| Project Name | | Confidentiality Level |
|---|---|---|
| Database Management System | | Only for Recipients Reference |
| Project ID | Version | Document Code |
| BP | 1.0 | Project ID_SD_003 |

# Database Management System
# Software System Design Specification

| | | | |
|---|---|---|---|
| Prepared by | | Date | |
| Reviewed by | | Date | |
| Approved by | | Date | |

# Contents

1 Introduction....4
1.1 Purpose....4
1.2 Scope....4
1.2.1 Name....4
1.2.2 Functions....4
2 Level 0 Design Description....5
2.1 Software System Context Definition....5
2.2 Design Considerations....6
2.2.1 Design Alternatives....6
2.2.2 Design Constraints....7
2.2.2.1 Standards compliance....7
2.2.2.2 Hardware Limitations....7
2.2.2.3 Technology Limitations....7
3 Level 1 Design Description....7
3.1 System Architecture....7
3.2 Decomposition Description....9
3.2.1 Database Management....9
3.2.2 Table Management....10
3.2.3 Field Management....10
3.2.4 Data Management....11
3.2.5 Index Management....11
3.2.6 Client Management....12
3.2.7 Transaction Management....12
3.2.8 Integrity Management....12
3.2.9 Database Maintenance....12
3.2.10 Security Management....12
3.3 Dependency Description....12
4 Data Structure Design....12
4.1 Data Type....12
4.2 Integrity....13
4.2.1 Entity Integrity....13
4.2.2 Referential Integrity....13
4.2.3 User-defined Integrity....13
4.3 Database file....13
4.4 Database Description File....15
4.4.1 File Name....15
4.4.2 File Structure....15
4.4.3 Database Information Structure....15
4.5 Table Description File....15
4.5.1 File Name....15
4.5.2 File Structure....15
4.5.3 Table Information Structure....15

4.6 Table Definition File ..... 16
4.6.1 File Name ..... 16
4.6.2 File Structure ..... 16
4.6.3 Field Information Structure ..... 16
4.7 Record File ..... 16
4.7.1 File Name ..... 16
4.7.2 File Strcuture ..... 16
4.7.3 Record Information Structure ..... 17
4.8 Integrity Description File ..... 17
4.8.1 File Name ..... 17
4.8.2 File Structure ..... 17
4.8.3 File Information Structure ..... 17
4.9 Index Description File ..... 17
4.9.1 File Name ..... 17
4.9.2 File Structure ..... 18
4.9.3 Index Information Structure ..... 18
4.9.4 Index data file ..... 18
5 UI Design ..... 19
5.1 Main Interface ..... 19
5.2 Show Table Structure Interface ..... 19
5.3 Select Record Interface ..... 19
5.4 Crate Table Interface ..... 20
5.5 Add Field Interface ..... 20
5.6 Insert Record Interface ..... 21
6 Detailed Design of Module ..... 21
6.1 DataStructure.h ..... 21
6.1.1 Overview ..... 21
6.1.2 DatabaseBlock ..... 22
6.1.2.1 Overview ..... 22
6.1.2.2 Definition ..... 22
6.1.3 TableBlock ..... 22
6.1.3.1 Overview ..... 22
6.1.3.2 Definition ..... 22
6.1.4 FieldBlock ..... 22
6.1.4.1 Overview ..... 22
6.1.4.2 Definition ..... 22
6.2 Global.h file ..... 23
6.2.1 Overview ..... 23
6.2.2 Macro ..... 23
6.3 Entity Class ..... 23
6.3.1 CTableEntity ..... 23
6.4 Other Class ..... 25
7 Error Design ..... 25

# 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 users (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.



# 2 Level 0 Design Description

## 2.1 Software System Context Definition

The architecture of the database management system (DBMS) 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.

