WickyDB 总体设计报告

1 Wicky DB 系统概述

1.1 背景

1.1.1 编写目的

本项目名称为 WickyDB，它是一个精简型单用户 SQL 引擎(DBMS)MiniSQL，允许用户通过字符界面输入 SQL 语句实现表的建立/删除；索引的建立/删除以及表记录的插入/删除/查找等功能。

1.1.2 项目背景

本项目是《数据库系统设计》课程等期末大程。在设计和实现 WickyDB 过程中，我们加深了对数据库系统中 table、index、record、catalog、buffer 等模块等理解，并且对各个模块之间的调用与合作有了更加清晰的认识。

1.2 功能描述

WickyDB 实现了实验要求中的所有功能，包括表的创建与删除、索引的建立与删除、记录的插入与删除、查询功能、SQL 脚本的执行，此外，我们还加入了 show table 和 describe table 功能。以 student 表为例，具体描述如下：

①表的创建与删除：

```
create table student (
        sno int,
        sname char(16) unique,
        sage int,
        sgender char (1),
        score float,
        primary key ( sno )
);
drop table student;
```

②索引的创建与删除

```
create index stunameidx on student ( sname );
drop index stunameidx on student;
```

③记录的插入与删除

```
insert into student values (201,'wy201',21,'F',84.0);
```

若该语句执行成功，则输出执行成功信息；若失败，则告诉用户失败的原因。

```
delete from student;
delete from student where sno = '100';
delete from student where sno>=100;
```

若该语句执行成功，则输出执行成功信息，其中包括删除的记录数；若失败，则告诉用户失败的原因。

④查询功能

```
select * from student;
select * from student where sno =104 ;
select * from student where score >= 90 and score <=95;
select * from student where sgender<>'M';
```

若该语句执行成功且查询结果不为空，则按行输出查询结果，第一行为属性名，其余每一行表示一条记录；若失败，则告诉用户失败的原因。

⑤SQL 脚本的执行

```
execfile    insert_student.sql;
```

SQL 脚本文件中可以包含任意多条 SQL 语句，WickyDB 读入该文件，然后按序依次逐条执行脚本中的 SQL 语句。

⑥其他功能

show table;	显示 WickyDB 中所有存在的表名称
desc table student;	描述 student 表所有 attribute 名称和类型
exit；	退出 WickyDB

1.3 运行环境和配置

WickyDB 的运行环境为 Windows 和 Mac OS，使用 C++语言编写，我们采用 make 工具来编译和运行。

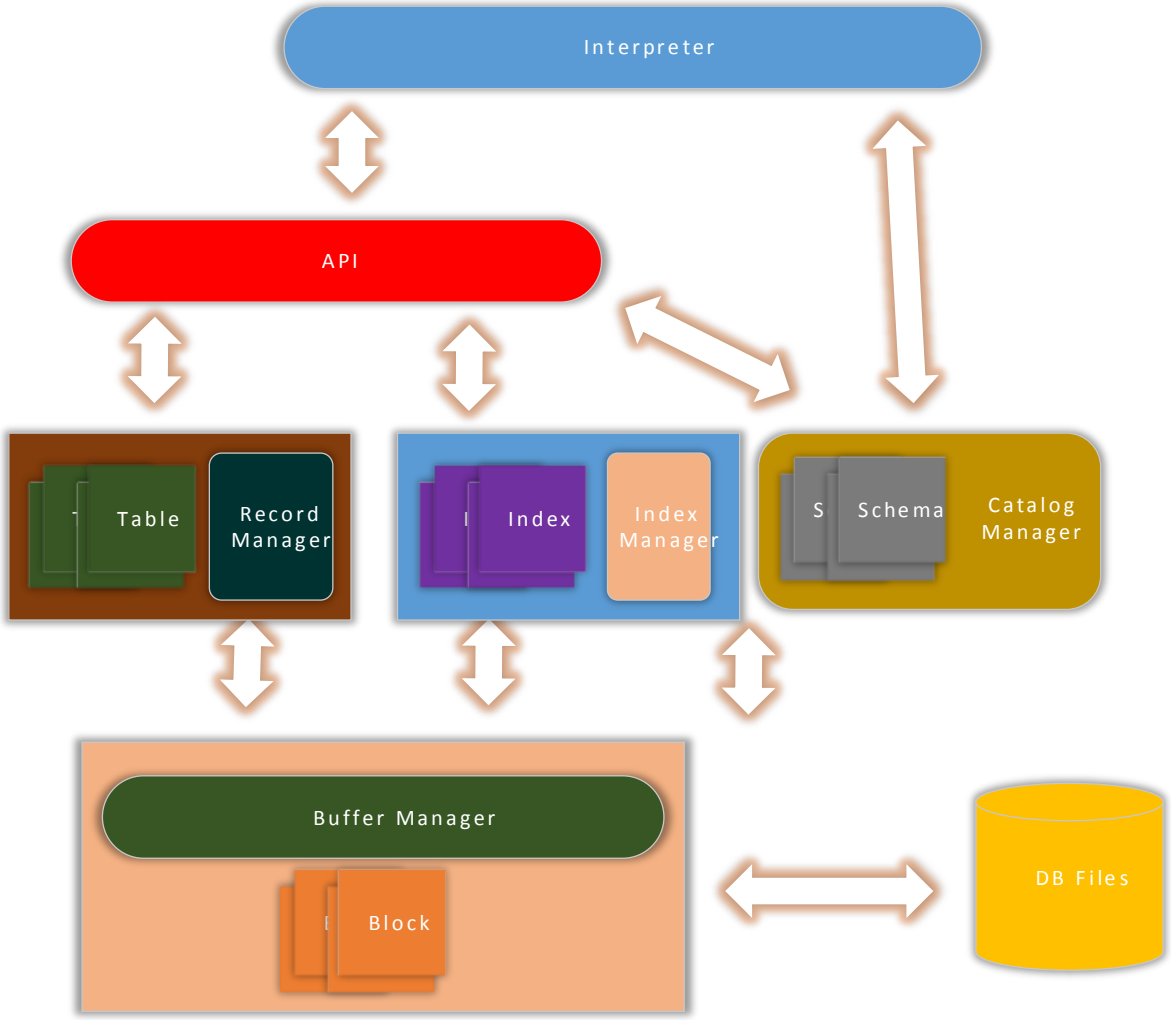
1.4 参考资料

Database System Concepts（Six Edition）

2 WickyDB 系统结构设计

2.1 总体设计：

按照实验要求，WickyDB 的主体被分为 Interpreter、API、RecordManager、IndexManager、CatalogManager 以及 BufferManager 几个部分。下图是我们设计的几个主要对象及其相互关系，能够比较具象地描绘我们的设计意图。



