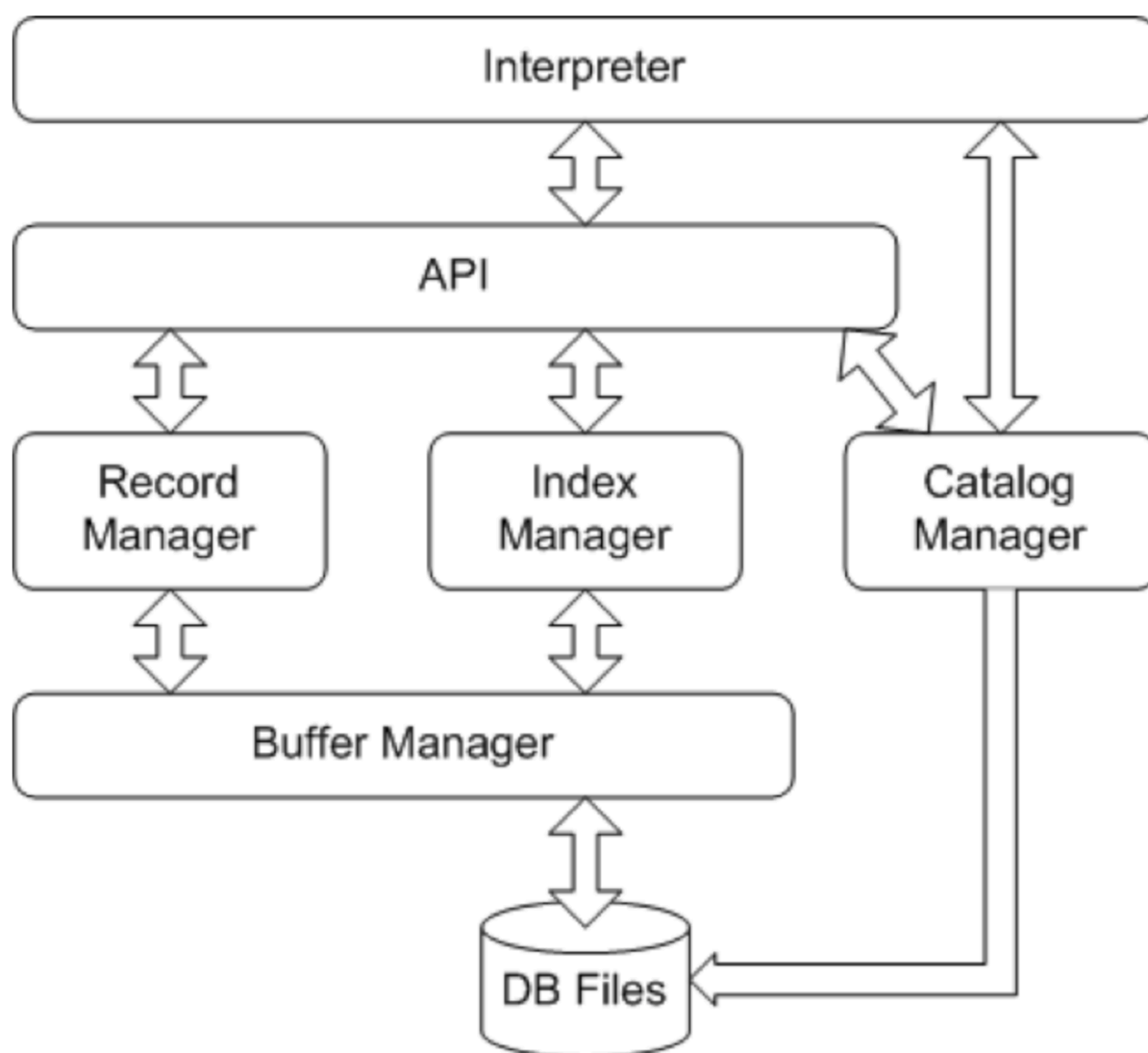


核心模块设计



Interpreter

解释和执行sql语句，格式化sql语句并且生成SQL statements对象，并且把这个对象传递给API中的合适的函数，如果遇到语法错误，将会抛出异常。主要有以下的类：

- Interpreter: 解释SQL语句的类;
- TKey: 在SQL语句中存储值的统一的类;
- SQL: 基类;
- SQLCreateDatabase: 创建database;
- SQLDropDatabase: 删除database;
- SQLDropTable: 删除表;
- SQLDropIndex: 删除索引;
- SQLUse: 使用sql;
- SQLCreateTable: 创建表;
- SQLInsert: 插入sql;

- SQLExec:执行sql;
- SQLSelect: 查询;
- SQLCreateIndex: 创建索引;
- SQLDelete:删除值;
- SQLUpdate:更新值;

API

提供了Record Manager, Index Manager and Catalog Manager的接口, 方便interpreter使用。

Record Manager

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要求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。

Index Manager

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+树中节点大小应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。

- IndexManager
- BPlusTree: B+ tree
- BPlusTreeNode: B+ tree node

Catalog Manager

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还必需提供访问及操作上述信息的接口, 供Interpreter和API模块使用。

- CatalogManager
- Database
- Table
- Attribute
- Index

Buffer Manager

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，块的大小应为文件系统与磁盘交互单位的整数倍，一般可定为4KB或8KB。

记录管理模块（RecordManager）和索引管理模块（IndexManager）向缓冲区管理申请所要的数据，缓冲区管理器首先在缓冲区中查看数据是否存在，若存在，直接返回，否则，从磁盘中将数据读入缓冲区，然后返回。

最近最少使用(LRU)算法：每次访问一个块的时候，如果是重新读取的话把该块的iTime置为1，如果是已经有的话把它的使用次数加一。采用这种方式记录主要是因为对于Index部分，比如根节点的使用率一直保持非常高。

换出脏块时需要写出块。

测试

1. 创建数据库，create database haojs;

```
MiniDB> create database haojs;
SQL TYPE: #CREATE DATABASE#
SQL STATEMENT: create database haojs
DB NAME: haojs
Creating database: haojs
Database folder created!
Catalog written!
```

```
MiniDB> |
```

2. 显示当前的数据库，show databases;

```
MiniDB> show databases;  
SQL TYPE: #SHOW DATABASES#  
SQL STATEMENT: show databases  
DATABASE LIST:  
haojs
```

```
MiniDB>
```

3. 使用某个数据库， use haojs;

```
MiniDB> use haojs;  
SQL TYPE: #USE#  
SQL STATEMENT: use haojs  
DB NAME: haojs
```

```
MiniDB>
```

4. 创建表： create table stu (num int, score float, gender char(1), primary key (num));

```
MiniDB> create table stu (num int, score float, gender char(1), primary key  
    (num));  
SQL TYPE: #CREATE TABLE#  
SQL STATEMENT: create table stu ( num int , score float , gender char ( 1 ) ,  
    primary key ( num ) )  
TABLE NAME: stu  
COLUMN: num  
TYPE: int  
COLUMN: score  
TYPE: float  
COLUMN: gender  
PRIMARY KEY: num  
Creating table: stu  
Table file /Users/anapodoton/MiniDBData/haojs/stu.records created!  
Catalog written!
```

```
MiniDB>
```

5. 展示数据库中创建的表， show tables;

```
MiniDB> show tables;  
SQL TYPE: #SHOW TABLES#  
SQL STATEMENT: show tables  
CURRENT DATABASE: haojs  
TABLE LIST:  
stu
```

```
MiniDB> |
```

6. 向表中插入数据, insert into stu values (111, 85.5, '0');

insert into stu values (333, 95.5, '1');

```
MiniDB> insert into stu values (111, 85.5, '0');  
SQL TYPE: #INSERT#  
SQL STATEMENT: insert into stu values ( 111 , 85.5 , '0' )  
TABLE NAME: stu  
0 : 111  
1 : 85.5  
2 : 0
```

```
MiniDB> insert into stu values (333, 95.5, '1');  
SQL TYPE: #INSERT#  
SQL STATEMENT: insert into stu values ( 333 , 95.5 , '1' )  
TABLE NAME: stu  
0 : 333  
1 : 95.5  
2 : 1
```

7. 查询, select * from stu;

```
MiniDB> select * from stu;  
SQL TYPE: #SELECT#  
SQL STATEMENT: select * from stu  
TABLE NAME: stu  
num      score    gender  
111      85.5      0  
333      95.5      1
```

8. 删除,delete from stu where score=85.5;

```
MiniDB> delete from stu where score=85.5;  
SQL TYPE: #DELETE#  
SQL STATEMENT: delete from stu where score = 85.5  
TABLE NAME: stu  
score 0 85.5
```

```
MiniDB> select * from stu;  
SQL TYPE: #SELECT#  
SQL STATEMENT: select * from stu  
TABLE NAME: stu  
num      score    gender  
333      95.5      1
```

```
MiniDB> |
```

9. 更新, update stu set score = 98.5 where score =95.5;

```

MiniDB> update stu set score = 98.5 where score =95.5;
SQL TYPE: #UPDATE#
SQL STATEMENT: update stu set score = 98.5 where score = 95.5
TABLE NAME: stu
score 98.5
score 0 95.5

```

```

MiniDB> select * from stu;
SQL TYPE: #SELECT#
SQL STATEMENT: select * from stu
TABLE NAME: stu
num      score    gender
333      98.5     1

```

10. 创建索引, create index stuNum on stu (num);

```

MiniDB> create index stuNum on stu (num);
SQL TYPE: #CREATE INDEX#
SQL STATEMENT: create index stuNum on stu ( num )
INDEX NAME: stuNum
TABLE NAME: stu
COLUMN NAME: num
*****
KeyCount: 1, NodeCount: 1, Level: 1, Root: 0
-----
BlockNum: 0 Count: 1, Parent: -1  IsLeaf:1
Keys: { 333      }
Vals: { 0000000  }
NextLeaf:  -1





```

```

MiniDB> select * from stu;
SQL TYPE: #SELECT#
SQL STATEMENT: select * from stu
TABLE NAME: stu
num      score    gender
333      98.5     1
*****
KeyCount: 1, NodeCount: 1, Level: 1, Root: 0
-----
BlockNum: 0 Count: 1, Parent: -1  IsLeaf:1
Keys: { 333      }
Vals: { 0000000  }
NextLeaf:  -1

```

我们在MiniDBData目录下可以看到如下的文件。

 catalog	 stu.records
 haojs ▶	 stuNum.index