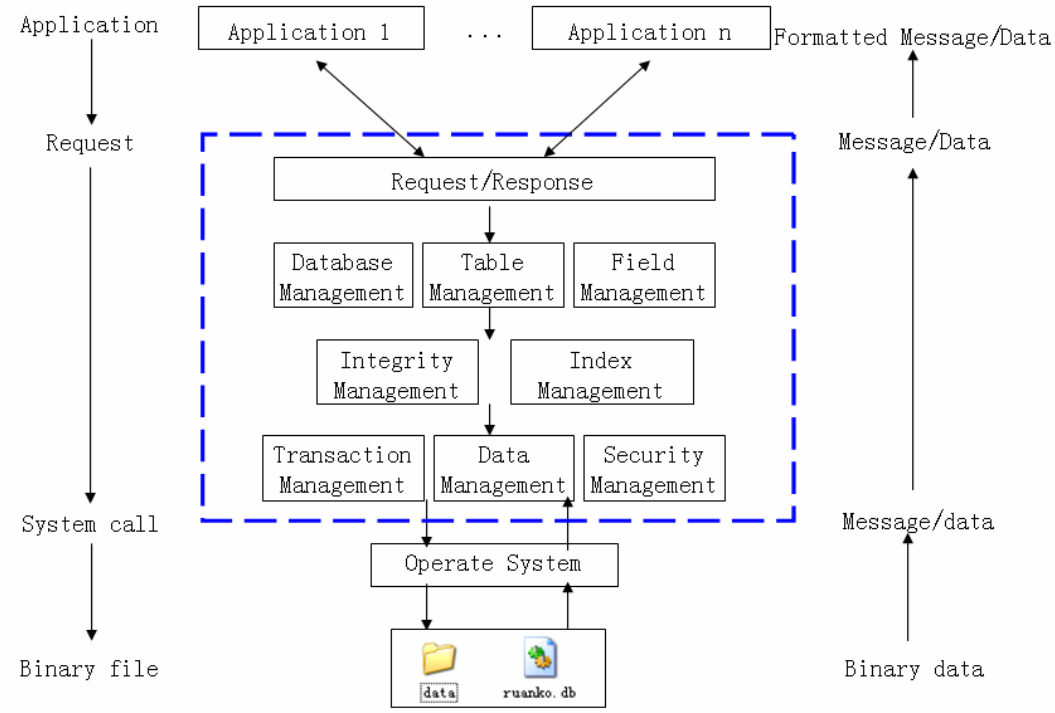



**Figure 2-1 Database management system structure diagram**

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:

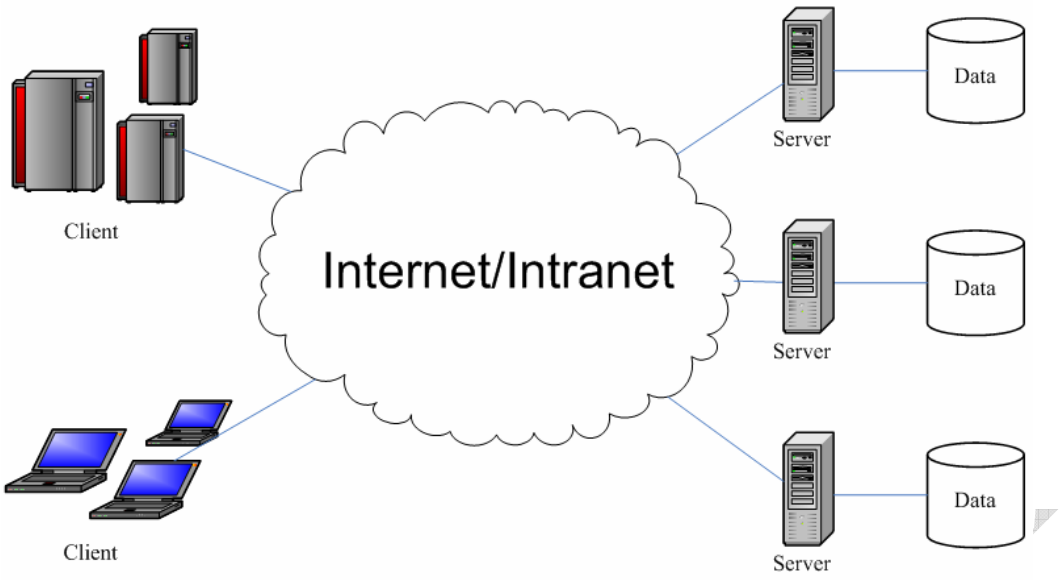
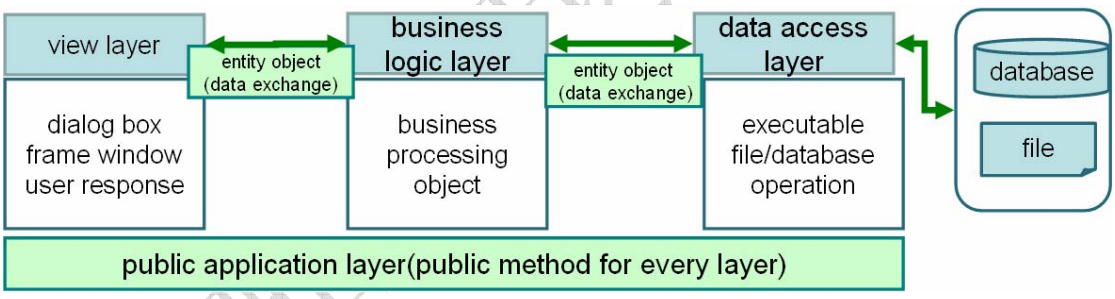



