@moreD-PC:~/Workspace/mydb$ ./mydb
Welcome to mydb!

Powered by Wen He and Zhu Fengmin.

Type SQL statements end with ';' or type 'exit' or 'quit' to quit.
Run './mydb -h' to see more options.

mydb> show databases;
No databases in set.
mydb> create database test;
Database test created.
mydb> use test;
Current database changed to test
mydb> █
```

图 4: 新建数据库并切换

然后，我们新建一个带有主键、自增字段、默认值字段的表，再新建一个有唯一字段、非空字段、域完整性约束和外键的表。建立表结构后，我们往表里添加一些合法记录。

```
mydb> use test;
Current database changed to test
mydb> create table T1 (id int auto_increment, name varchar(16), age int default 18, primary key(id));
Table T1 created.
mydb> create table T2 (cid int unique, tid int, mobile varchar(11) not null, foreign key (tid) references T1(id), check (cid>0));
Table T2 created.
mydb> insert into T1 (name, age) values('moreD', 20), ('paul', 21);
mydb> insert into T1 (name) values('somebody');
mydb> insert into T2 values (1, 1, '13000000000'), (2, 2, '13111111111');
mydb> select * from T1;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20    | 'moreD' | 1     |
| 21    | 'paul'  | 2     |
| 18    | 'somebody' | 3    |
+-----+-----+-----+
3 records found in total.
mydb> select * from T2;
+-----+-----+-----+
| T2.tid | T2.mobile | T2.cid |
+-----+-----+-----+
| 1      | '13000000000' | 1      |
| 2      | '13111111111' | 2      |
+-----+-----+-----+
2 records found in total.
mydb>
```

图 5: 建表并插入记录

我们再尝试往表里添加非法记录。

```
mydb> insert into T1 (id, name, age) values(1, 'sth', 1);
[ERROR] value in column 'id' is not UNIQUE.
mydb> insert into T2 (cid, tid) values(1, 1);
[ERROR] value in column 'cid' is not UNIQUE.
mydb> insert into T2 (cid, tid) values(5, 5);
[ERROR] invalid reference in FOREIGN KEY to 'T1.id'
mydb> insert into T2 (cid, tid) values(5, 2);
[ERROR] Column mobile is NOT NULL.
mydb>
```

图 6: 插入非法记录时报错

然后我们尝试使用各种查询语句。

```

mydb> select * from T1;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'moreD' | 1      |
| 21     | 'paul'  | 2      |
| 18     | 'somebody' | 3      |
+-----+-----+-----+

3 records found in total.
mydb> select * from T1 where id>2;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 18     | 'somebody' | 3      |
+-----+-----+-----+

1 records found in total.
mydb> select * from T1 where age between 18 and 20;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'moreD' | 1      |
| 18     | 'somebody' | 3      |
+-----+-----+-----+

2 records found in total.
mydb> █

```

图 7: 查询 1

```

mydb> select * from T1 where age between 18 and 20;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'moreD' | 1      |
| 18     | 'somebody' | 3      |
+-----+-----+-----+

2 records found in total.
mydb> select * from T1 where name like '%o%';
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'moreD' | 1      |
| 18     | 'somebody' | 3      |
+-----+-----+-----+

2 records found in total.
mydb> select min(age) from T1;
+-----+
| T1.age |
+-----+
| 18     |
+-----+

1 records found in total.
mydb> █

```

图 8: 查询 2

```

mydb> select * from T1 where age between 18 and 20;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'moreD' | 1     |
| 18     | 'somebody' | 3     |
+-----+-----+-----+

2 records found in total.
mydb> select * from T1 where name like '%o%';
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'moreD' | 1     |
| 18     | 'somebody' | 3     |
+-----+-----+-----+

2 records found in total.
mydb> select min(age) from T1;
+-----+
| T1.age |
+-----+
| 18     |
+-----+

1 records found in total.
mydb>

```

图 9: 查询 3

再尝试更新和删除语句。

```

mydb> select * from T1;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'moreD' | 1     |
| 21     | 'paul'   | 2     |
| 18     | 'somebody' | 3     |
+-----+-----+-----+

3 records found in total.
mydb> update T1 set name='mored' where name='moreD';
mydb> select * from T1;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'mored' | 1     |
| 21     | 'paul'   | 2     |
| 18     | 'somebody' | 3     |
+-----+-----+-----+

3 records found in total.
mydb> delete from T1 where age<19;
mydb> select * from T1;
+-----+-----+-----+
| T1.age | T1.name | T1.id |
+-----+-----+-----+
| 20     | 'mored' | 1     |
| 21     | 'paul'   | 2     |
+-----+-----+-----+

2 records found in total.
mydb>

```

图 10: 更新和删除

最后删除表和数据库。

```

mydb> show tables;
+-----+
| Tables in test |
+-----+
| T1             |
| T2             |
+-----+
2 rows in set.
mydb> drop table T2;
Table T2 has been dropped.
mydb> show tables;
+-----+
| Tables in test |
+-----+
| T1             |
+-----+
1 rows in set.
mydb> show databases;
+-----+
| Database |
+-----+
| test     |
+-----+
1 rows in set.
mydb> drop database test;
Database test has been dropped.
mydb> show databases;
No databases in set.
mydb>

```

图 11: 删除表及数据库

## 4.2 远程连接功能

首先，我们启动一个服务端。

```

$ ./mydb -s 8010
[SOCKET] Server started at port 8010.

```

图 12: 启动服务端

接下来，我们通过客户端连接服务端。

```

$ ./mydb -r 127.0.0.1 -p 8010
[SOCKET] Connecting 127.0.0.1:8010...
[SOCKET] OK.
Welcome to mydb!

Powered by Wen He and Zhu Fengmin.

Type SQL statements end with ';' or type 'exit' or 'quit' to quit.
Run './mydb -h' to see more options.

mydb>

```

图 13: 客户端连接服务端

最后，我们输入一系列 SQL 语句，测试服务端返回的信息都是正确的。

```

mydb> create database test;
[ERROR] database test already exists, skipping...
mydb> create database test1;
Database test1 created.
mydb> use test1;
[ERROR] database test1 not exists.
mydb> use test1;
Current database changed to test1
mydb> create table t1 (id int, info varchar(255));
Table t1 created.
mydb> insert into t1 values (1, 'info1'), (2, 'info2');
OK.
mydb> select * from t1;
+-----+-----+
| t1.info | t1.id |
+-----+-----+
| 'info1' | 1     |
| 'info2' | 2     |
+-----+-----+

2 records found in total.
mydb> update t1 set id=10 where info like '%info%';
OK.
mydb> select * from t1;
+-----+-----+
| t1.info | t1.id |
+-----+-----+
| 'info1' | 10    |
| 'info2' | 10    |
+-----+-----+

2 records found in total.
mydb> qui
-> ;
[PARSER] at :1.1-3: syntax error.
mydb> quit
Bye!

```

图 14: 远程操作数据库

退出客户端后，服务端仍然等待新的客户端连接。

```

$ ./mydb -s 8010
[SOCKET] Server started at port 8010.
[SOCKET] Service started for sock 4
.[SOCKET] Server stopped for sock 4

```

图 15: 断开连接

## 5 项目链接

<https://github.com/paulzfm/mydb>

## 6 使用方法

详见 README.md，或前往 Github: <https://github.com/paulzfm/mydb#mydb>。

## 7 小组分工

请见表 4。

表 4: 小组分工

功能	负责人
记录管理	朱俸民
系统管理	温和
SQL 解析与语义检查	朱俸民
查询执行	温和
交互终端与远程连接	朱俸民
索引	温和

## 参考资料

- [1] Bison. <https://engineering.purdue.edu/~milind/ece573/2015fall/project/bison.html>.
- [2] SQL Tutorial. <http://www.w3schools.com/sql/default.asp>.
- [3] 冯建华, 周立柱. 数据库系统设计与原理.
- [4] cppreference. <http://en.cppreference.com/>.
- [5] RapidJSON. <http://rapidjson.org/index.html>.