| Project Name | | Confidentiality Level |
|---|---|---|
| Database Management System | | Public |
| Project ID | Version | Document Code |
| 00001 | V1.0 | 01 |

# Database Management System
# System Design Specification

# Revision Record

| Date | Revision Version | CR ID /Defect ID | Sec No. | Change Description | Author |
|---|---|---|---|---|---|
| 4.29 | 1.0 | Null | 01 | Frist write | 15301014 |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

# Content



Keywords:
DBMS RDBMS MFC SDI Dialog C/S MVC Integrity Database Table Filed Record

**Abstract:**

This system is the first version of the database management system, including database creation, database table management, record addition and query, integrity constraint implementation, index creation and implementation, multi-user, concurrency processing, transaction processing, database backup and recovery, etc. This document describes the design of each function module to help the software developers analyze and design the software better. At the same time, it helps the clients to give better suggestions.

| Abbreviations | Full spelling | Chinese explanation |
|---|---|---|
| SDI | Single Document Interface | 单文档界面 |
| MFC | Microsoft Foundation Classes | 微软基础类库 |
| DBMS | Database Management System | 数据库管理系统 |
| RDBMS | Relational Database Management System | 关系型数据库管理系统 |
| C/S | Client/Server | 客户/服务器模式 |
| MVC | Model View Controller | 模型-视图-控制器 |

# 1 Introduction

## 1.1 Purpose

This document describes the Database Management System's design process, including general design and detailed design, in order to conduct project team to implement coding and unit testing. Expected reader of this specification is intermediate user (refers to project team member, client representative, testing staff, QA and etc.)

## 1.2 Scope

### 1.2.1 Name

Database Management System

### 1.2.2 Functions

Software functions please refer to the requirements specification document: requirement analysis of Database Management System.

### 1.2.3 Applications

Not available.

# 2 Level 0 Design Description

## 2.1 Software System Context Definition

The architecture of the database management system includes: DBMS architecture, user interface, syntax analysis, query processing, directory management, concurrency control, recovery mechanism, physical storage management, etc. This system mainly implements the basic functions of query processing, directory management, concurrency control, and recovery mechanism. The physical storage management directly utilizes the file management function of the operating system. The syntax analysis and user interface will not be implemented temporarily.

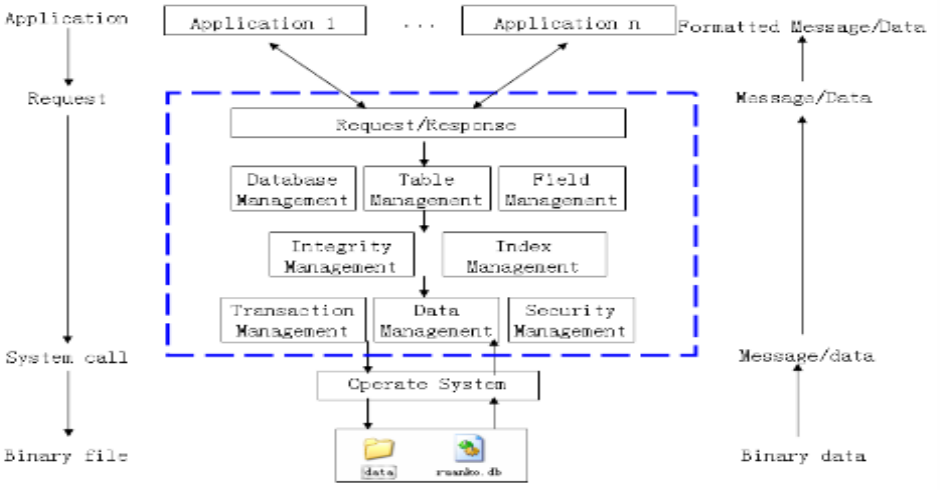



**Figure 2-1 Database management system structure diagram**

This system is a desktop application with window interface. The architecture is C/S structure. Many servers can be created on the whole network. Each server is responsible for its own data storage. Through the network, the client can connect to any one or many servers. A server can provide service for many clients. The system architecture diagram is as follows:

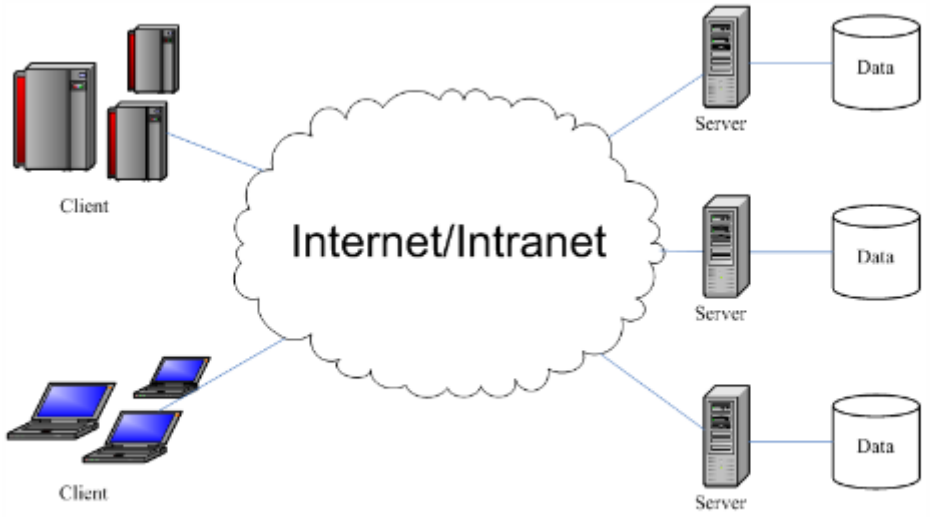
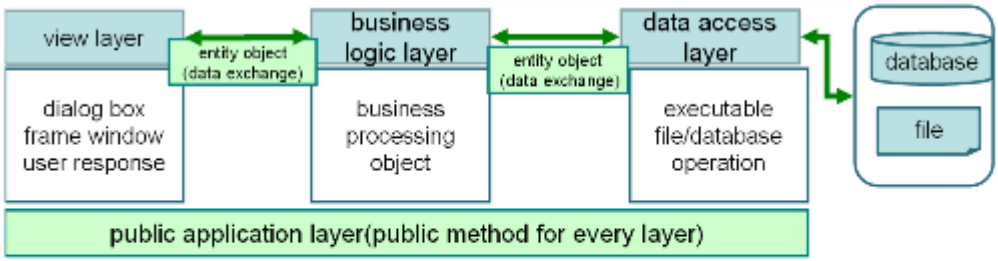



**Figure 2-2 Database Management System Structure**

## 2.2 Design Considerations

### 2.2.1 Design Alternatives

**1. Program structure design**



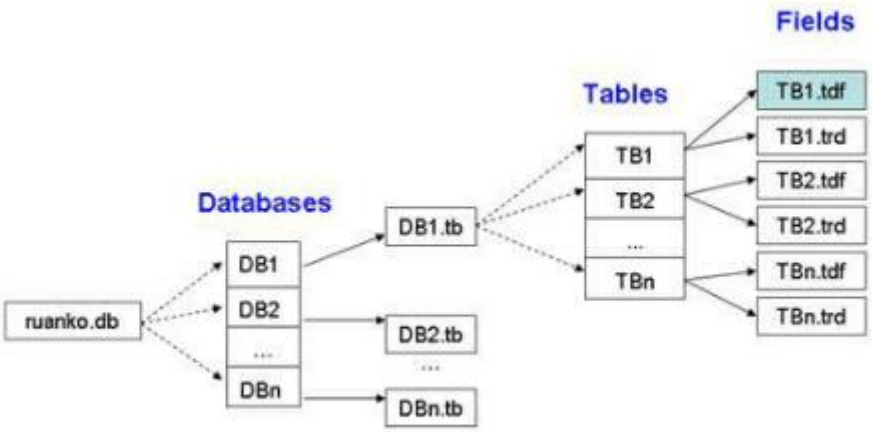

Based on the logic responsibility, the program software structure can be divided into 3 layers: presentation layer, business logic layer and data access layer. Data in each layer is transferred by "entity class" (data object). Besides, public class that irrelative with business may be used in layers in the program as the "Tool class."

**2. Data Storage Structure**

The system stores data with the binary file of the operating system, and saves the definition data and data information with the folder and file. DBMS system definition files include the database description file (ruanko.db), table description file (*.tb), table definition file (saves field information, *.tdf), index description file (*.tid), integrity description file (*.tic). The data files of DBMS include the record file (*.trd), index data file (*.ix), log file (*.log), transaction data file (*.tac), temporary file (*.tmp), etc.

The relation diagram between the database description file, table description file, table definition file and record file is as follows:




## 2.2.2 Design Constraints

### 2.2.2.1 Standards compliance

Could expand the specifications that do not exist in the following requirements, but it cannot contrary to the standard. It follows :< COE technical requirement standard of Ruanko Lab>, <COE programming standard requirement of Ruanko Lab>

### 2.2.2.2 Hardware Limitations

The minimum configuration of the hardware:

CPU: 1GHZ

RAM: 128MB

To ensure that the game can run smoothly, the configuration best meet the following requirements:

CPU: 1.8GHZ

RAM: 1G

### 2.2.2.3 Technology Limitations

Parallel operation: Allow multiple games running at the same time, and can ensure the correctness and completeness of the data.

Coding standard: COE programming standard requirement of Ruanko Lab

## 2.2.3 Other Design Considerations

Not available.

# 3 Level 1 Design Description

## 3.1 System Architecture

### 3.1.1 Description of the Architecture

The system is developed according to the thought of "divide and rule". The functions are divided into many modules to manage and develop individually. The detailed module division of the system is shown below:

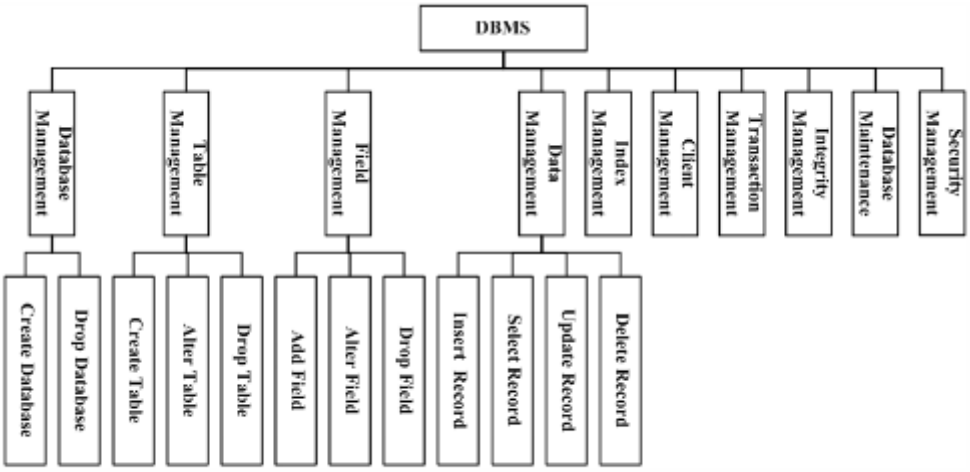



**Figure 3-3 Database management system function structure diagram**

Modules are divided according to three-layer structure and shown below:

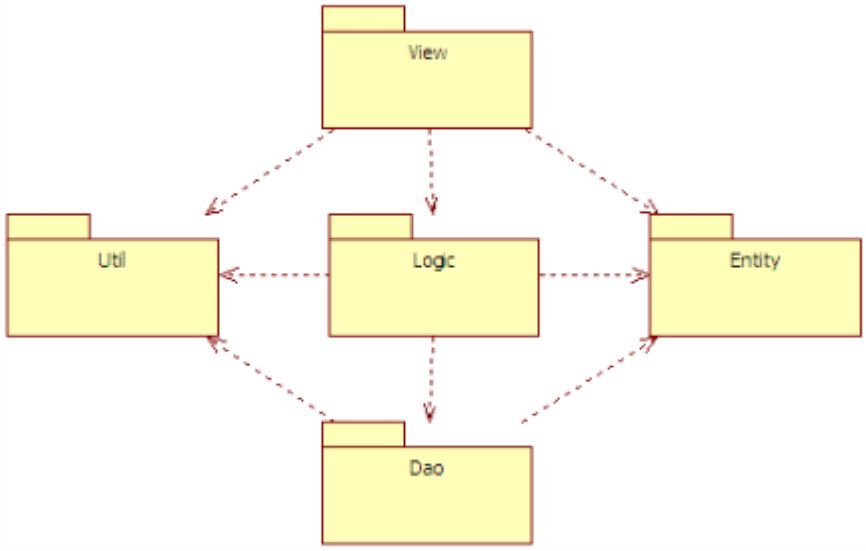



**Figure 3-4 Program structure diagram**

1. Project Structure
   This system is divided into two main parts of the database server project and database enterprise manager. The client is a SDI project. The server is a dialog project.
   The project name of the client is RKDBMS.
   The project structure is shown below:

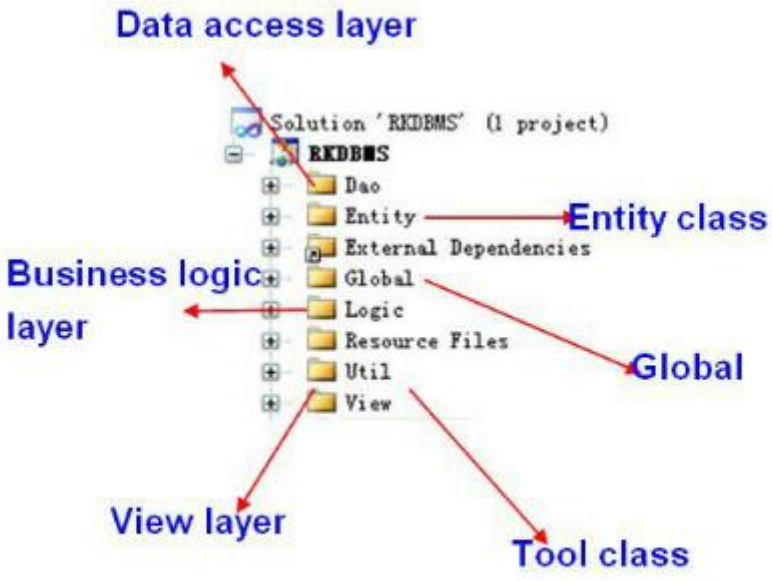



**Figure 3-5 Client project structure diagram**

The structure of server program is the same as the client. ( omitted )

2. Project program structure
   Make logic to program by by VS2012 and VS2017 Solution Explorer through Filter

| Layer | Filter | Class |
|---|---|---|
| view layer | View | frame class-CMainFrame<br>view class-CRKDBMSView<br>dialog box class |
| business logic layer | Logic | Document class-CRKDBMSDoc<br>business logic process class |
| data access layer | Dao | the class of writing and reading operation on data file |
| | Entity | entity class |
| | Util | tool class |
| | Global | global defined file<br>application class--CRKDBMSApp |

## 3.2 Decomposition Description

### 3.2.1 Database Management

1. Overview

   Create and delete the database. Implement database definition file creation, modification and query.

2. Functions

| Module Name | Function Name | Function Description |
|---|---|---|
| Database Management | Create Database | Implement the function of database creation. Corresponding SQL statement: CREATE DATABASE <database name>. |
| | Drop Database | Implement the function of database deletion. Corresponding SQL statement: DROP DATABASE <database name>. |

### 3.2.2 Module/Subsystem 2 Description

**1. Overview**

Finish the functions of table creation, modification and deletion. Implement the table description file creation and update.

**2、 Functions**

| Module Name | Function Name | Function Description |
|---|---|---|
| Table Management | Create Table | Implement database table creation function. Corresponding SQL statement: CREATE TABLE <table name>. |
| | Alter Table | Implement database table modification function. Corresponding SQL statement: ALTER TABLE <table name> <alter table action>. |
| | Drop Table | Implement database table deletion function. Corresponding SQL statement: DROP TABLE <table name>. |

## 3.3 Dependency Description

This project is a windows window and depends on the operating system. The data storage depends on the file management system of the operating system, the communication between the server and client depends on TCP/IP network communication protocol.

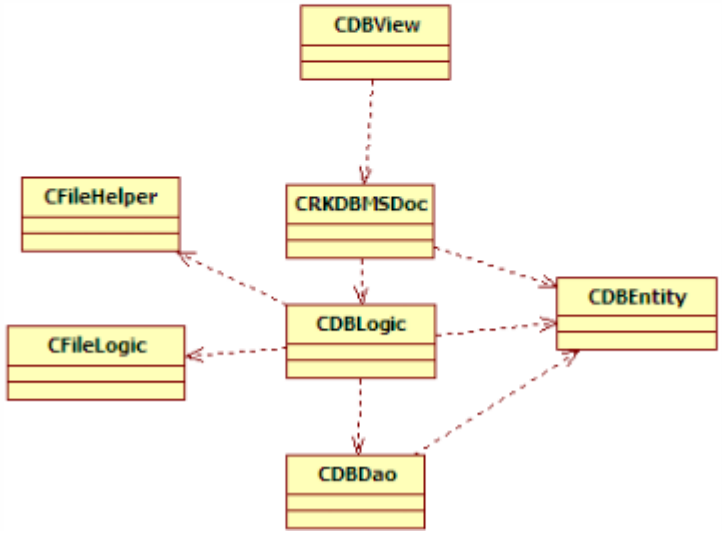
 

# 4 Level 2 Design Description (Optional)

## 4.1 Create Database

By default, the system creates a database named "Ruanko". The basic information of the database is saved in CDBEntity. Display the created database name with tree view CDBView. Through document class CRKDBMSDoc, pass the database information to the view class. Pass the database information to CDBLogic class with CRKDBMSDoc class. CDBLogic class passes the database information to CDBDao class. CDBDao class saves the database information in "Ruanko.db" file.

The class relationship diagram is as follows:




1. CDBView
Display Database Structur

2. CRKDBMSDoc
The document class that
interface display and upda

3. CDBLogic
The database business log
not exist, create the defau

4. CDBDao
Database Information
"Ruanko.db" file in binary

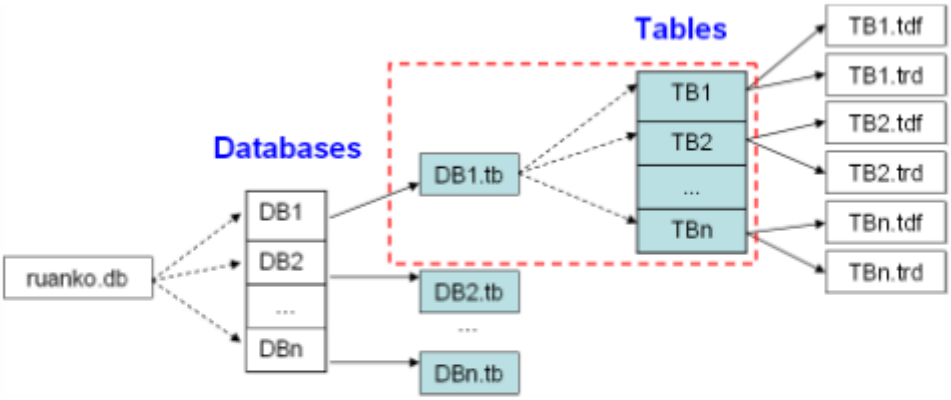
5. CFileLogic
According to the database

6. CFileHleper
Create the folders and the

## 4.2 Module Name 2

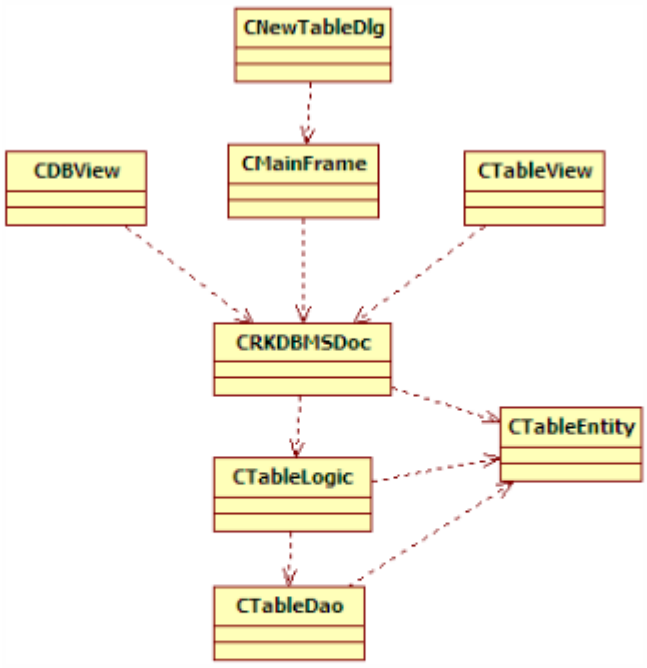
 

According to database name Ruanko, create a database table description file Ruanko.tb to save the table information in the database. By reading table description file (Ruanko.db), we can find the table structure definition file (*.tdf) and the data record file (*.trd).




Save the database information with CDBEntity. Save the table information with CTableEntity. Through CMainFrame class, response Create Table Menu event and display Create Table dialog box. By creating table CNewTableDlg, get the input table name. Pass the input table name to CRKDBMSDoc. CRKDBMSDoc passes the database name and table name to CTableLogic. CTableLogic passes the table file path and table information to CTableDao. CTableDao saves the table information into file in binary format. After the successful save operation, display the table name in CDBView, and display the table structure in CTableView.

The class relationship diagram is as follows:




1. CMainFrame

The frame window class that displays views, and responses Create Table Menu event.

2. CNewTableDlg

Create Table Dialog Box class that gets the input table name through control mapping.

3. CDBView

Display Database Structure View class that displays the table name under the database node of tree view.

4. CTableView

Display Table Structure View class that displays the table structure with List control.

5.CRKDBMSDoc

The document class that stores the database and table information, and controls the interface display and update.

6. CTableLogic

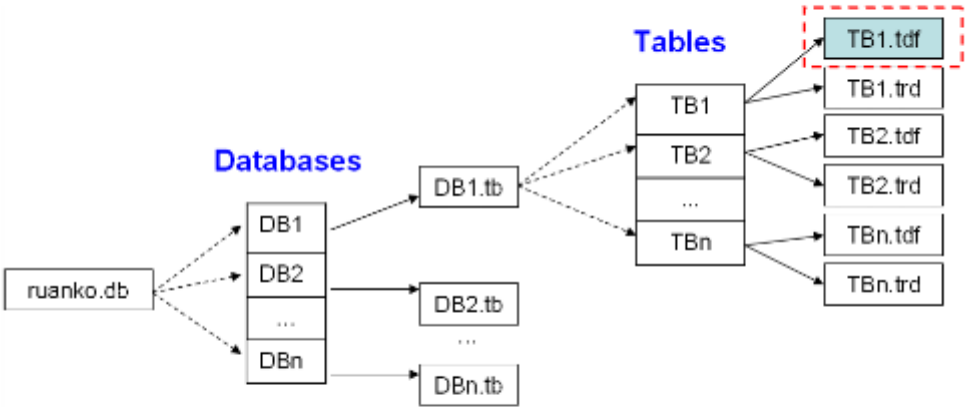
The table business logic class.

7. CTableDao

The table file operation class that saves the table information into file in binary format.

## 4.3 Add Field

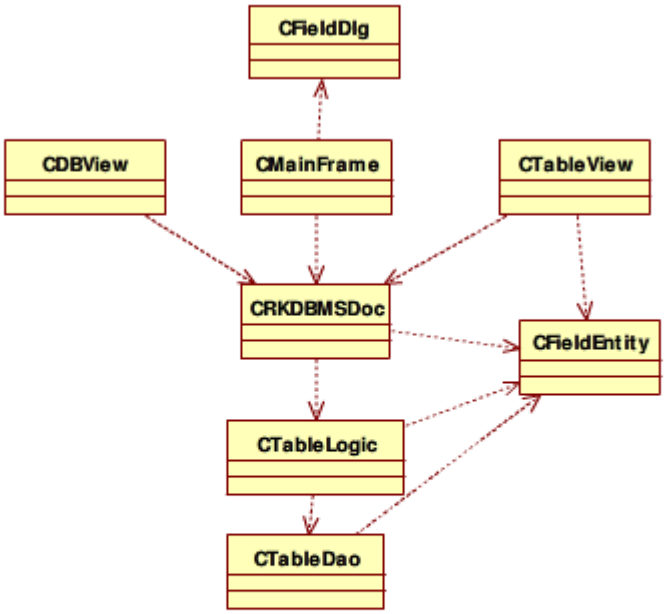
According to the table name, create the table definition file (*.tdf) to save the table structure defined by users. Save the file path into the table description file. By reading the table description file, we can find the table definition file (*.tdf). Save the field information into the table definition file.




CMainFrame class responses Add Field event. In CDBView, the user selects the table. Receive the input field information in CFieldDlg. The field information is saved in CFieldEntity. CRKDBMSDoc passes the database name, table and field to CTableLogic. CTableLogic saves the field information into the table definition file. CDBView displays the field name under the column child node of the table node. CTableView displays the field information.

The class relationship diagram is as follows:




1. CFieldDlg

Add Field Dialog Box class that receives the input field information.

2. CMainFrame

The frame window class that responses Add Field event.

3. CRKDBMSDoc

The document class that saves the database information, table information and field information, and controls the interface display and update.

4. CDBView

The database view class that displays the table name in the database and the field name in the table.

5. CTableView

The table structure view class that displays the field information in the selected table.

6. CTableLogic

The table business logic class that calls the methods of CTableDao class to save the field information into the table definition file (*.tdf), and modify the table description file (*.tb).

7. CTableDao

The table database processing class that saves the field information into the table description file.

8.CFieldEntity

The field information entity class.

# 5 Data Structure/Database Design

*NULL*

# 6 UI Design

*Not available.*

# 7 Detailed Design of Module

## 7.1 DataStructure.h

### 7.1.1 Overview

Database information block

### 7.1.2 Definition

```
struct DatabaseBlock{
    BOOLE type;        // Database type
    VARCHAR name;    // Database name
    VARCHAR filepath;   // Database file path
    DATETIME crtime;    // Creation time
};
```

### 7.1.3 Attributes

*The common data structure should be described here.*

| Visibility | Name | Type | Brief descriptions |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

### 7.1.4 Methods

*Every method should be described.*

#### 7.1.4.1 Method1

(1) Method Descriptions

| | |
|---|---|
| Prototype | |
| Description | |
| Calls | |
| Called By | |
| Input | |
| Output | |
| Return | |
| Exception | |

(2) Implementation Descriptions

*Pseudo codes could be used to depict the method definitions. This is the most important part of LLD.*

#### 7.1.4.2 Method2

## 7.2 Class2 Design