# 数据库接口文档

物联仓储解决方案

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | 文件标识： | ZJM-SJKJK-001 |
| 当前版本： | 1.1 |
| 作 者： | 张家铭 |
| 完成日期： | 2016/10/13 |

**班级：星创客16071——秋名山的老司机们**

**组长：胡梦龙**

**组员：胡梦龙、张家铭、朱秀非**

**李璐婷、梁怀文、赵赫、王恒**

**编制：王恒**

**保密级别：一级保密。仅限本组观看，禁止外传**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修改历史：** | | | | |
| 版本 | 时间 | 修改人 | 内容 | 备注 |
| 1.0 | 2016/10/13 | 张家铭 | 建立 |  |
| 1.1 | 2016/10/14 | 张家铭 | 修改查找返回值 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 保密协议

本文档所有者是上述研发组（秋明山的老司机们）。本文档的持有人（以下简称“持有人”）是且仅是上述研发组成员，持有人有随时查看本文档的权利，在获得本文档编写人员的同意下，持有人允许修改本文档中部分内容。本文档的持有人有义务保护本文档不被他人私自查看、篡改、复制或伪造。持有人以外人员在获得文档编写人员的许可下，仅可以查看本文档内容，不得记录、复制、篡改或伪造。任何人查看本文档时，都有义务保证本文档的相关内容不被泄密。本文档不设保密期限，以公开声明（该声明需研发组长手签）的形式解除对本文档的保密。

一切查看本文档的人员皆视为同意并遵守以上协议，一旦出现违背上述协议的情况，本文档所有者（秋名山的老司机们）有追究相关责任人法律责任的权利。

This document belongs to the development of 秋明山的老司机们, and the owners are only the developers of this group. The owners have the rights to read this document, and modify this document with the permissions of the editors of this document. The owners have the obligations to protect this document not to be read, modify, copy or plagiarized by the others. Man who is out of the owners can only read this document without recording, copy, modify or plagiarized under the permissions of the editors of this document. Everyone who read this document has the obligations to protect the secret of the document. This document has no limit of dates to abolishment this secret declaration, and it can be abolished by the public declaration (with leader’s signature).

Everyone who read this document are all consider as to follow this secret declaration, and the owners (秋名山的老司机们) have rights to ask the illegal responsibilities who has obey this declaration.

## 数据库

库名：QMSSLSJ.db

### 辅助表SqlHelper\_Sqlite3.h

1. 定义了所有表结构体
2. 定义联合体datatype
3. 定义链表所需所有函数

4．定义了数据库开启及sql语句执行函数。

### 宏定义：

#define DB\_NAME "QMSSLSJ.db"

### 结构体：

//物品表

typedef struct{

int ArticleID;

char ArticleName[50];

int StorageID;

int TypeID;

}Tbl\_Article\_Info;

//物品种类表

typedef struct{

int TypeID;

char TypeName[50];

}Tbl\_Article\_Type;

//视频历史记录表

typedef struct{

int CameraID;

int StorageID;

char CameraPath[50];

char CameraData[30];

}Tbl\_Camera\_Info;

//设备控制表

typedef struct{

int DeviceID;

bool RFIDSet;

bool AlarmSet;

bool LedSet;

bool BeepSet;

int FanSet;

int SegSet;

bool IsSet;

}Tbl\_Device\_Control;

//设备上传历史记录表

typedef struct{

int HistoryID;

int DeviceID;

char HistoryDate[30];

float Temperature;

float Humidity;

float Luminosity;

int Electricity;

bool IsAlarm;

}Tbl\_Device\_History;

//设备表

typedef struct{

int DeviceID;

int StorageID;

char DeviceName[50];

char Identification[20];

bool IsOnLine;

bool IsAlarm;

}Tbl\_Device\_Info;

//设备参数表

typedef struct{

int DeviceID;

float MinTem;

float MaxTem;

float MinHum;

float MaxHum;

float MinLit;

float MaxLit;

int TimeSet;

bool IsSet;

}Tbl\_Device\_Parameter;

//射频表

typedef struct{

int RFID\_ID;

int ArticleID;

char RFID[20];

}Tbl\_RFID\_Info;

//用户权限表

typedef struct{

int RoleID;

char RoleName[20];

}Tbl\_Role\_Info;

