DBMS Software System Design Specification

Revision Record

| Date | Revision Version | CR ID /Defect ID | Sec No. | Change Description | Author |
| --- | --- | --- | --- | --- | --- |
| 2017/4/28 | 1.00 | no | 1001 | Add Introduction and level 0 | 刘彤 |
| 2017/4/29 | 1.01 | no | 1002 | Add level1 | 刘彤 |
| 2017/4/30 | 1.02 | no | 1003 | Add database design | 刘彤 |
| 2017/5/1 | 1.03 | no | 1004 | Add UI design | 刘彤 |
| 2017/5/2 | 1.04 | no | 1005 | add Design of Module | 刘彤 |

**Content**

[1 Introduction 5](#_Toc385599054)

[1.1 Purpose 5](#_Toc385599055)

[1.2 Scope 5](#_Toc385599056)

[1.2.1 Name 5](#_Toc385599057)

[1.2.2 Functions 5](#_Toc385599058)

[1.2.3 Applications 5](#_Toc385599059)

[2 Level 0 Design Description 5](#_Toc385599060)

[2.1 Software System Context Definition 5](#_Toc385599061)

[2.2 Design Considerations 5](#_Toc385599062)

[2.2.1 Design Alternatives 5](#_Toc385599063)

[2.2.2 Design Constraints 5](#_Toc385599064)

[2.2.3 Other Design Considerations 6](#_Toc385599065)

[3 Level 1 Design Description 6](#_Toc385599066)

[3.1 System Architecture 6](#_Toc385599067)

[3.1.1 Description of the Architecture 6](#_Toc385599068)

[3.1.2 Representation of the Business Flow 6](#_Toc385599069)

[3.2 Decomposition Description 6](#_Toc385599070)

[3.2.1 Module/Subsystem 1 Description 6](#_Toc385599071)

[3.2.2 Module/Subsystem 2 Description 6](#_Toc385599072)

[3.3 Dependency Description 6](#_Toc385599073)

[4 Data Structure/Database Design 7](#_Toc385599081)

[5 UI Design 7](#_Toc385599082)

[6 Detailed Design of Module 8](#_Toc385599083)

[6.1 Class1 Design 8](#_Toc385599084)

[6.1.1 Overview 8](#_Toc385599085)

[6.1.2 Class Diagram 8](#_Toc385599086)

[6.1.3 Attributes 8](#_Toc385599087)

[6.1.4 Methods 8](#_Toc385599088)

[6.2 Class2 Design 9](#_Toc385599089)

Keywords: database, architecture, attribute, design

Abstract: Select, update, insert and delete the data for the database.

List of abbreviations:

| Abbreviations | Full spelling | Chinese explanation |
| --- | --- | --- |
| DB | Database | 数据库 |
| UI | User Interface | 用户界面 |

1. Introduction

## 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.).

## Scope

* + 1. Name

Database Management System.

* + 1. Functions

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

* + 1. Applications

Select, update, insert and delete the data for the database.

1. Level 0 Design Description

## 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. 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.

## Design Considerations

* + 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.

* + 1. Design Constraints
       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>.

* + - 1. 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.

* + 1. Other Design Considerations

.

1. Level 1 Design Description

## 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.

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

1. Project program structure Make logic to program by VS2015 Solution Explorer through Filter.

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

.

## Decomposition Description

* + 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>.* |

* + 1. 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>. |

## 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.

1. Level 2 Data Structure Design

## 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 |

## Integrity

* + 1. Entity Integrity

PRIMARY KEY

* + 1. Function Illustration

FOREIGN KEY

* + 1. User-defined Integrity

1. CHECK

2. UNIQUE

3. NOT NULL

4. DEFAULT

5. IDENTITY

1. UI Design
2. Detailed Design of Module

## Entity Class

* + 1. CTableEntity

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

* + 1. 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 |

* + 1. Methods
       1. Method1

1. Method Descriptions

|  |  |  |
| --- | --- | --- |
| 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 |

## Other Class

Other class design is the same as CTableEntity.