AIMDB简介

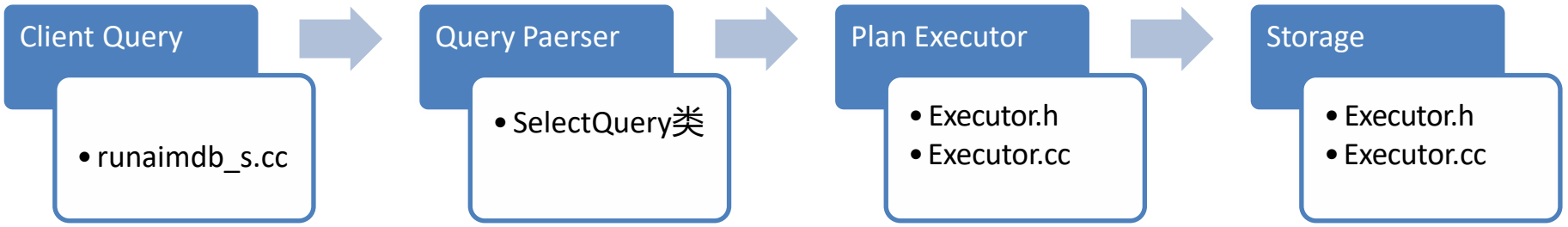
AIMDB

- 一个内存数据库系统
- 支持数据存储和索引功能
- 主要模块包括
  - 数据库框架、存储管理、索引管理、查询执行模块

代码结构

- 代码目录结构
  - System: 主目录，实现数据存储、访问、索引
  - Debug: 调试目录
  - Example: 程序示例
- 编译测试

AIMDB




```
int main (void) {

    if (global_init ()) return -1;
    if (load_schema (file_schema)) return -2;
    if (load_data   (file_data,table_name, 2)) return -3;

    ///< here start to test your code by call executor
    test ();

    global_shut ();
    return 0;

}
```

Schema

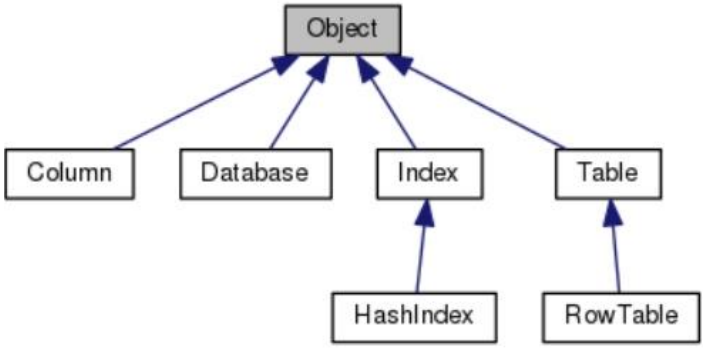
load_schema()中调用的schema文件

example_schema.txt

```
DATABASE    example_database_1
    TABLE   example_table_1 ROWTABLE
        COLUMN  example_column_1    INT8
        COLUMN  example_column_2    INT16
        COLUMN  example_column_3    INT32
        COLUMN  example_column_4    INT64
        COLUMN  example_column_5    CHARN   16
        COLUMN  example_column_6    FLOAT32
        COLUMN  example_column_7    FLOAT64
        COLUMN  example_column_8    DATE
        COLUMN  example_column_9    TIME
        COLUMN  example_column_10   DATETIME
        INDEX   example_index_1 HASHINDEX   example_column_1    example_column_2    example_column_5
```

TASK1: 数据库框架 (schema)

- 包括2个部分(Schema和Catalog)
- Schema部分(schema.h)
  - 定义Database,Table,Column,Index的上层接口
  - 保持统一的接口，便于扩展实现其底层结构



类继承关系图

Object Definition

```
int64_t o_id;
ObjectType o_type;
char o_name[OBJ_NAME_MAX];
```

数据库框架(schema)

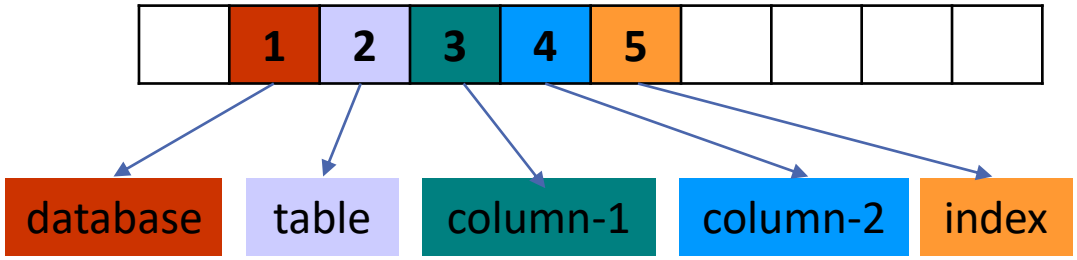
- Table接口(schema.h)
  - bool select (int64_t record_rank, char *dest)
  - bool select (char *row_pointer, char *dest)
  - bool selectCol (int64_t record_rank, int64_t column_rank, char *dest)
  - bool selectCol (char *row_pointer, int64_t column_rank, char *dest)
  - bool selectCols (int64_t record_rank, int64_t column_total, int64_t *column_ranks, char *dest)
  - bool selectCols (char *row_pointer, int64_t column_total, int64_t *column_ranks, char *dest)

Table接口举例

```
/**
 * select all columns' data.
 * @param  record_rank the n th row in the table storage
 * @param  dest        buffer to store result
 * @retval true        success
 * @retval false       failure
 */
virtual bool select(int64_t record_rank, char *dest) {
    return false;
}
/**
 * select several column data.
 * @param  record_rank  the n th row in the table storage
 * @param  column_total total number of columns to select
 * @param  column_ranks array of column_rank, column_rank is the n th column in table pattern
 * @param  dest         buffer to store result
 * @retval true         success
 * @retval false        failure
 */
virtual bool selectCols(int64_t record_rank, int64_t column_total,
                        int64_t * column_ranks, char *dest) {
    return false;
}
```

数据库框架(catalog)

- Catalog，元数据表(catalog.h)
  - std::vector <Object *> cl_id_obj;



  - std::unordered_map <std::string, Object *> cl_name_obj;
- Catalog，存储数据库对象的容器
  - 数据库对象–Database、Table、Column、Index

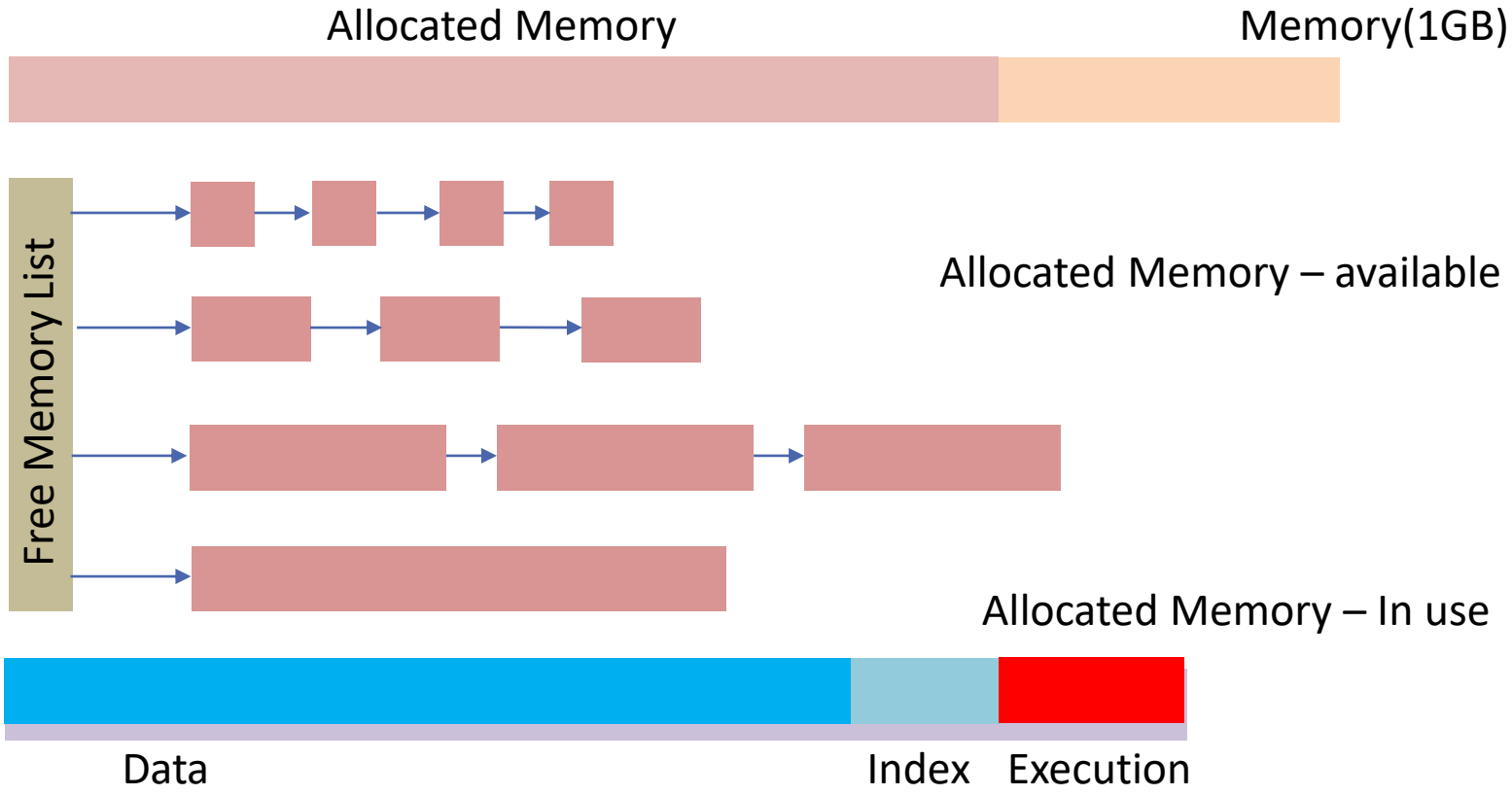
数据库框架(catalog)

- 数据库对象：Database, Table, Column, Index
- 建立对象
  - catalog.createXXX to create objects
- 获得对象地址
  - catalog.getObjById/Name to get pointer

- 定义对象间的关系
- 建立对象间关系之后，初始化数据库
  - catalog.initDatabase

TASK2: 存储管理

- 分配整块内存，减少malloc次数



存储管理

- 通用接口(mymemory.h)
  - int64_t alloc(char *&p, int64_t size);
  - int64_t free(char *p, int64_t size);

- 注意事项
  - init初始化时，参数total必须是2的次幂
  - alloc,free都需要size; 所以alloc后需要记录size

索引管理(hash)

- Index接口(schema.h)
  - bool set_ls(void *i_data1, void *i_data2, void *info);
  - bool set_ls(void *i_data1[], void *i_data2[], void *info);
  - bool lookup(void *i_data, void *info, void *&result);
  - bool lookup(void *i_data[], void *info, void *&result);

  - bool insert(void *i_data, void *p_in);
  - bool insert(void *i_data[], void *p_in);
  - bool del(void *i_data);
  - bool del(void *i_data[]);

TASK3: 查询执行

- 操作符(operator)
  - 选择，选择条件(函数指针)
  - 投影，投影数据(data or pointer)
  - 连接，(索引用法)
  - 分组统计，(统计量数据的存储)
  - 排序，(排序方法)
- 方法
  - 上层operator调用下层operator的函数(Volcano)
  - 使用有限容量的中间表缓存数据(ResultTable)
  - 自顶向下处理数据，节省运行占用空间

查询执行

- 输入数据介绍(executor.h)
  - schema文本文件(.txt)和table数据文件(.tab)
  - 分析请求以SelectQuery类给出，main函数调用
- 输出要求说明(executor.h)
  - 结果以ResultTable表示，然后将其输出到标准输出stdout，代码中提供了一个参考输出函数
  - 每行代表一条记录，记录内各列以一个TAB(\t)分隔，行尾没有TAB，只有换行符(\n)
  - 最终提交的结果中stdout不要有任何其他输出

查询执行

- 怎么在AIMDB中做?
  - 继承Operator类实现具体功能的子类
  - 解析SelectQuery生成QueryPlan(Operator调用关系树)
  - 使用上述子类实现Executor的exec接口函数

```
int Executor::exec(SelectQuery *query, ResultTable *result)
```

- Tips
  - 充分利用doxygen生成的说明文档
  - 从debug目录下的部件测试中了解用法

练习

- TASK1: Schema
  - 支持的数据类型有哪些?
  - Table是如何定义的?
  - Index的类型有哪些?
  - 支持“键”吗?
  - Create table过程?
  - 编程练习：Print schema（参考debug）
    - eg. 请打印example_column_2（example_schema.txt）

```
Column- c_id: 4 c_type: 2 c_size: 2 c_name: example_column_2 c_offset: 1
```

练习

- TASK2: Data
  - 每行每列如何存储?
  - 编程练习：打印对应record id的行
- TASK3: Executor
  - 解析SelectQuery类
  - 解析ResultTable类
  - 编程练习：Query选择对应record id的列放到result table

中国科学院大学
University of Chinese Academy of Sciences

祝顺利！