**API模块设计说明**

浙江大学·计算机科学与技术学院·软件工程专业

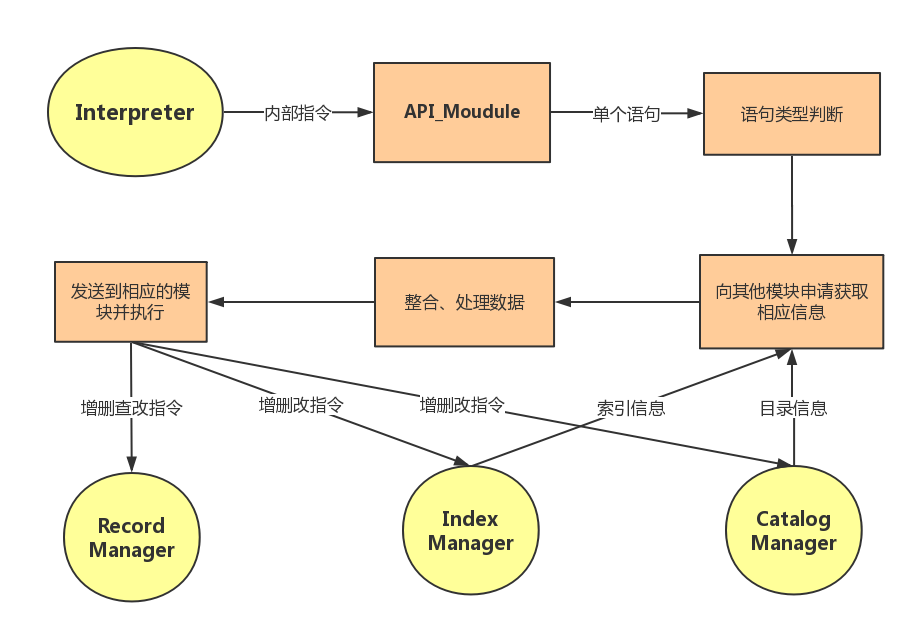
程浩然 傅诤哲 叶慕祈 石磊 钱程

API模块是整个系统的核心。其主要功能是根据Interpreter层解释生成的内部命令，将其进行解析之后，统筹调度Record Manager、Index Manager和Catalog Manager完成相应的功能操作。

**1 实现原理**

API模块会首先解析Interpreter模块发来的SQL字符串指令，并分辨其指令类型并验证其合法性；随后，API模块将向Catalog Manager或Index Manager模块发送请求，以获取完成SQL指令功能所需要的数据；在数据准备步骤完成之后，API模块会对获取到的数据进行整合，并分发给Catalog Manager、Index Manager与Record Manager模块，以完成对底层数据库文件的相应操作，从而最终实现SQL相应的功能。

**2 算法说明**

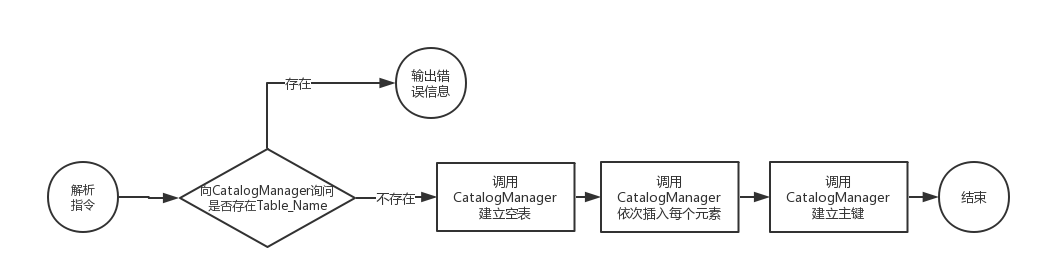


本模块接收Interpreter发来的指令，并解析成相应类型的指令：创建数据表/创建索引/删除数据表/删除索引/查询数据/插入数据/删除数据/帮助/退出指令。对每个单独的指令，本模块会预先调用其他的子模块做好数据准备，并将准备好的数据整合之后，发送给相应的子模块进行执行。