//仓库表

typedef struct{

int StorageID;

char StorageName[50];

char StorageAddress[200];

}Tbl\_Storage\_Info;

//系统参数表

typedef struct{

char key[20];

char value[20];

}Tbl\_System\_Parameter;

//用户表

typedef struct{

int UserID;

char UserName[20];

char Password[20];

char NickName[50];

int RoleID;

}Tbl\_User\_Info;

//联合体

typedef union {

Tbl\_Article\_Info \_Tbl\_Article\_Info;

Tbl\_Article\_Type \_Tbl\_Article\_Type;

Tbl\_Camera\_Info \_Tbl\_Camera\_Info;

Tbl\_Device\_Control \_Tbl\_Device\_Control;

Tbl\_Device\_History \_Tbl\_Device\_History;

Tbl\_Device\_Info \_Tbl\_Device\_Info;

Tbl\_Device\_Parameter \_Tbl\_Device\_Parameter;

Tbl\_RFID\_Info \_Tbl\_RFID\_Info;

Tbl\_Role\_Info \_Tbl\_Role\_Info;

Tbl\_Storage\_Info \_Tbl\_Storage\_Info;

Tbl\_System\_Parameter \_Tbl\_System\_Parameter;

Tbl\_User\_Info \_Tbl\_User\_Info;

}datetype;

//链表节点

typedef struct node{

datetype \*data;

struct node \*next;

}Node,\*INode;

//链表

typedef struct {

INode front;

INode rear;

int lengh;

}SqlQueue,\*SqlLinkQueue;

### 接口：

//创建链表

SqlLinkQueue init\_linkqueue();

成功返回链表指针，失败返回NULL。

//插入链表

bool in\_linkqueue(SqlLinkQueue q,datetype \*value);

参数：q传入链表

value保存信息的结构体

成功返回true，失败返回false。

//释放链表资源

bool free\_linkqueue(SqlLinkQueue q);

参数：q传入链表

value保存信息的结构体

成功返回true，失败返回false。

//创建数据库

sqlite3\* connent\_database();

成功返回一个创建好的DB数据库描述符。失败返回NULL。

//执行sql

bool exec\_sql(char \*sql);

传入一条sql语句执行，只限增、删、改。

sql： sql语句。

成功返回true，失败返回false。

//封装get\_table

int my\_get\_table(int \*pnColum\_p, int \*pnRow\_p, char \*\*\*pazResult\_p, char\* sql);

封装后的get\_table,与Sqlite3\_get\_table类似

参数：pnColum\_p列数，

pnRow\_p行数，

pazResult\_p 结果，

sql语句

## 表接口：

所有表提供统一接口

#### 接口说明

1. 构造（表名+init）

传入参数返回负值好的结构体。

1. 添加（表名+Add）

传入结构体对标进行增加操作。

1. 删除（表名+Delete）

传入结构体对标进行删除操作。

1. 修改（表名+Modify）

传入结构体对标进行修改操作。

1. 查询（表名+Find）

查询表内容，所有内容按行赋值给结构体后插入链表。

返回查询到的条数，失败返回0

1. 查询单条（表名+FindSingle）

查询表返回第一条数据的结构体

1. 分页查询（表名+FindPage）

按照分页查找表内容

返回查询到的条数，失败返回0

1. 查询总页数（表名+GetTotalPageCount）

获取分页表的总页数。

#### 接口实现

##### 物品表

//构造函数

Tbl\_Article\_Info Tbl\_Article\_InfoInit(int ArticleID,char \* ArticleName,int StorageID,int TypeID);

//增加

bool Tbl\_Article\_InfoAdd(Tbl\_Article\_Info \_Tbl\_Article\_Info);

//删除

bool Tbl\_Article\_InfoDelete(char \*Con);

//修改

bool Tbl\_Article\_InfoModify(Tbl\_Article\_Info \_Tbl\_Article\_Info);

//查询

int Tbl\_Article\_InfoFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Article\_Info Tbl\_Article\_InfoFindSingle(char \*Con);

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

int Tbl\_Article\_InfoFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Article\_InfoGetTotalPageCount(char \*Con,int PageSize);

##### 物品类型表

//构造函数

Tbl\_Article\_Type Tbl\_Article\_TypeInit(int TypeID,char \* TypeName);