**Figure 2-2 Database Management System Structure**

## 2.2 Design Considerations

### 2.2.1 Design Alternatives

1. Program structure design



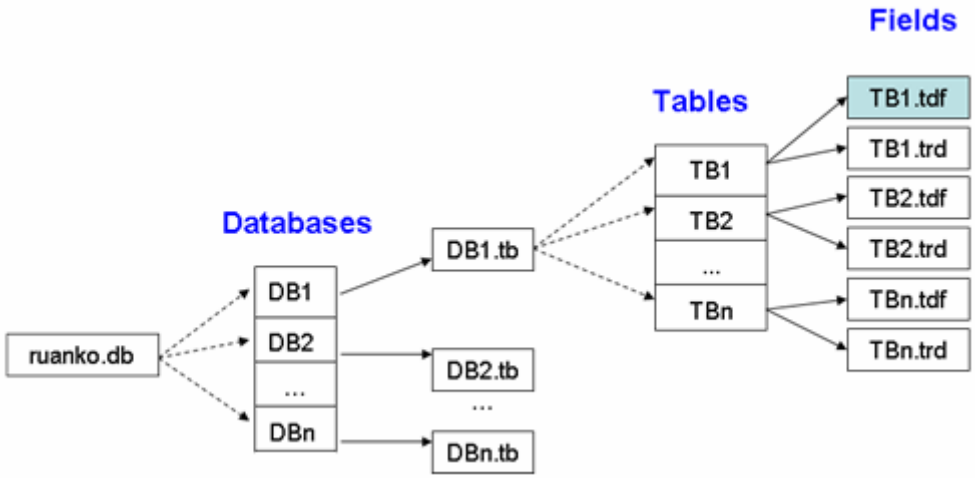

Based on the logic responsibility, the program software structure can be divided into 3 layers: presentation layer, business logic layer and data access layer. Data in each layer is transferred by “entity class” (data object). Besides, public class that irrelative with business may be used in layers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

# 3 Level 1 Design Description

## 3.1 System 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:

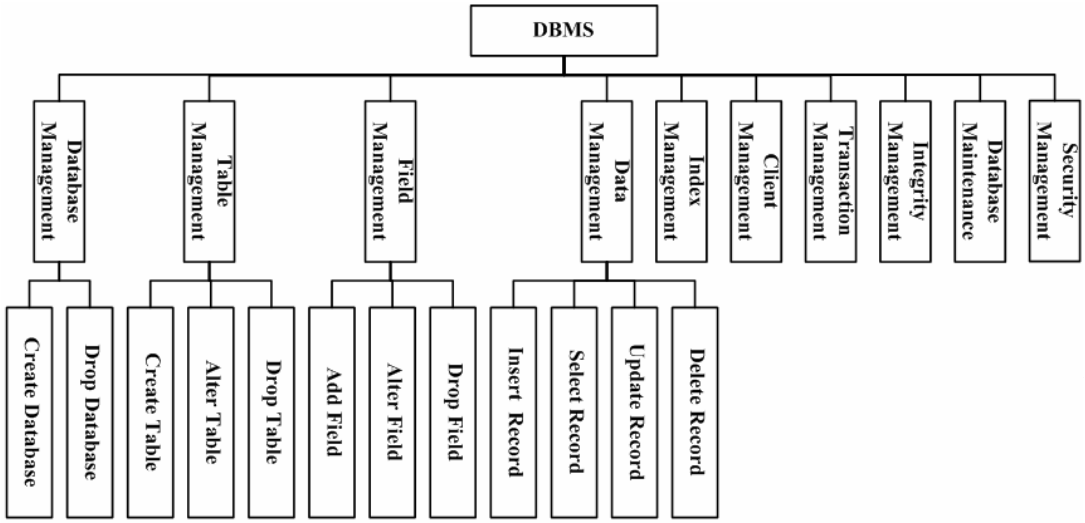



**Figure 3-1 Database management system function structure diagram**

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

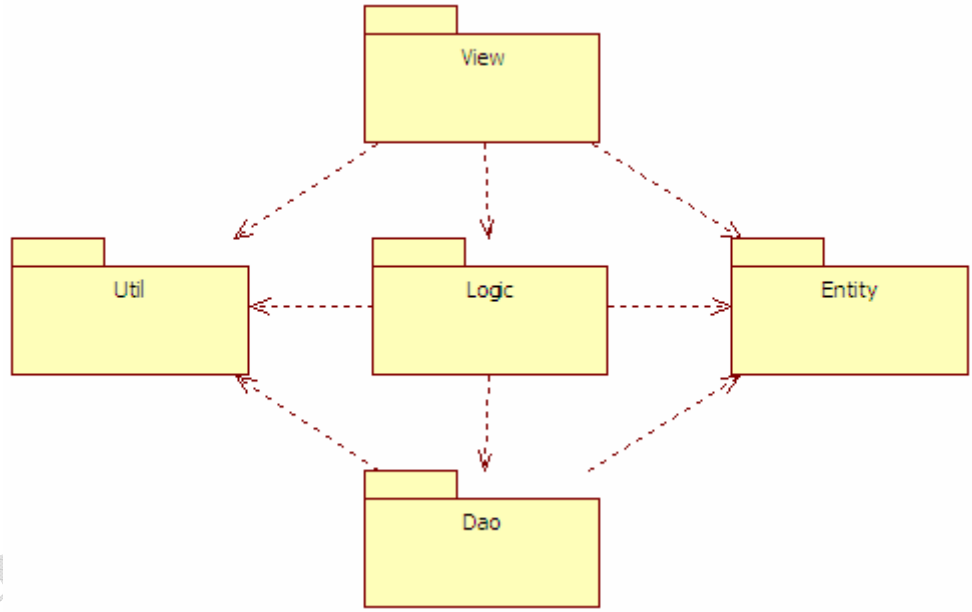



**Figure 3-2Program structure diagram**

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:

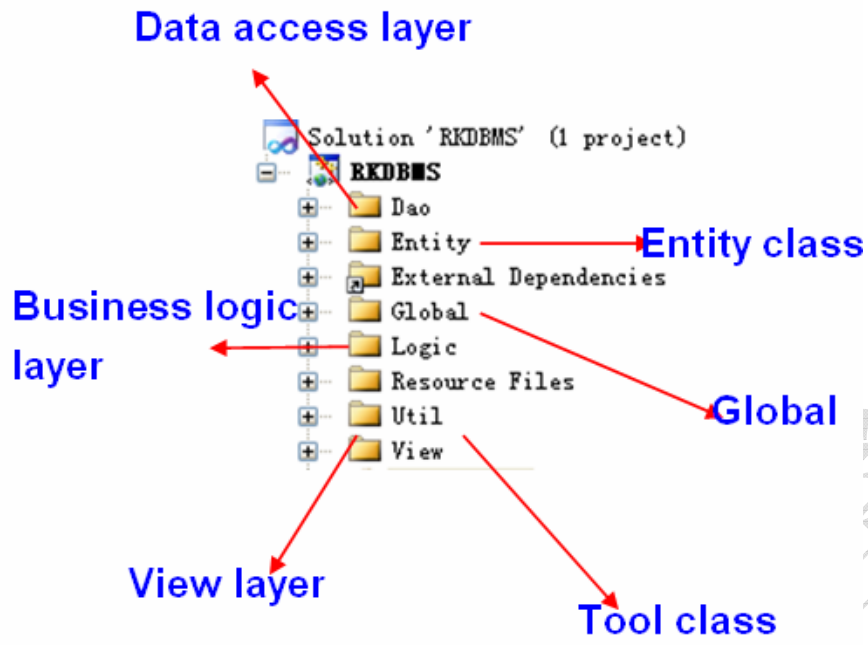



**Figure 3-3 Client project structure diagram**

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

2. Project program structure

Make logic to program by VS2010 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 Table Management

**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>. |

## 3.2.3 Field Management

**1. Overview**

Finish the functions of table field additon, modification and deletion. Implement table field definition file creation, modification and query.

**2. Functions**

| Module | Function Name | Function Description |
|---|---|---|

| Name | | |
|---|---|---|
| Field Management | Add Field | In created table, add fields. Corresponding SQL statement: ALTER TABLE <table name> ADD COLUMN <column name> <column definition>. |
| | Alter Field | Modify the field information in the table. Corresponding SQL statement: ALTER TABLE <table name> MODIFY COLUMN <column name> <alter column action>. |
| | Drop Field | Delete the field in the table. Corresponding SQL statement: ALTER TABLE <table name> DROP COLUMN <column name> <drop behavior>. |

## 3.2.4 Data Management

**1. Overview**

Implement the functions of data storage, update, modification and query.

**2. Functions**

| Module Name | Function Name | Function Description |
|---|---|---|
| Data Management | Insert Record | Insert a record into the database table. Corresponding SQL statement: INSERT INTO <table name> <column name list> VALUES <insert value list>. |
| | Update Record | Update the record in the database table. Corresponding SQL statement is: UPDATE <table name> SET <column name> = <update value> [ WHERE <search condition> ]. |
| | Select Record | Query all the records in the table. Corresponding SQL statement is: SELECT * FROM <table name> [ WHERE <search condition> ]. |
| | Delete Record | Delete the record in the table. Corresponding SQL statement is: DELETE FROM <table name> [ WHERE <search condition> ]. |

## 3.2.5 Index Management

Establish the index for the key field in the database table. In the data operation, optimize the query with the index.

### 3.2.6 Client Management

Implement the client-server structure. The client can connect to main servers. The server can provide service for many clients.

### 3.2.7 Transaction Management

Implement transaction management function in the database.

### 3.2.8 Integrity Management

Implement the functions of database integrity constraint check and management.

### 3.2.9 Database Maintenance

Implement the functions of database backup and recovery.

### 3.2.10 Security Management

Implement user management, permission management.

## 3.3 Dependency Description

This project is a Windows window program 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 Data Structure Design

## 4.1 Data Type

| System data type | Description | Size | Program data type |
|---|---|---|---|
| INTEGER | Integer type | 4byte | int |
| BOOL | Boolean type | 1byte | bool |

| | | | |
|---|---|---|---|
| DOUBLE | Float type | 2byte | double |
| VARCHAR(n) | String type, maximum length is 255, ended with "\0" to mark the end of a string. | (n+1)byte | char[n+1] |
| DATETIME | Data time type | 16byte | SYSTEMTIME |

## 4.2 Integrity

### 4.2.1 Entity Integrity

PRIMARY KEY

### 4.2.2 Referential Integrity

FOREIGN KEY

### 4.2.3 User-defined Integrity

1. CHECK
2. UNIQUE
3. NOT NULL
4. DEFAULT
5. IDENTITY

## 4.3 Database file

This system is a relational database management system. It stores data with the binary file.

1. File Design

The files in DBMS are mainly divided into two types: data definition file and data file

(1) Data definition file: saves the definitions of various objects in DBMS.

(2) Data file: saves various data in DBMS.