|  |  |
| --- | --- |
| **内部指令对应表** | |
| SQL语句 | 内部指令 |
| create database name; | 00name |
| create table name(a int, b float, c char(1), primary key(a)); | 01name,a + 0,b - 0,c 1 0,a #, |
| create index name on table(attr); | 02name table attr |
| drop database name; | 10name |
| drop table name; | 11name |
| drop index name; | 12name |
| select \* from table; | 20\*,table |
| select \* from table where a = 10 and b >= 20; | 21\*,table,a = 10, b >= 20, |
| insert into student values (‘12345678’,’wy’,22,’M’); | 30name,12345678,wy,22,m |
| delete from name; | 40name |
| delete from name where a = 10 | 41name,a = 10 |
| execfile filename; | 50filename |
| help; | 80 |
| quit; | 70 |

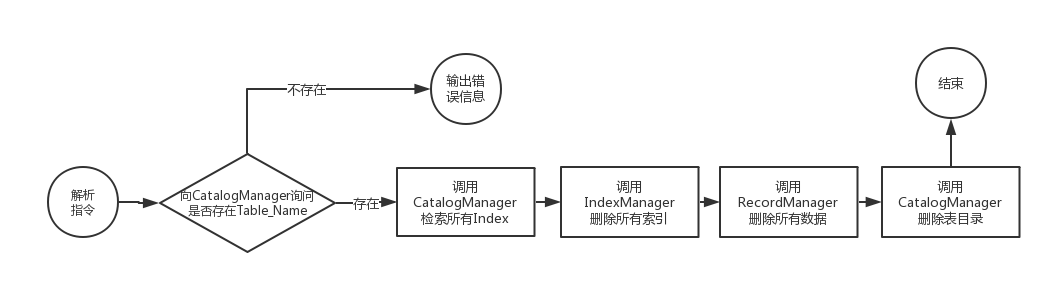
**2.1 创建数据表**

当Interpreter传入指令表头为00时，语句被解析为创建数据表指令。并解析出Table\_Name（新建表名）等信息。API模块会执行以下操作：



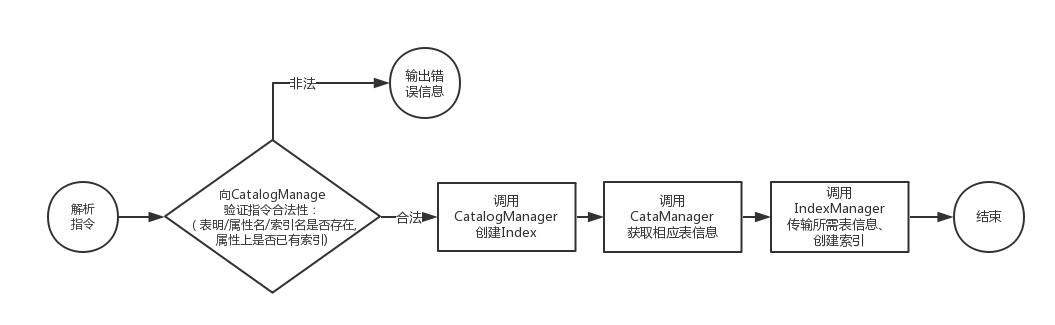
**2.2 删除数据表**

当Interpreter传入指令表头为11时，语句被解析为删除数据表指令。并解析出Table\_Name（删除表名）等信息。API模块会执行以下操作：



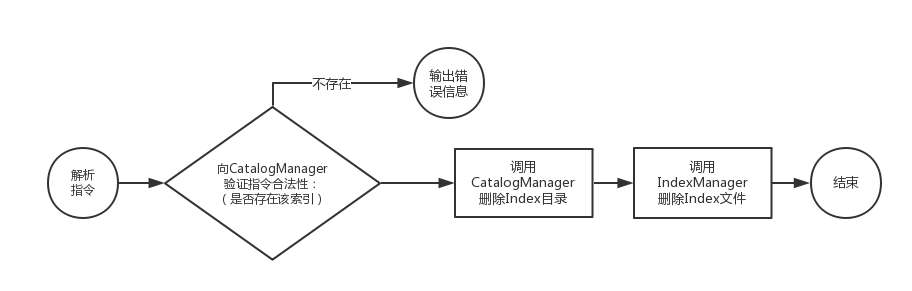
**2.3 创建索引**

当Interpreter传入指令表头为02时，语句被解析为创建索引指令。并解析出name（索引名）、table（表名）、attr（属性名）等信息。API模块会执行以下操作：