Interpreter 使用 Yacc 和 Lex，产生友好的用户交互界面，使得用户能够很方便地使用 WickyDB。API 与 Interpreter 紧密相连，所有的 SQL 语句都能在 API 层找到相应的 API，并进行执行。RecordManager 负责管理数据的插入、检索与删除，是 WickyDB 的重要模块。IndexManager 能够创建与删除索引，为 API 层提供可供使用的索引。CatalogManager 负责处理整个系统的所有元数据，包括表的数量、属性、索引的属性、所属等等。从 CatalogManager 可以获取 Schema 以完成对表与索引的初始化。最下层的 BufferManager 直接与文件交流，对上也提供类似文件的接口，基本上透明的设计，用户可以很方便地使用 BufferManager 而感觉不到 Block 的存在。

2.2 Interpreter 模块：

Interpreter 模块直接与用户交互，主要实现以下功能：
1. 程序流程控制，即“启动并初始化 ‘接收命令、处理命令、显示命令结果’ 循环 退出”流程。
2. 接收并解释用户输入的命令，生成命令的内部数据结构表示，同时检查命令的语法正确性和语义正确性，对正确的命令调用 API 层提供的函数执行并显示执行结果，对不正确的命令显示错误信息。

具体实现中，整个 Interpreter 被一个上层的对象封装起来，通过进一步的封装，Lex 和 Yacc 可以很好地被嵌入到我们系统的体系中来。
前端的所有词语都在 Lex 中进行分析，而后被传到 Yacc 进行语法分析，Yacc 的语法分析成功后将调用相应的 API 进行处理。

2.3 API 模块：

API 模块是整个系统的核心，其主要功能为提供执行 SQL 语句的接口，供 Interpreter 层调用。该接口以 Interpreter 层解释生成的命令内部表示为输入，根据 Catalog Manager 提供的信息确定执行规则，并调用 Record Manager、Index Manager 和 Catalog Manager 提供的相应接口进行执行，最后返回执行结果给 Interpreter 模块。

2.4 CatalogManager 模块：

Catalog Manager 负责管理数据库的所有模式信息，包括：
1. 数据库中所有表的定义信息，包括表的名称、表中字段（列）数、主键、定义在该表上的索引。
2. 表中每个字段的定义信息，包括字段类型、是否唯一等。
3. 数据库中所有索引的定义，包括所属表、索引建立在那个字段上等。

Catalog Manager 还提供了访问及操作上述信息的接口，供 Interpreter 和 API 模块使用。

具体实现中为了便于对表的操作以及实现良好的封装性，我定义了`Attribute`、`Schema`和`Catalog Manager` 三个类来管理数据库中的表的信息。其中，Attribute 定义了表中每个字段的信息，包括字段名、类型、长度、是否唯一等。Schema 定义了单个表的所有信息，包括表名、字段列表、主键、索引列表等，对单个表的修改操作都是在 Schama 类中实现的。而对整张表的创建、删除、读取、保存等操作则要通过 CatalogManager 类来完成。

2.5 RecordManager 模块：

Record Manager 负责管理记录表中数据的数据文件。主要功能为实现数据文件的创建与删除（由表的定义与删除引起）、记录的插入、删除与查找操作，并对外

提供相应的接口。其中记录的查找操作能够支持不带条件的查找和带一个条件的查找（包括等值查找、不等值查找和区间查找）。
数据文件由一个或多个数据块组成，块大小应与缓冲区块大小相同。一个块中包含一条至多条记录，为简单起见，只要求支持定长记录的存储，且不要求支持记录的跨块存储。

2.6 Indexmanager 模块：

Index Manager 负责 B+树索引的实现，实现 B+树的创建和删除（由索引的定义与删除引起）、等值查找、插入键值、删除键值等操作，并对外提供相应的接口。B+树中节点大小与缓冲区的块大小相同，B+树的叉数由节点大小与索引键大小计算得到。

具体实现中，使用了一个动态的回收机制来管理整个磁盘空间。设计上是每个文件的第一个块记录关于当前索引的空间分配信息。包括根节点位置、索引空间范围以及所有未分配的空间。用于管理整个索引的空间分配。以整个索引的布局由两个部分组成，管理空间信息的首块以及存储具体索引信息的节点，这些内容共同组成了整个索引的数据库文件。这种布局的索引文件结构能够尽可能地利用磁盘空间，在封装之后能够非常方便地申请与释放磁盘空间。

2.7 BufferManager 模块：

Buffer Manager 负责缓冲区的管理，主要功能有：

1. 根据需要，读取指定的数据到系统缓冲区或将缓冲区中的数据写出到文件
2. 实现缓冲区的替换算法，当缓冲区满时选择合适的页进行替换
3. 记录缓冲区中各页的状态，如是否被修改过等
4. 提供缓冲区页的 pin 功能，及锁定缓冲区的页，不允许替换出去

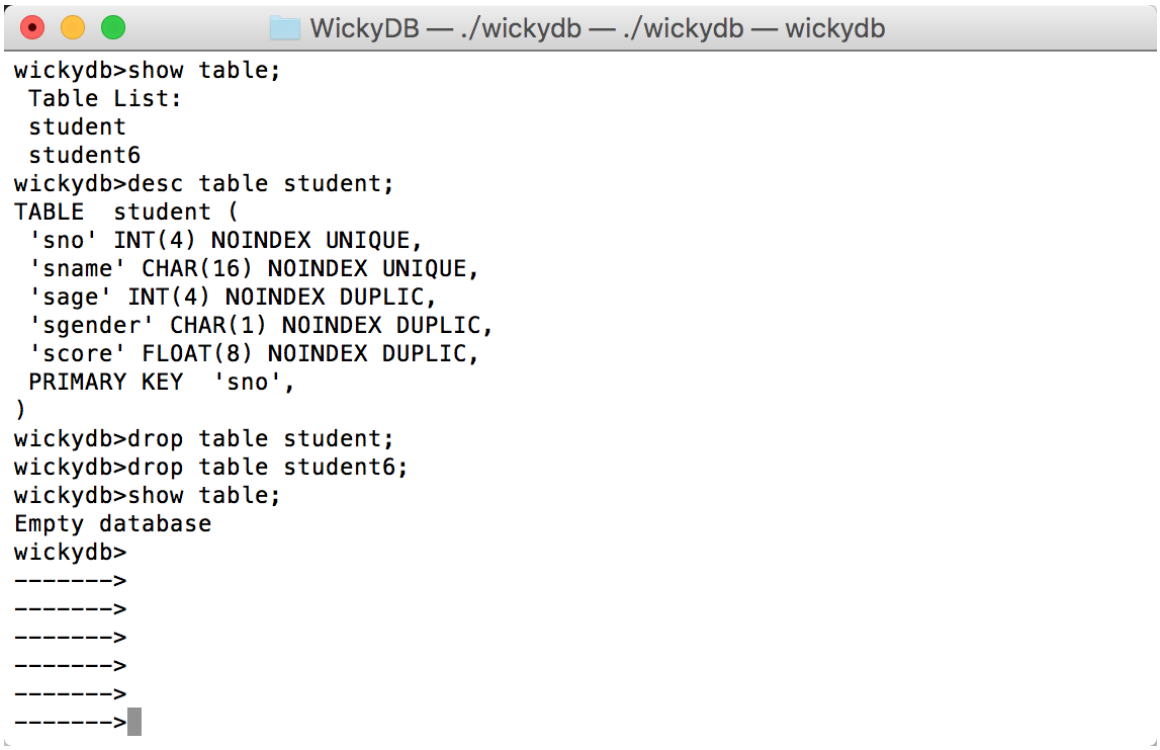
为提高磁盘 I/O 操作的效率，缓冲区与文件系统交互的单位是块，块的大小为文件系统与磁盘交互单位的整数倍，定为 4KB 或 8KB。

3 测试方案和测试样例

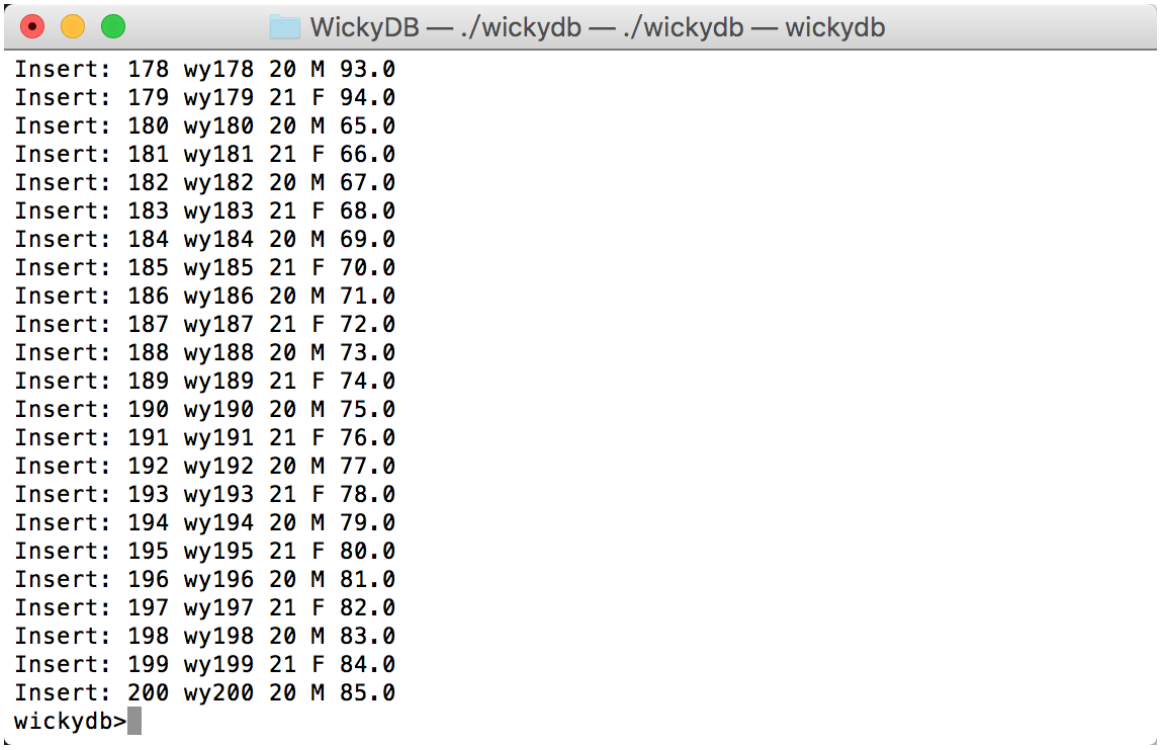
我们的测试展示按照以下指令进行：

```
show table;
desc table student;
drop table student;
drop table student6;
show table;
execfile small.sql

select * from student;
insert into student values (201,'wy201',21,'F',84.0);
insert into student values (202,'wy202',20,'M',85.0);
select * from student;
insert into student values (201,'wy201',21,'F',84.0);

create index stunameidx on student ( sname );

select * from student where sno =104 ;
select * from student where score >= 90 and score <=95;
select * from student where sage > 20 and sgender = 'F';

delete from student where sno = '100';
delete from student where sname = 'wy2';
select * from student;

select * from student where sgender <> 'F';

drop index stunameidx on student;

execfile large.sql
create index stunameidx on student6 ( sname );
select * from student6;
delete from student6 where sno = 200000;
insert into student6 values (19,'ww19',21,'F',82.0);
select * from student6;
delete from student6;
select * from student6;
drop index stunameidx on student6;
```

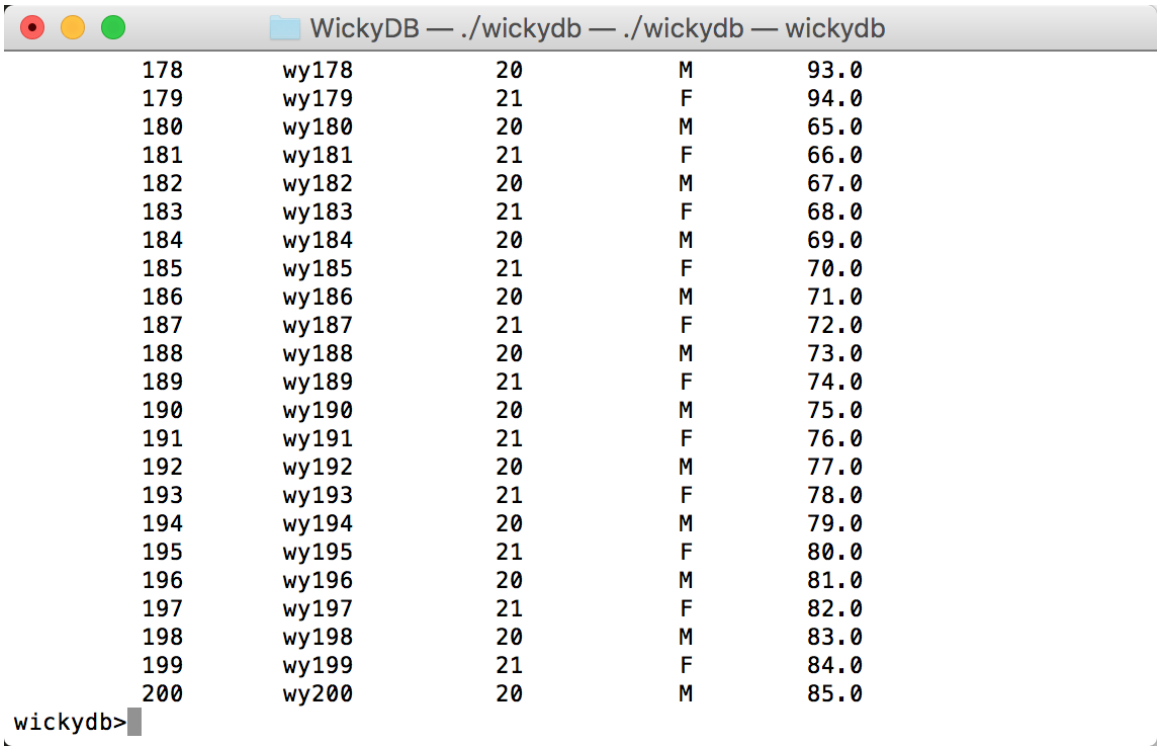
接下来按照上面的流程依次进行操作：

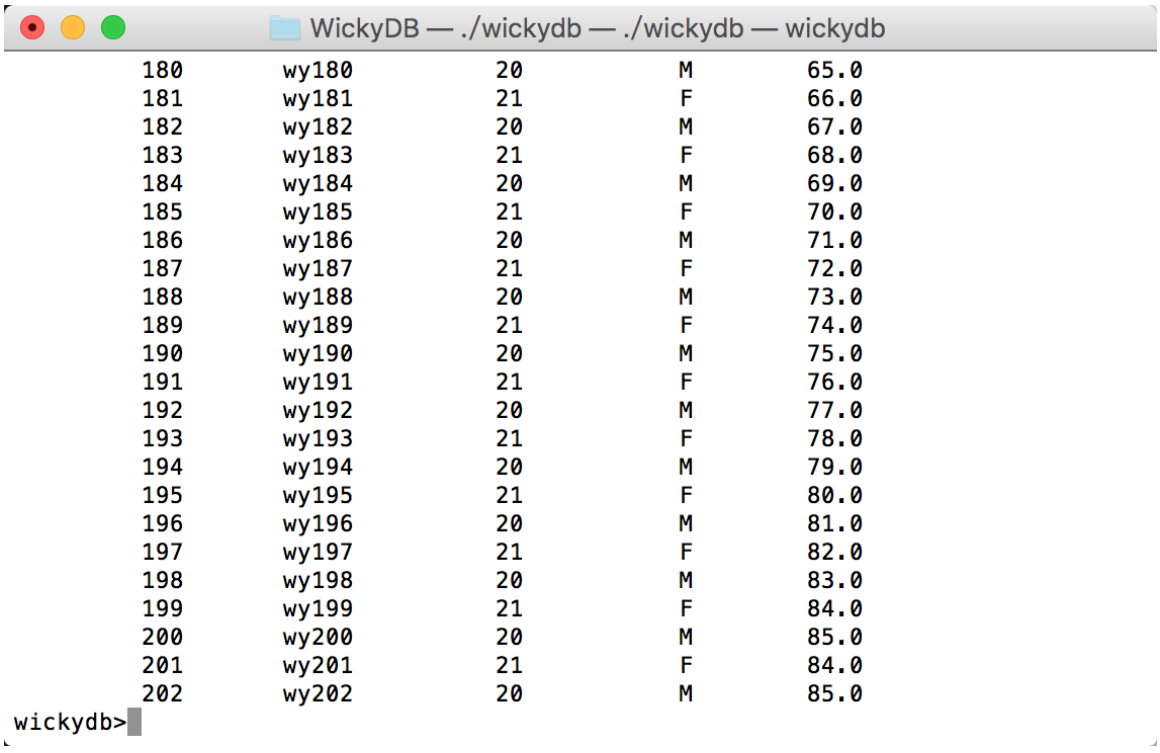


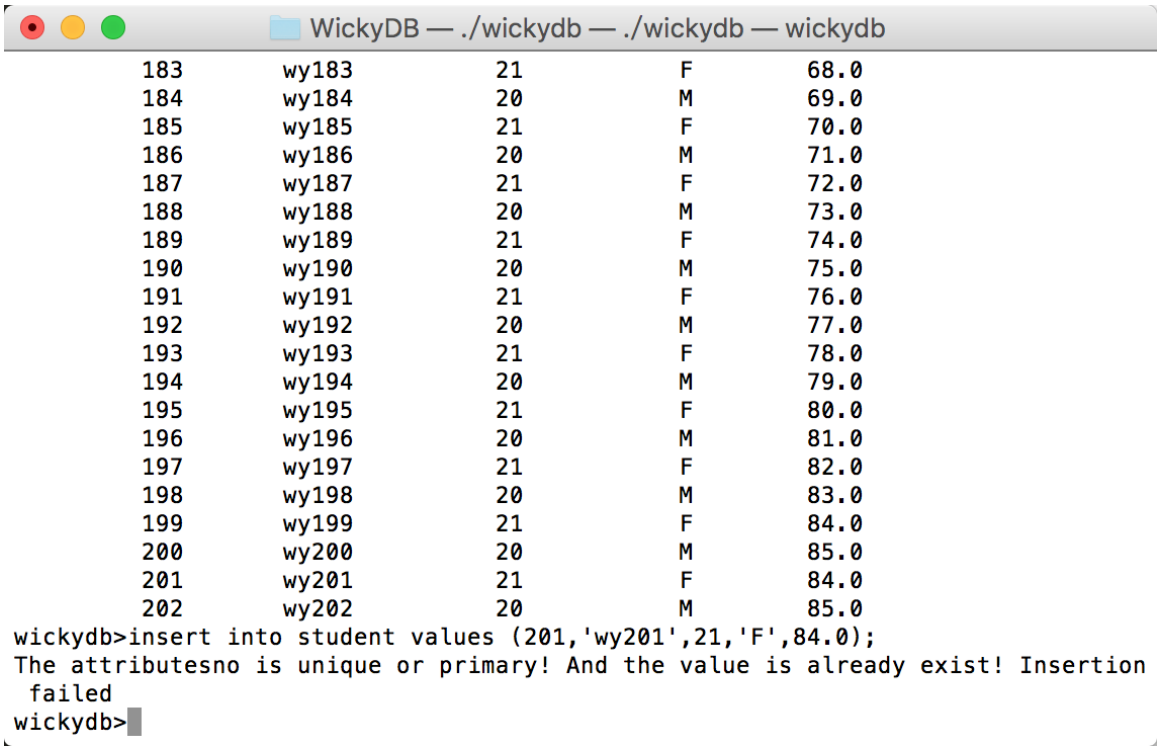
执行 execfile small.sql：

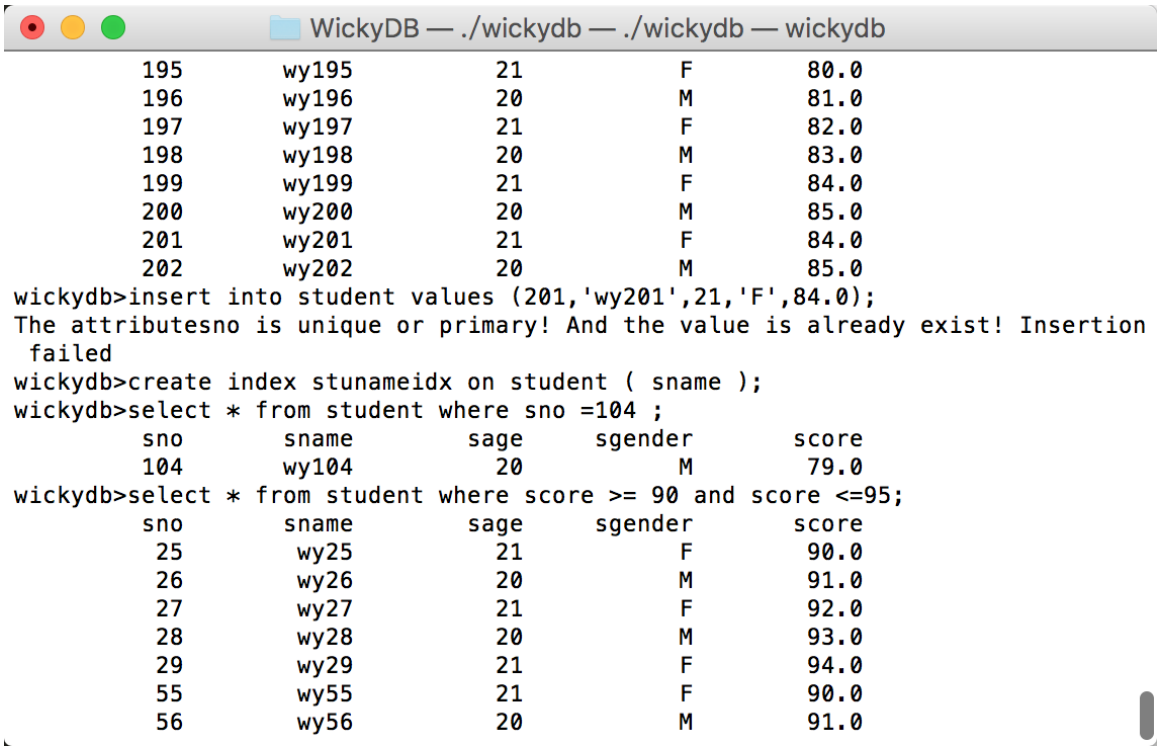


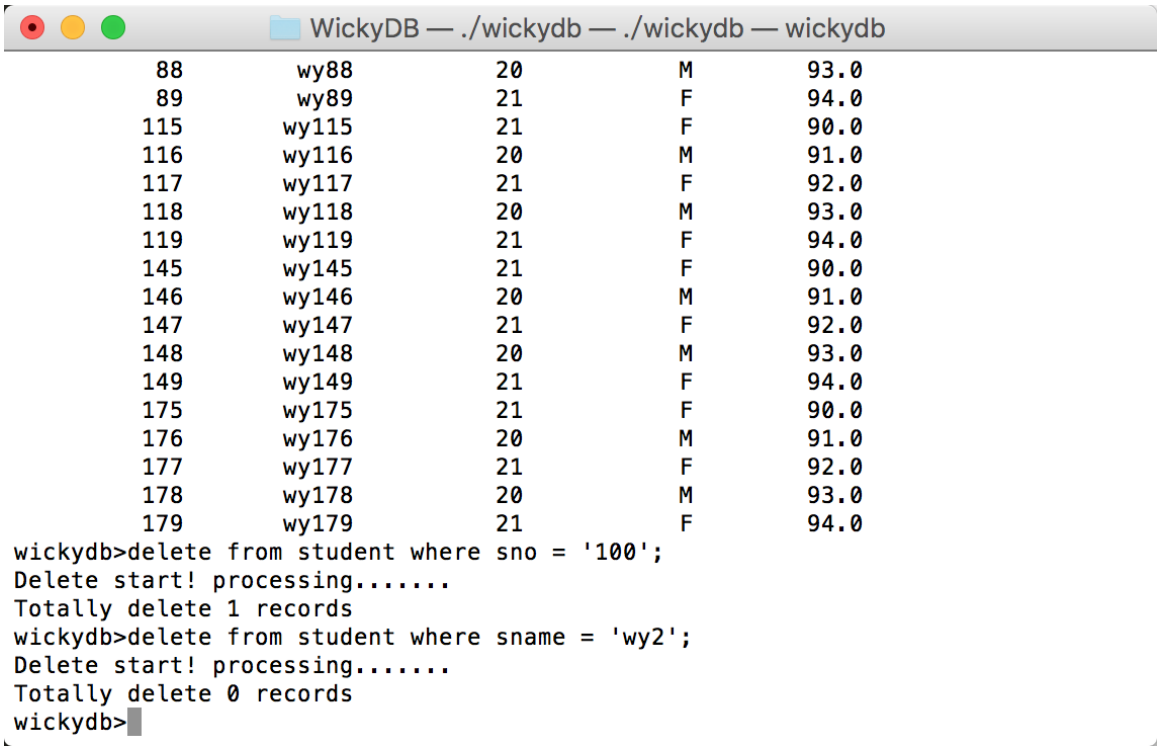
select * from student:

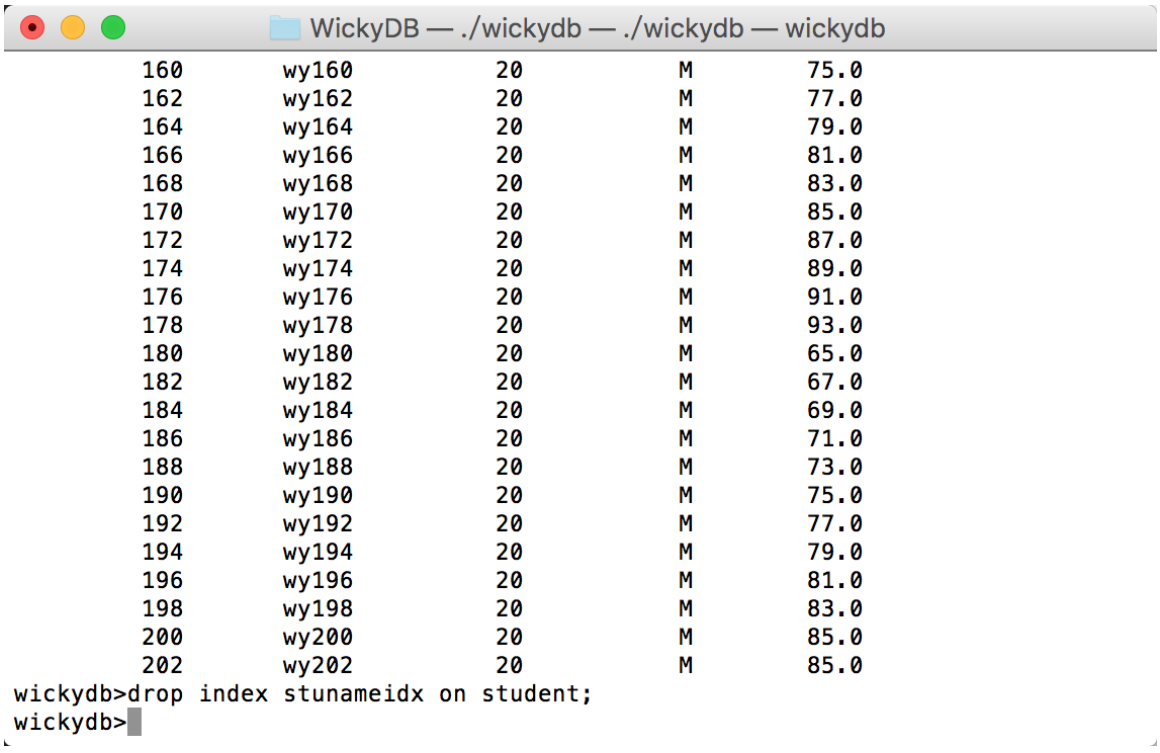

WickyDB — ./wickydb — ./wickydb — wickydb
178 wy178 20 M 93.0
179 wy179 21 F 94.0
180 wy180 20 M 65.0
181 wy181 21 F 66.0
182 wy182 20 M 67.0
183 wy183 21 F 68.0
184 wy184 20 M 69.0
185 wy185 21 F 70.0
186 wy186 20 M 71.0
187 wy187 21 F 72.0
188 wy188 20 M 73.0
189 wy189 21 F 74.0
190 wy190 20 M 75.0
191 wy191 21 F 76.0
192 wy192 20 M 77.0
193 wy193 21 F 78.0
194 wy194 20 M 79.0
195 wy195 21 F 80.0
196 wy196 20 M 81.0
197 wy197 21 F 82.0
198 wy198 20 M 83.0
199 wy199 21 F 84.0
200 wy200 20 M 85.0
wickydb>

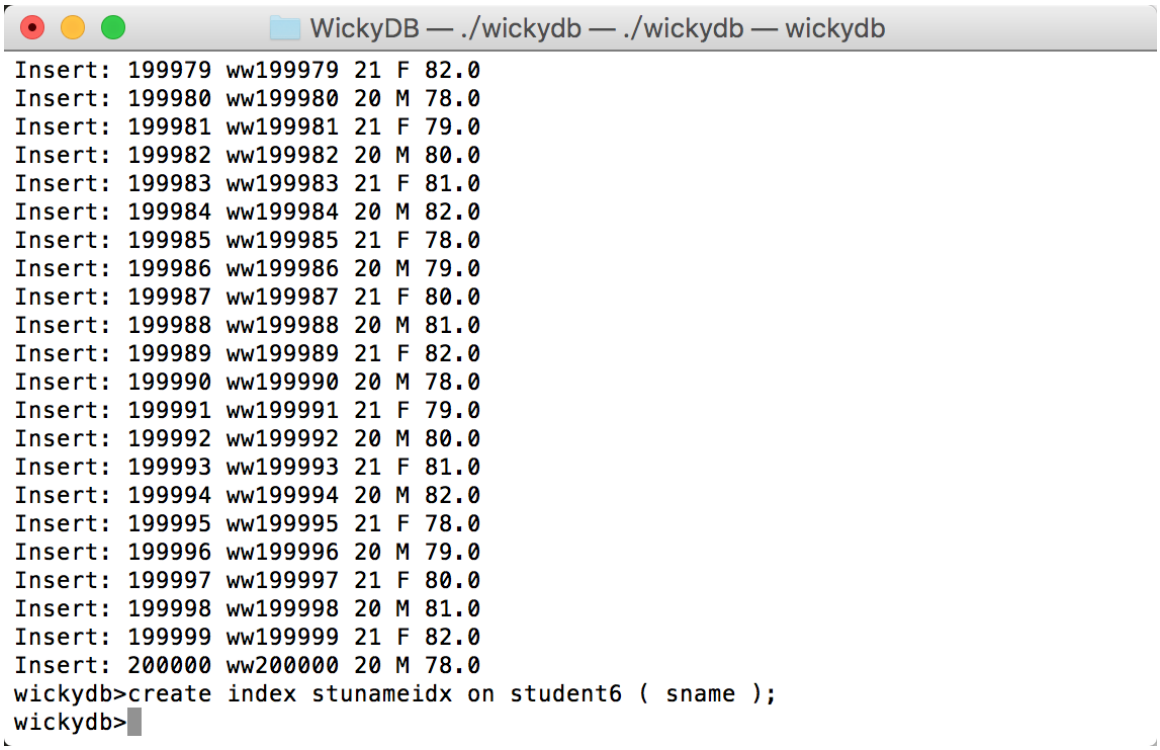

WickyDB — ./wickydb — ./wickydb — wickydb
180 wy180 20 M 65.0
181 wy181 21 F 66.0
182 wy182 20 M 67.0
183 wy183 21 F 68.0
184 wy184 20 M 69.0
185 wy185 21 F 70.0
186 wy186 20 M 71.0
187 wy187 21 F 72.0
188 wy188 20 M 73.0
189 wy189 21 F 74.0
190 wy190 20 M 75.0
191 wy191 21 F 76.0
192 wy192 20 M 77.0
193 wy193 21 F 78.0
194 wy194 20 M 79.0
195 wy195 21 F 80.0
196 wy196 20 M 81.0
197 wy197 21 F 82.0
198 wy198 20 M 83.0
199 wy199 21 F 84.0
200 wy200 20 M 85.0
201 wy201 21 F 84.0
202 wy202 20 M 85.0
wickydb>

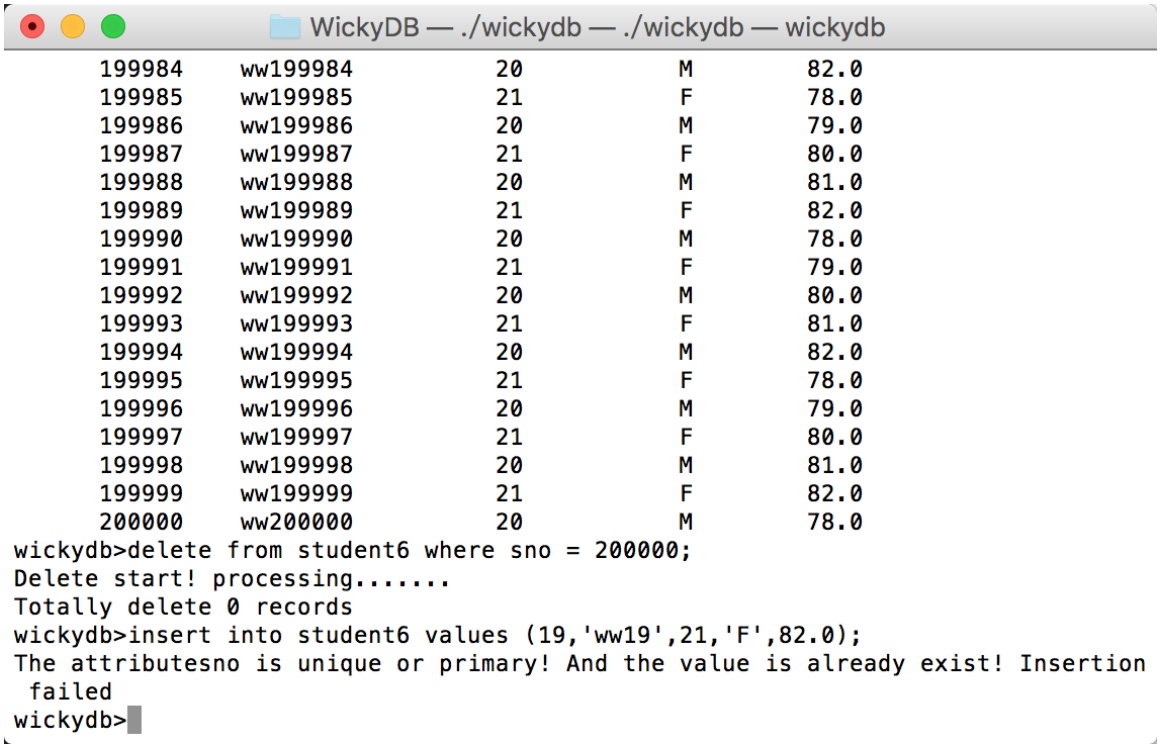

WickyDB — ./wickydb — ./wickydb — wickydb
183 wy183 21 F 68.0
184 wy184 20 M 69.0
185 wy185 21 F 70.0
186 wy186 20 M 71.0
187 wy187 21 F 72.0
188 wy188 20 M 73.0
189 wy189 21 F 74.0
190 wy190 20 M 75.0
191 wy191 21 F 76.0
192 wy192 20 M 77.0
193 wy193 21 F 78.0
194 wy194 20 M 79.0
195 wy195 21 F 80.0
196 wy196 20 M 81.0
197 wy197 21 F 82.0
198 wy198 20 M 83.0
199 wy199 21 F 84.0
200 wy200 20 M 85.0
201 wy201 21 F 84.0
202 wy202 20 M 85.0
wickydb>insert into student values (201,'wy201',21,'F',84.0);
The attributesno is unique or primary! And the value is already exist! Insertion failed
wickydb>

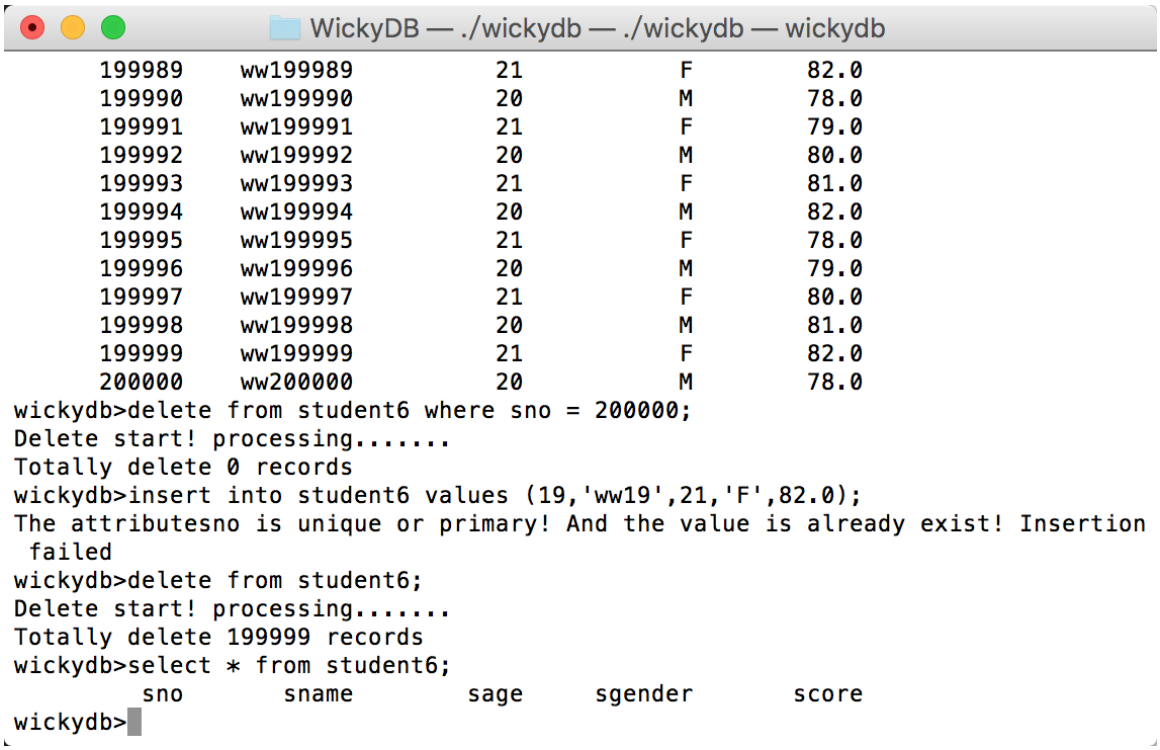

WickyDB — ./wickydb — ./wickydb — wickydb
195 wy195 21 F 80.0
196 wy196 20 M 81.0
197 wy197 21 F 82.0
198 wy198 20 M 83.0
199 wy199 21 F 84.0
200 wy200 20 M 85.0
201 wy201 21 F 84.0
202 wy202 20 M 85.0
wickydb>insert into student values (201,'wy201',21,'F',84.0);
The attributesno is unique or primary! And the value is already exist! Insertion failed
wickydb>create index stunameidx on student (sname);
wickydb>select * from student where sno =104 ;
sno sname sage sgender score
104 wy104 20 M 79.0
wickydb>select * from student where score >= 90 and score <=95;
sno sname sage sgender score
25 wy25 21 F 90.0
26 wy26 20 M 91.0
27 wy27 21 F 92.0
28 wy28 20 M 93.0
29 wy29 21 F 94.0
55 wy55 21 F 90.0
56 wy56 20 M 91.0


WickyDB — ./wickydb — ./wickydb — wickydb
88 wy88 20 M 93.0
89 wy89 21 F 94.0
115 wy115 21 F 90.0
116 wy116 20 M 91.0
117 wy117 21 F 92.0
118 wy118 20 M 93.0
119 wy119 21 F 94.0
145 wy145 21 F 90.0
146 wy146 20 M 91.0
147 wy147 21 F 92.0
148 wy148 20 M 93.0
149 wy149 21 F 94.0
175 wy175 21 F 90.0
176 wy176 20 M 91.0
177 wy177 21 F 92.0
178 wy178 20 M 93.0
179 wy179 21 F 94.0
wickydb>delete from student where sno = '100';
Delete start! processing.......
Totally delete 1 records
wickydb>delete from student where sname = 'wy2';
Delete start! processing.......
Totally delete 0 records
wickydb>


WickyDB — ./wickydb — ./wickydb — wickydb
160 wy160 20 M 75.0
162 wy162 20 M 77.0
164 wy164 20 M 79.0
166 wy166 20 M 81.0
168 wy168 20 M 83.0
170 wy170 20 M 85.0
172 wy172 20 M 87.0
174 wy174 20 M 89.0
176 wy176 20 M 91.0
178 wy178 20 M 93.0
180 wy180 20 M 65.0
182 wy182 20 M 67.0
184 wy184 20 M 69.0
186 wy186 20 M 71.0
188 wy188 20 M 73.0
190 wy190 20 M 75.0
192 wy192 20 M 77.0
194 wy194 20 M 79.0
196 wy196 20 M 81.0
198 wy198 20 M 83.0
200 wy200 20 M 85.0
202 wy202 20 M 85.0
wickydb>drop index stunameidx on student;
wickydb>


WickyDB — ./wickydb — ./wickydb — wickydb
Insert: 199979 ww199979 21 F 82.0
Insert: 199980 ww199980 20 M 78.0
Insert: 199981 ww199981 21 F 79.0
Insert: 199982 ww199982 20 M 80.0
Insert: 199983 ww199983 21 F 81.0
Insert: 199984 ww199984 20 M 82.0
Insert: 199985 ww199985 21 F 78.0
Insert: 199986 ww199986 20 M 79.0
Insert: 199987 ww199987 21 F 80.0
Insert: 199988 ww199988 20 M 81.0
Insert: 199989 ww199989 21 F 82.0
Insert: 199990 ww199990 20 M 78.0
Insert: 199991 ww199991 21 F 79.0
Insert: 199992 ww199992 20 M 80.0
Insert: 199993 ww199993 21 F 81.0
Insert: 199994 ww199994 20 M 82.0
Insert: 199995 ww199995 21 F 78.0
Insert: 199996 ww199996 20 M 79.0
Insert: 199997 ww199997 21 F 80.0
Insert: 199998 ww199998 20 M 81.0
Insert: 199999 ww199999 21 F 82.0
Insert: 200000 ww200000 20 M 78.0
wickydb>create index stunameidx on student6 (sname);
wickydb>


WickyDB — ./wickydb — ./wickydb — wickydb
199984 ww199984 20 M 82.0
199985 ww199985 21 F 78.0
199986 ww199986 20 M 79.0
199987 ww199987 21 F 80.0
199988 ww199988 20 M 81.0
199989 ww199989 21 F 82.0
199990 ww199990 20 M 78.0
199991 ww199991 21 F 79.0
199992 ww199992 20 M 80.0
199993 ww199993 21 F 81.0
199994 ww199994 20 M 82.0
199995 ww199995 21 F 78.0
199996 ww199996 20 M 79.0
199997 ww199997 21 F 80.0
199998 ww199998 20 M 81.0
199999 ww199999 21 F 82.0
200000 ww200000 20 M 78.0
wickydb>delete from student6 where sno = 200000;
Delete start! processing.......
Totally delete 0 records
wickydb>insert into student6 values (19,'ww19',21,'F',82.0);
The attributesno is unique or primary! And the value is already exist! Insertion
failed
wickydb>



4 分组与设计分工

本组成员：
海杰文（组长）3130000656
余秋滨 3130102203
章海威 3130102326
肖邵斌 3130104016

本系统的分工如下：

海杰文：

负责整体框架的设计、各个接口的决定以及 Interpreter 以及 IndexManager 的程序编写。其中 Interpreter 使用 Lex 与 Yacc 编写，在具体报告中将给出详细的语法分析树以及如和 API 层相连接。而 IndexManager 仅负责制造 Index，具体的检索功能在 Index 中实现。

余秋滨：

负责 RecordManager 的设计与代码编写。并且编写与数据操作有关的 API 层代码。

章海威：

负责 CatalogManager 的设计与代码编写。并且编写与元数据操作有关的 API 层代

码。

肖邵斌：

负责 BufferManager 的设计、代码编写以及测试工作。BufferManager 对上提供透明的读写操作接口，用户感觉不到 Block 的存在，BufferManager 可以很好地管理 Block，从而起到 Cache 的效果。

5 总结与展望

本小组最终全部完成了 MiniSQL 的全部要求，但是我们对整个系统的功能仍有许多不满意的地方。从 Interpreter 和 API 方面，我们已经设计好了 Join、Projection 以及 Update 等操作的接口，而且从这个系统设计之初就已经在考虑 select 语句的嵌套等操作。而其他各个更基础的部分，都有各种各样的可供优化的空间。比如我们可以让 Index 支持数据的范围检索，实现更为精巧的 Block 替换算法，从各个方面来提高数据库的性能。这次实验是一次非常好的体验，让我们加深了对数据系统的理解，同样也让我们体会到了团队合作的重要性。