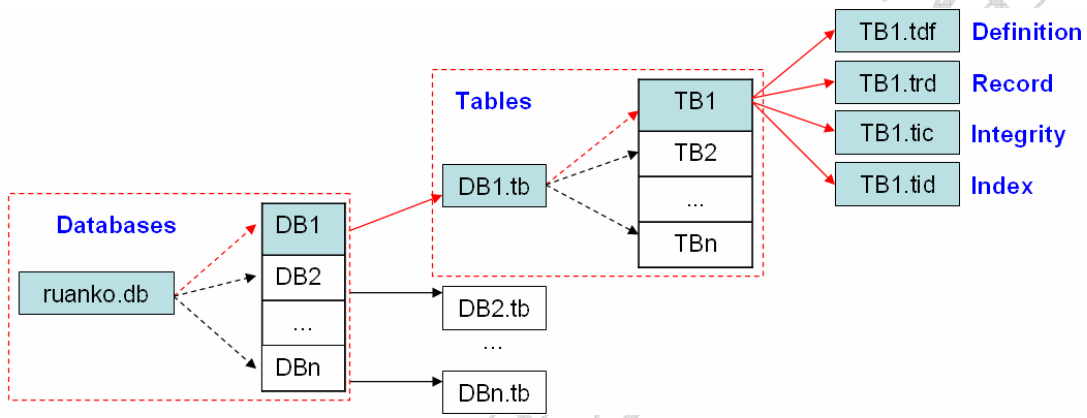
| Type | File | Name | Remark |
|---|---|---|---|
| Data definition file | Database description file | ruanko.db | Save the database information. |
| | Table description file | *.tb | Save the table information. |
| | Table definition file | *.tdf | Save the field |

| | | | |
|---|---|---|---|
| | | | information. |
| | Integrity description file | *.tic | Save the integrity constraints. |
| | Index description file | *.tid | Save the definition of the index. |
| Data File | Record File | *.trd | |
| | Index data file | *.ix | |
| | Log file | *.log | |

2. File Structure

The database management system supports multi-database. A database can include many tables. A table includes the data of the table definition, integrity constraint, index, record and so on.




3. Directory Structure

Taking [DBMS_ROOT] as the root directory, each database creates a folder to save various files in the database. Path :[DBMS_ROOT]\data\DB_NAME\.

Example: create a “Ruanko” database, and create "Student“ table in the database. The presented directory structure is as follows:

| File | Path |
|---|---|
| Database description file | [DBMS_ROOT]\ruanko.db |
| Table description file | [DBMS_ROOT]\data\Ruanko\Ruanko.tb |
| Table definition file | [DBMS_ROOT]\data\Ruanko\Student.tdf |
| Record file | [DBMS_ROOT]\data\Ruanko\Student.trd |
| Integrity description file | [DBMS_ROOT]\data\Ruanko\Student.tic |
| Index description file | [DBMS_ROOT]\data\Ruanko\Student.tid |

## 4.4 Database Description File

### 4.4.1 File Name

ruanko.db

### 4.4.2 File Structure

| | | | |
|---|---|---|---|
| DatabaseBlock 1 | DatabaseBlock 2 | …… | DatabaseBlock N |

### 4.4.3 Database Information Structure

| Structure Member | Data type | Description |
|---|---|---|
| name | CHAR[128] | Database Name |
| type | BOOL | Database type |
| filename | CHAR[256] | The database data file path |
| crtime | DATETIME | Creation time |

## 4.5 Table Description File

### 4.5.1 File Name

[Database_Name].tb

### 4.5.2 File Structure

| | | | |
|---|---|---|---|
| TableBlock 1 | TableBlock 2 | …… | TableBlock N |

### 4.5.3 Table Information Structure

| Structure Member | Data type | Description |
|---|---|---|
| name | CHAR[128] | Table Name |
| record_num | INTERGER | Records number |
| field_num | INTERGER | Fields number |
| tdf | CHAR[256] | The path of Table definition file |
| tic | CHAR[256] | The path of Integrity description file |
| trd | CHAR[256] | The path of Record File |

| | | |
|---|---|---|
| tid | CHAR[256] | The path of Index data file |
| crtime | DATETIME | Table creation time |
| mtime | INTERGER | Last modification time |

## 4.6 Table Definition File

### 4.6.1 File Name

[Table_Name].tdf

### 4.6.2 File Structure

| |
|---|
| FieldBlock 1 |
| FieldBlock 2 |
| ..... |
| FieldBlock N |

### 4.6.3 Field Information Structure

| Structure Member | Data type | Description |
|---|---|---|
| order | INTERGER | Field order |
| name | CHAR[128] | Field name |
| type | INTERGER | Field Type |
| param | INTERGER | Field type parameter |
| mtime | DATETIME | Last modification time |
| integrities | INTERGER | Integrity constraints |

## 4.7 Record File

### 4.7.1 File Name

[Table_Name].trd

### 4.7.2 File Strcuture

| | | | |
|---|---|---|---|
| Record 1 | Record 2 | …… | Record N |

### 4.7.3 Record Information Structure

1. In DBMS, a record store format by user-defined.
2. Based on the characteristics of data storage, all of the blocks and the field size are stored in the adjustment of a multiple of 4, in order to improve the efficiency of the data read.