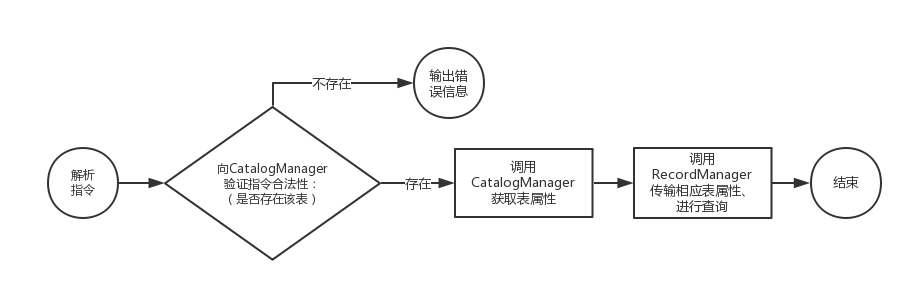
**2.4 删除索引**

当Interpreter传入指令表头为02时，语句被解析为删除索引指令。并解析出name（索引名）。API模块会执行以下操作：



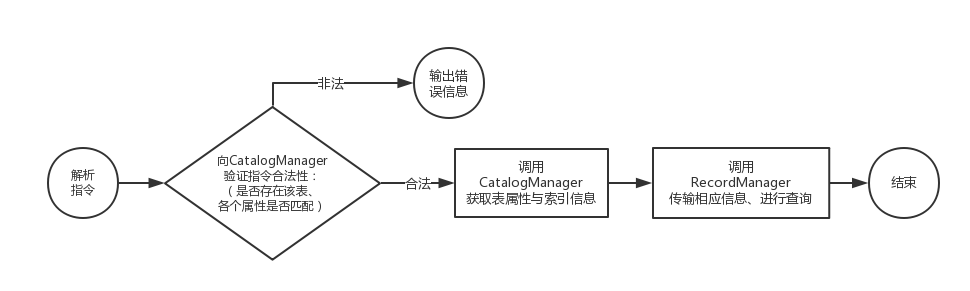
**2.5 查询数据（不含where）**

当Interpreter传入指令表头为20时，语句被解析为查询数据（不含where）指令。并解析出Table\_Name（表名）等信息。API模块会执行以下操作：

****

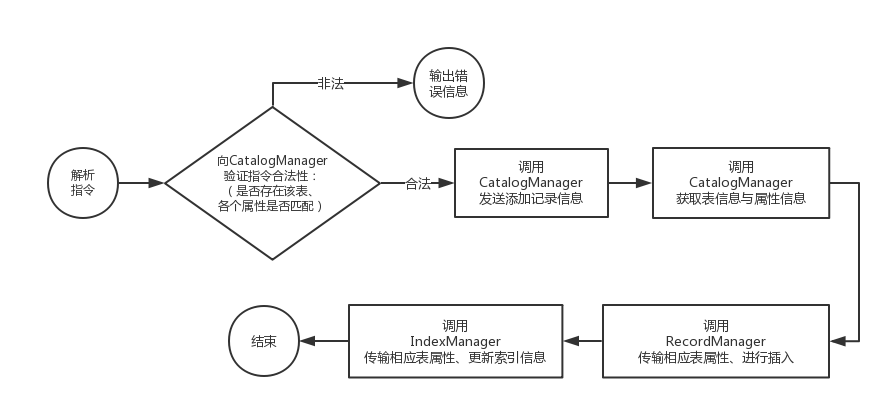
**2.6 查询数据（含where）**

当Interpreter传入指令表头为21时，语句被解析为查询数据（含where）指令。并解析出Table\_Name（表名）、各个条件（Attr、op、Num数组）等信息。API模块会执行以下操作：

****

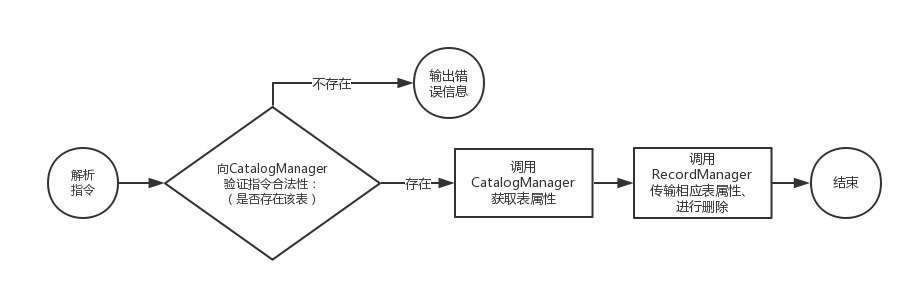
**2.7 插入数据**

当Interpreter传入指令表头为30时，语句被解析为插入数据指令。并解析出Table\_Name（表名）、各个属性值等信息。API模块会执行以下操作：

****

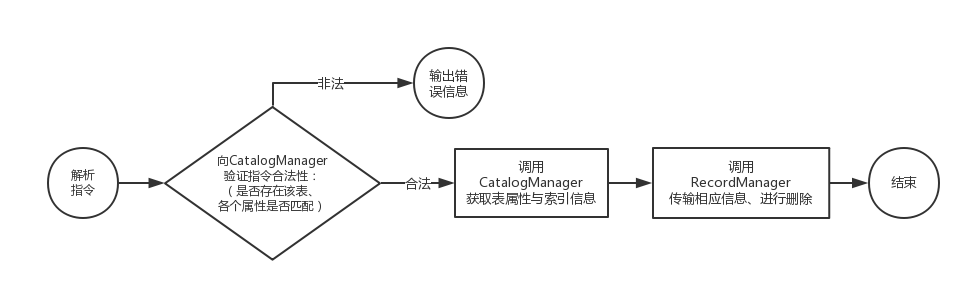
**2.8 删除数据（不含where）**

当Interpreter传入指令表头为40时，语句被解析为删除数据（不含where）指令。并解析出Table\_Name（表名）等信息。API模块会执行以下操作：



**2.9 删除数据（含where）**

当Interpreter传入指令表头为31时，语句被解析为删除数据（含where）指令。并解析出Table\_Name（表名）、各个条件（Attr、op、Num数组）等信息。API模块会执行以下操作：

****

**2.10 帮助请求**

当Interpreter传入指令表头为80时，语句被解析为帮助指令，并直接输出帮助信息。

**2.11 退出指令**

当Interpreter传入指令表头为70时，语句被解析为帮助指令，并返回0值。当主函数接收到API模块返回的0值时，会终止读入死循环，并退出主程序。

**3 接口详情**

* **API与Catalog Manager**

API会根据interpreter的指令，调用Catalog M的以下函数：

1．建立和删除记录

|  |  |
| --- | --- |
| **建立数据表** | Create\_Table(string Table\_Name); |
| **建立表元素** | Create\_Attr(string Table\_Name, string Attr\_Name, string Attr\_Type, string Attr\_Argu)  其中Attr\_Argu为char(n)中的n，类型为string |
| **删除数据表记录** | Drop\_Table(string Table\_Name); |

注：这里建立与删除的都是数据表/元素的目录信。具体的数据（DBfile里的数据）的空间的释放是通过IndexManager和RecordManager实现的。

2.查询相关信息

|  |  |
| --- | --- |
| **表信息相关** | |
| **检查数据库是否存在** | int DBExists(string database) |
| **检查表是否存在** | int tableExists(string database, string table) |
| **询问表信息** | Table get\_table\_info(string table); |
| **检查列信息** | Int attrExists(string database, string table, sstring attr) |
| **检查某字段是否唯一** | int isUnique(string table, string attr) |
| **索引相关** | |
| **检查某字段是否有索引** | int hasIndex(string table, string attr) |
| **返回字段所有索引** | Index indexName(string table, string attr) |
| **检查某表是否有主键** | int isPK(string table, string attr) |
| **寻找表中主键** | CString pkOnTable(string table) |
| **获取某索引信息** | get\_index\_info(string index) |
| **获取所有索引信息** | get\_all\_index() |
| **返回索引的总数** | int indexCount(string table) |
| **获取索引所在表** | CString tableIndexOn(string index) |
| **获取索引所在字段** | CString attrIndexOn(string index) |

* **API与Index Manager**

API模块会根据Interpreter的指令，调用Index M的以下函数：

**1.新建删除索引文档**

|  |  |
| --- | --- |
| **建立索引** | **Create\_Index(string Table\_Name, int N,**  **int Attr\_N，int\* Attr\_Type, string\* Attr\_Name,**  **string Attr, string IndexName, int IndexType)**  注释：  Table\_Name是表名；  Attr\_Type是表内各个参数的类型 ( 1:int 2:float -n:char(n) );  Attr\_Name是表内各个参数的名字;  Int Attr\_N是表中的属性数（列数，也就是粉色数组的size）  string Attr, 索引所在参数  string IndexName, 索引名  int IndexType 索引的属性  IndexType定义规则：1为int，2为float，-n为char(n) |
| **删除索引** | **Delete\_Index(string IndexName)** |

**2.向索引文档添加/删除元素**

|  |  |
| --- | --- |
| **添加元素** | **Index\_InsertElement(int N, string IndexName, int IndexType,**  **int Key1, float Key2, string Key3)**  注释：  N为该元素在record文档中应在的地址  IndexType的定义规则同上  Key为元素的具体值（当Type为int时，Key1有效，Key2/Key3传入的值为空，不用管它；其它类型同理） |
| **删除元素** | 删除操作由record发出请求 |

【Record M】

API模块会调用Record M的以下函数：

|  |  |
| --- | --- |
| **查询 SELECT** | |
| **不含Where的查询** | **Select\_All(int N, Table\_Name, int Attr\_N，**  **int\* Attr\_Type, string\* Attr\_Name)** |
| **含Where的查询** | **Select(int N, Table\_Name, int Attr\_N,**  **int\* Attr\_Type, string\* Attr\_Name,**  **Int op\_N, string\* Attr, int\* op, string\* Num,**  **string\* Index)**  注释：  Table\_Name是表名；  Attr\_Type是表内各个参数的类型 ( 1:int 2:float -n:char(n) );  Attr\_Name是表内各个参数的名字;  Int N是表中的记录数（行数）  Int Attr\_N是表中的属性数（列数）  *SELECT \* FROM Table\_Name WHERE (Attr[1] op[1] Num[1]) AND (…) AND (…) AND …*  op\_N是where后条件的个数  Attr是参与where的属性名  op是算术比较符：= <> < > <= >= 分别对应 0 1 2 3 4 5  Num是参与比较的值，跟Attr的类型有关（传过去的是string类型）  Index是某个Attr是否含有索引，有则返回索引名，无则为空  （查询操作中需要RecordM与IndexM进行交互） |
| **插入 INSERT** | |
| **插入数据** | **Insert(int N,Table\_Name, int Attr\_N,**  **int\* Attr\_Type, string\* Attr\_Name,**  **string\* value)**  这里的value是数据值（已经验证过每个数据与其对应的行属性的类型了，也就是说这个value是必定合法的） |
| **删除 DELETE** | |
| **删除所有数据** | **Delete\_All(int N,Table\_Name)** |
| **含Where的删除** | **Delete(int N,Table\_Name, int Attr\_N,**  **int\* Attr\_Type, string\* Attr\_Name,**  **Int op\_N, string\* Attr, int\* op,**  **string\* Num, string\* Index)**  （删除操作中需要RecordM与IndexM进行交互）  参数条件说明同SELECT |