//增加

bool Tbl\_Article\_TypeAdd(Tbl\_Article\_Type \_Tbl\_Article\_Type);

//删除

bool Tbl\_Article\_TypeDelete(char \*Con);

//修改

bool Tbl\_Article\_TypeModify(Tbl\_Article\_Type \_Tbl\_Article\_Type);

//查询

int Tbl\_Article\_TypeFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Article\_Type Tbl\_Article\_TypeFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Article\_TypeFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Article\_TypeGetTotalPageCount(char \*Con,int PageSize);

##### 视频记录表

//构造函数

Tbl\_Camera\_Info Tbl\_Camera\_InfoInit(int CameraID,int StorageID,char \* CameraPath,char \* CameraDate);

//增加

bool Tbl\_Camera\_InfoAdd(Tbl\_Camera\_Info \_Tbl\_Camera\_Info);

//删除

bool Tbl\_Camera\_InfoDelete(char \*Con);

//修改

bool Tbl\_Camera\_InfoModify(Tbl\_Camera\_Info \_Tbl\_Camera\_Info);

//查询

int Tbl\_Camera\_InfoFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Camera\_Info Tbl\_Camera\_InfoFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Camera\_InfoFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Camera\_InfoGetTotalPageCount(char \*Con,int PageSize);

##### 设备控制表

//构造函数

Tbl\_Device\_Control Tbl\_Device\_ControlInit(int DeviceID,char \* RFIDSet,char \* AlarmSet,char \* LedSet,char \* BeepSet,int FanSet,int SegSet,char \* IsSet);

//增加

bool Tbl\_Device\_ControlAdd(Tbl\_Device\_Control \_Tbl\_Device\_Control);

//删除

bool Tbl\_Device\_ControlDelete(char \*Con);

//修改

bool Tbl\_Device\_ControlModify(Tbl\_Device\_Control \_Tbl\_Device\_Control);

//查询

int Tbl\_Device\_ControlFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Device\_Control Tbl\_Device\_ControlFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Device\_ControlFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Device\_ControlGetTotalPageCount(char \*Con,int PageSize);

##### 历史记录表

//构造函数

Tbl\_Device\_History Tbl\_Device\_HistoryInit(int HistoryID,int DeviceID,char \* HistoryDate,char \* Temperature,char \* Humidity,char \* Luminosity,int Electricity,char \* IsAlarm);

//增加

bool Tbl\_Device\_HistoryAdd(Tbl\_Device\_History \_Tbl\_Device\_History);

//删除

bool Tbl\_Device\_HistoryDelete(char \*Con);

//修改

bool Tbl\_Device\_HistoryModify(Tbl\_Device\_History \_Tbl\_Device\_History);

//查询

int Tbl\_Device\_HistoryFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Device\_History Tbl\_Device\_HistoryFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Device\_HistoryFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Device\_HistoryGetTotalPageCount(char \*Con,int PageSize);

##### 设备表

//构造函数

Tbl\_Device\_Info Tbl\_Device\_InfoInit(int DeviceID,int StorageID,char \* DeviceName,char \* Identification,char \* IsOnLine,char \* IsAlarm);

//增加

bool Tbl\_Device\_InfoAdd(Tbl\_Device\_Info \_Tbl\_Device\_Info);

//删除

bool Tbl\_Device\_InfoDelete(char \*Con);

//修改

bool Tbl\_Device\_InfoModify(Tbl\_Device\_Info \_Tbl\_Device\_Info);

//查询

int Tbl\_Device\_InfoFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Device\_Info Tbl\_Device\_InfoFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Device\_InfoFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Device\_InfoGetTotalPageCount(char \*Con,int PageSize);

##### 设备参数表

//构造函数

Tbl\_Device\_Parameter Tbl\_Device\_ParameterInit(int DeviceID,char \* MinTem,char \* MaxTem,char \* MinHum,char \* MaxHum,char \* MinLit,char \* MaxLit,int TimeSet,char \* IsSet);

//增加

bool Tbl\_Device\_ParameterAdd(Tbl\_Device\_Parameter \_Tbl\_Device\_Parameter);

//删除

bool Tbl\_Device\_ParameterDelete(char \*Con);

//修改

bool Tbl\_Device\_ParameterModify(Tbl\_Device\_Parameter \_Tbl\_Device\_Parameter);

//查询

int Tbl\_Device\_ParameterFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Device\_Parameter Tbl\_Device\_ParameterFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Device\_ParameterFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Device\_ParameterGetTotalPageCount(char \*Con,int PageSize);

##### 射频表

//构造函数

Tbl\_RFID\_Info Tbl\_RFID\_InfoInit(int RFID\_ID,int ArticleID,char \* RFID);

//增加

bool Tbl\_RFID\_InfoAdd(Tbl\_RFID\_Info \_Tbl\_RFID\_Info);

//删除

bool Tbl\_RFID\_InfoDelete(char \*Con);

//修改

bool Tbl\_RFID\_InfoModify(Tbl\_RFID\_Info \_Tbl\_RFID\_Info);

//查询

int Tbl\_RFID\_InfoFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_RFID\_Info Tbl\_RFID\_InfoFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_RFID\_InfoFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_RFID\_InfoGetTotalPageCount(char \*Con,int PageSize);

##### 用户权限表

//构造函数

Tbl\_Role\_Info Tbl\_Role\_InfoInit(int RoleID,char \* RoleName);

//增加

bool Tbl\_Role\_InfoAdd(Tbl\_Role\_Info \_Tbl\_Role\_Info);

//删除

bool Tbl\_Role\_InfoDelete(char \*Con);

//修改

bool Tbl\_Role\_InfoModify(Tbl\_Role\_Info \_Tbl\_Role\_Info);

//查询

int Tbl\_Role\_InfoFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Role\_Info Tbl\_Role\_InfoFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Role\_InfoFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Role\_InfoGetTotalPageCount(char \*Con,int PageSize);

##### 仓库表

//构造函数

Tbl\_Storage\_Info Tbl\_Storage\_InfoInit(int StorageID,char \* StorageName,char \* StorageAddress);

//增加

bool Tbl\_Storage\_InfoAdd(Tbl\_Storage\_Info \_Tbl\_Storage\_Info);

//删除

bool Tbl\_Storage\_InfoDelete(char \*Con);

//修改

bool Tbl\_Storage\_InfoModify(Tbl\_Storage\_Info \_Tbl\_Storage\_Info);

//查询

int Tbl\_Storage\_InfoFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_Storage\_Info Tbl\_Storage\_InfoFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_Storage\_InfoFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_Storage\_InfoGetTotalPageCount(char \*Con,int PageSize);

##### 系统参数表

//构造函数

Tbl\_System\_Parameter Tbl\_System\_ParameterInit(char \* key,char \* value);

//增加

bool Tbl\_System\_ParameterAdd(Tbl\_System\_Parameter \_Tbl\_System\_Parameter);

//删除

bool Tbl\_System\_ParameterDelete(char \*Con);

//修改

bool Tbl\_System\_ParameterModify(Tbl\_System\_Parameter \_Tbl\_System\_Parameter);

//查询

int Tbl\_System\_ParameterFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_System\_Parameter Tbl\_System\_ParameterFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_System\_ParameterFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_System\_ParameterGetTotalPageCount(char \*Con,int PageSize);

##### 用户表

//构造函数

Tbl\_User\_Info Tbl\_User\_InfoInit(int UserID,char \* UserName,char \* Password,int RoleID,char \* NickName);

//增加

bool Tbl\_User\_InfoAdd(Tbl\_User\_Info \_Tbl\_User\_Info);

//删除

bool Tbl\_User\_InfoDelete(char \*Con);

//修改

bool Tbl\_User\_InfoModify(Tbl\_User\_Info \_Tbl\_User\_Info);

//查询

int Tbl\_User\_InfoFind(SqlLinkQueue list,char \*Con);

//查询单条

Tbl\_User\_Info Tbl\_User\_InfoFindSingle(char \*Con);

/\*

\* 分页查找

\* Con 查找条件

\* Sort 排序条件

\* PageSize 每页多少条

\* CurrentPageIndex 第几页

\*/

int Tbl\_User\_InfoFindPage(SqlLinkQueue list,char \*Con,char \*Sort,int PageSize,int CurrentPageIndex);

//获取总页数

int Tbl\_User\_InfoGetTotalPageCount(char \*Con,int PageSize);