## 4.8 Integrity Description File

### 4.8.1 File Name

[Table_name].tic

### 4.8.2 File Structure

| | | | |
|---|---|---|---|
| Integrity 1 | Integrity 2 | …… | Integrity N |

### 4.8.3 File Information Structure

| Structure Member | Data type | Description |
|---|---|---|
| name | CHAR[128] | Integrity Name |
| field | CHAR[128] | Field Name |
| type | INTERGER | Type |
| param | CHAR[256] | parameter |

## 4.9 Index Description File

### 4.9.1 File Name

[Table_Name].tid

### 4.9.2 File Structure

| | | | |
|---|---|---|---|
| IndexBlock 1 | IndexBlock 2 | …… | IndexBlock N |

### 4.9.3 Index Information Structure

| Structure Member | Data type | Description |
|---|---|---|
| name | CHAR[128] | Name |
| unique | BOOLE | Unique index |
| asc | BOOLE | Order Type |
| field_num | INTEGER | Fields number |
| fields | CHAR[128][2] | Field value |
| record_file | CHAR[256] | The path of index record file |
| index_file | CHAR[256] | The path index data file |

### 4.9.4 Index data file

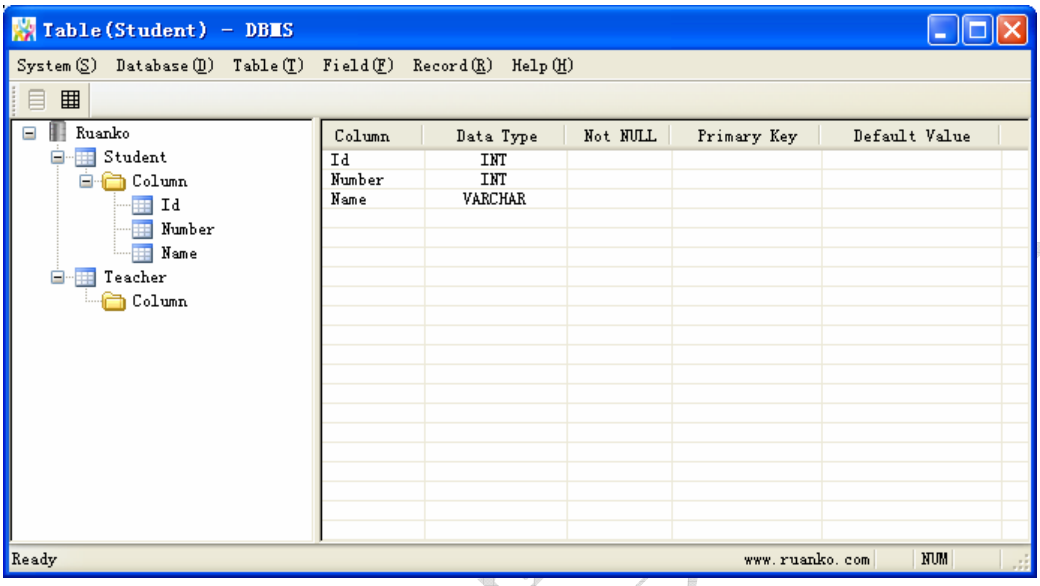
File Name: [index_name].ix
Folder: Table folder
Index Name: [Field_Name]Index



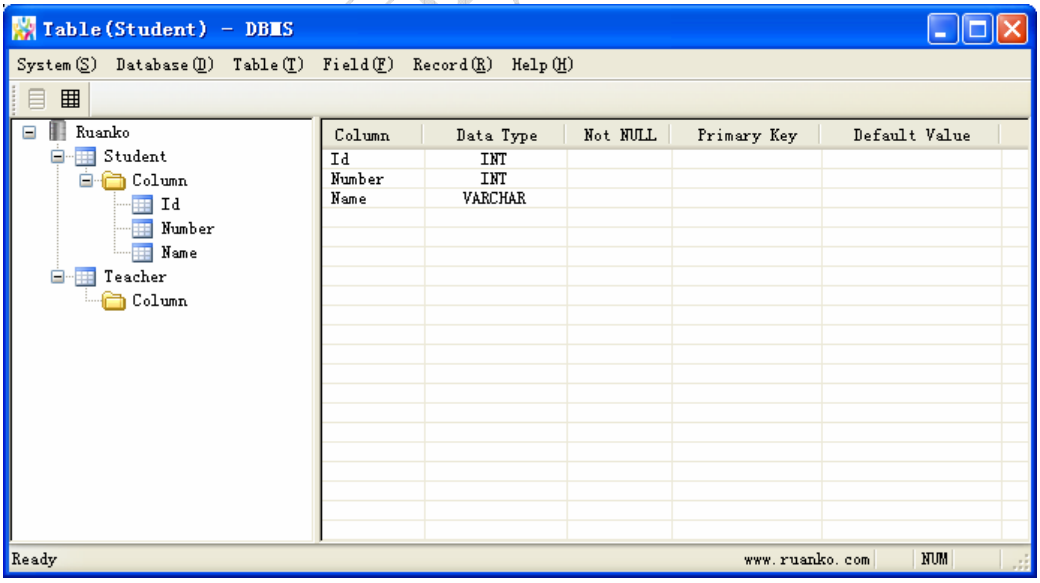
 

# 5 UI Design

## 5.1 Main Interface

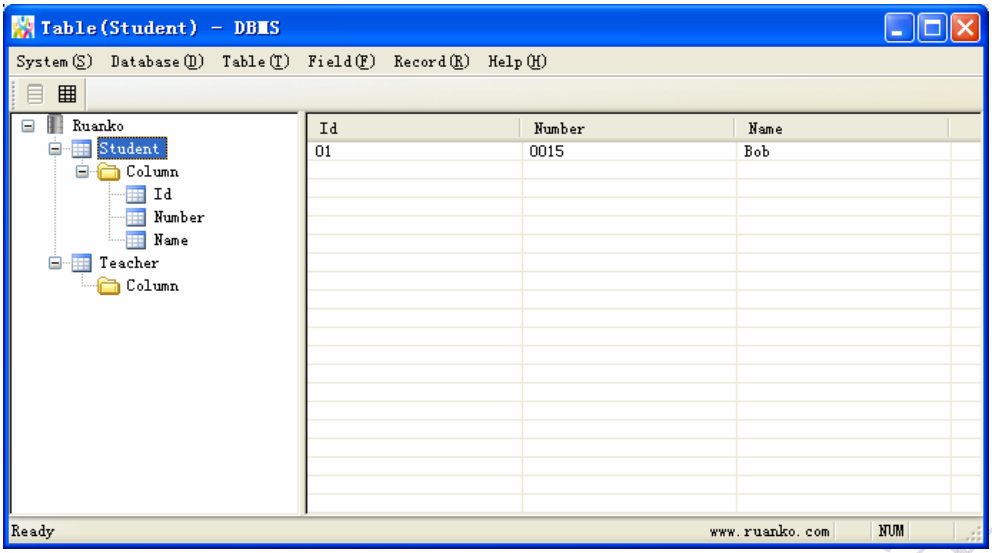



## 5.2 Show Table Structure Interface

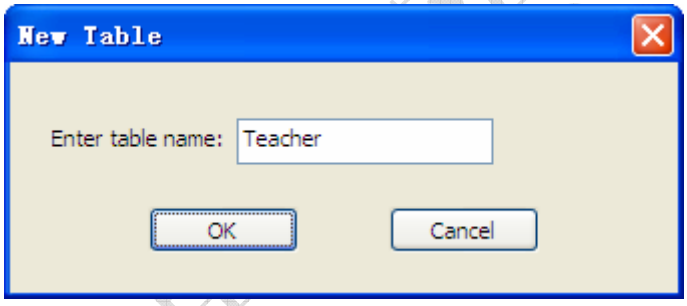



## 5.3 Select Record Interface

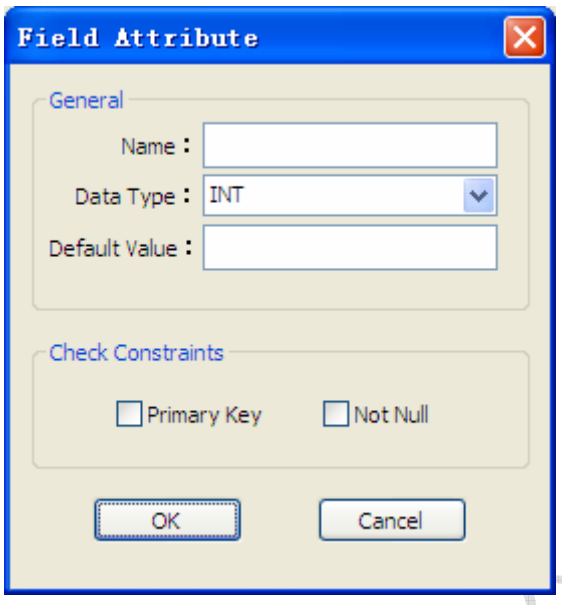



## 5.4 Crate Table Interface

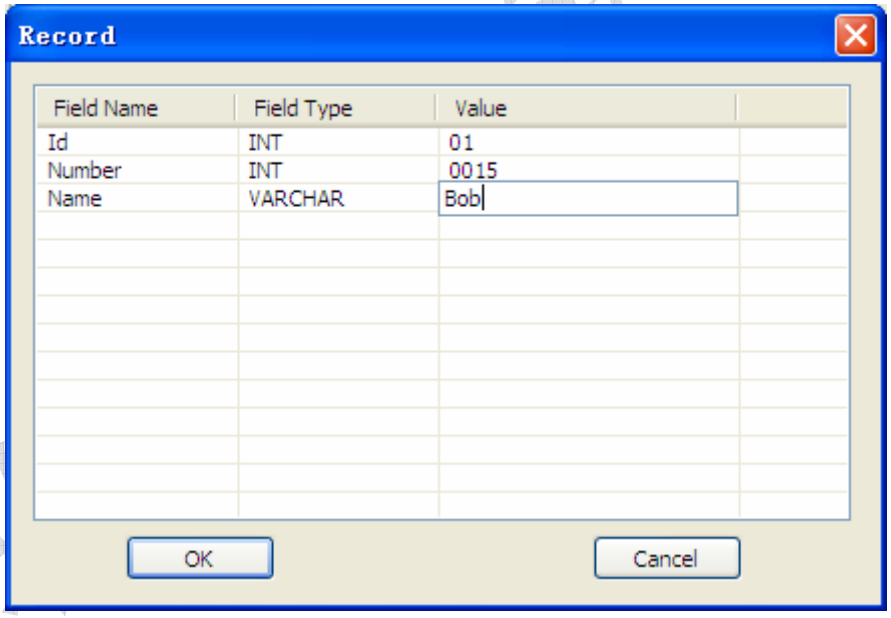



## 5.5 Add Field Interface




## 5.6 Insert Record Interface




# 6 Detailed Design of Module

## 6.1 DataStructure.h

### 6.1.1 Overview

Define the data type of the database management system.

## 6.1.2 DatabaseBlock

### 6.1.2.1 Overview

Database information block.

### 6.1.2.2 Definition

```
struct DatabaseBlock{
	BOOLE type;
	VARCHAR name;
	VARCHAR filepath;
	DATETIME crtime;
};
```

## 6.1.3 TableBlock

### 6.1.3.1 Overview

Table information block.

### 6.1.3.2 Definition

```
struct TableBlock
{
	VARCHAR name;
	INTEGER record_num;
	INTEGER field_num;
	VARCHAR tdf;
	VARCHAR trd;
	DATETIME crtime;
	DATETIME mtime;
};
```

## 6.1.4 FieldBlock

### 6.1.4.1 Overview

Field information block.

### 6.1.4.2 Definition

```
struct FieldBlock
```

```
{
    VARCHAR name;
    INTEGER type;
    INTEGER param;
    DATETIME mtime;
    INTEGER integrities;
};
```

## 6.2 Global.h file

### 6.2.1 Overview

Macro definition file.

### 6.2.2 Macro

| Macro Name | Value | Description |
|---|---|---|
| UPDATE_OPEN_DATABASE | 0x01 | Open database message |
| UPDATE_CREATE_TABLE | 0x02 | Create table message |
| UPDATE_EDIT_TABLE | 0x03 | Alter table message |
| UPDATE_ADD_FIELD | 0x04 | Add field message |
| UPDATE_OPEN_TABLE | 0x05 | Select record message |
| UPDATE_INSERT_RECORD | 0x06 | Insert record message |
| MENU_DATABASE | 1 | Database menu number |
| MENU_TABLE | 2 | Table menu number |
| MENU_FIELD | 3 | Field menu number |
| MENU_RECORD | 4 | Record menu number |
| MENU_OTHER | -1 | Other menu number |

## 6.3 Entity Class

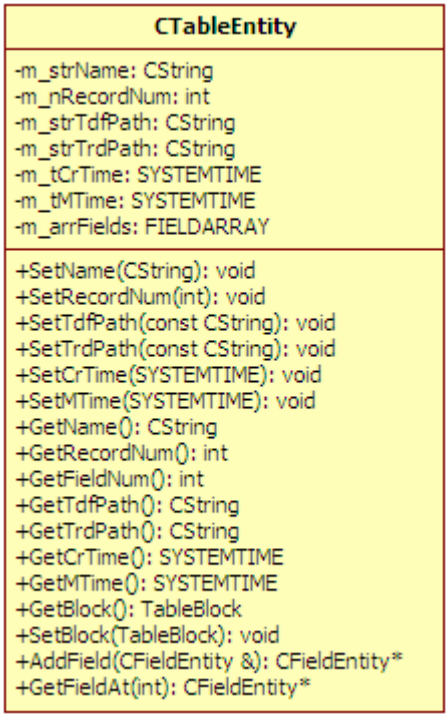
### 6.3.1 CTableEntity

**1. Overview**

The entity class of table information. Save the table information. Data in each layer is transferred by entity class object.

**2. Class Diagram**




**3. Attributes**

| Visibility | Name | Type | Brief descriptions |
|---|---|---|---|
| private | m_strName | CString | Table name |
| | m_nRecordNum | int | Records number |
| | m_strTdfPath | CString | The path of table definition file |
| | m_strTrdPath | CString | The path of record file |
| | m_tCrTime | SYSTEMTIME | Table creation time |
| | m_tMTime | SYSTEMTIME | Last modification time |

**4. Methods**

| Visibility | Prototype | Description |
|---|---|---|
| public | TableBlock GetBlock(); | Save table information to a TableBlock structure |
| | void SetBlock(TableBlock tb) | Use the data in a table information structure assignment for the data members |
| | void SetName(CString strName) | Set table name |
| | void SetRecordNum(int nNum) | Set records number |
| | void SetTdfPath(const CString strTdfPath) | Set the path of table definition file |
| | void SetTrdPath(const CString strTrdPath) | Set the path of record file |
| | void SetCrTime(SYSTEMTIME tTime) | Set crate table time |

| | | |
|---|---|---|
| | void SetMTime(SYSTEMTIME tTime) | Set last modification time |
| | CString GetName() | Get table name |
| | int GetRecordNum() | Get records number |
| | int GetFieldNum() | Get fields number |
| | CString GetTdfPath(); | Get the path of table definition file |
| | CString GetTrdPath(); | get the path of record file |
| | SYSTEMTIME GetCrTime(); | Get crate table time |
| | SYSTEMTIME GetMTime(); | Get last modification time |

## 6.4 Other Class

Other class design is the same as CTableEntity.

# 7 Error Design

For the program, define unified exception class CAppException, and process the exception with try, catch, throw. When the underlying exception appears, throw the exception class to higher layer, and include the exception information. The exception structure is shown below: