智慧养老数据交换软件V1.0

详细设计说明书

二〇一八年六月二日

目录

[一、 引言 5](#_Toc31654)

[1.1 编写目的 5](#_Toc10078)

[1.2 项目背景 6](#_Toc3885)

[1.3 使用范围 6](#_Toc27723)

[1.4 定义 7](#_Toc22269)

[1.5 参考资料 9](#_Toc4532)

[二、 软件与其他系统的关系 9](#_Toc11864)

[2.1 系统整体概况 9](#_Toc31366)

[2.2 接口系统概况 10](#_Toc22110)

[2.3 软件与接口系统关系 10](#_Toc19870)

[三、 软件总体设计 11](#_Toc9140)

[3.1 软件需求概括 11](#_Toc6501)

[3.2 定义 11](#_Toc3623)

[3.3 需求概述 11](#_Toc13651)

[3.3.1 功能性需求分析 11](#_Toc23212)

[3.3.2 非功能性需求分析 12](#_Toc22650)

[3.4 条件与限制 13](#_Toc6706)

[3.4.1 研发硬件配置： 13](#_Toc21903)

[3.4.2 客户使用配置 14](#_Toc19481)

[3.4.3 研发语言及编译器 14](#_Toc12294)

[3.4.4 软件支持工具 14](#_Toc10575)

[3.5 软件的设计思路与原则 14](#_Toc6969)

[四、 数据库驱动兼容系统设计 15](#_Toc1372)

[4.1 设计原则 16](#_Toc15292)

[4.2 数据库驱动兼容概念设计 16](#_Toc23793)

[4.3 数据库兼容驱动结构设计 17](#_Toc1092)

[4.4 总体结构和模块接口设计 17](#_Toc9643)

[4.5 模块功能逻辑关系 18](#_Toc4808)

[4.6 设计和描述 19](#_Toc20775)

[4.7 软件流程图 19](#_Toc22483)

[五、 软件函数及结构描述 21](#_Toc17832)

[5.1 func main() 21](#_Toc24848)

[5.2 func webserver() 21](#_Toc24782)

[5.3 func netsocket() 21](#_Toc10656)

[5.4 func dbblock() 21](#_Toc32056)

[5.5 func (web \*WEB)listen(conn \*net.Conn) 21](#_Toc9658)

[5.6 func (web \*WEB)recv(conn \*net.Conn) 21](#_Toc31238)

[5.7 func(web \*WEB)send(conn \*net.Conn) 22](#_Toc11422)

[5.8 func(web \*WEB)stop(conn \*net.Conn) 22](#_Toc31634)

[5.9 func (d \*Dialer) deadline(ctx context.Context, now time.Time) (earliest time.Time) 22](#_Toc15086)

[5.10 func (d \*Dialer) resolver() \*Resolver 22](#_Toc5858)

[5.11 func parseNetwork(ctx context.Context, network string, needsProto bool) (afnet string, proto int, err error) 22](#_Toc6365)

[5.12 func (r \*Resolver) resolveAddrList(ctx context.Context, op, network, addr string, hint Addr) (addrList, error) 22](#_Toc26006)

[5.13 func Dial(network, address string) (Conn, error) 23](#_Toc2781)

[5.14 func (d \*Dialer) Dial(network, address string) (Conn, error) 23](#_Toc11876)

[5.15 func (d \*Dialer) DialContext(ctx context.Context, network, address string) 23](#_Toc10414)

[5.16 func dialParallel(ctx context.Context, dp \*dialParam, primaries, fallbacks addrList) (Conn, error) 23](#_Toc28257)

[5.17 func dialSingle(ctx context.Context, dp \*dialParam, ra Addr) (c Conn, err error) 24](#_Toc5634)

[5.18 func (k \*contextKey) String() string { return "net/http context value " + k.name } 24](#_Toc12712)

[5.19 func hasPort(s string) bool { return strings.LastIndex(s, ":") > strings.LastIndex(s, "]") } 24](#_Toc28937)

[5.20 func isASCII(s string) bool 24](#_Toc24094)

[5.21 func refererForURL(lastReq, newReq \*url.URL) string 24](#_Toc11943)

[5.22 func (c \*Client) send(req \*Request, deadline time.Time) (resp \*Response, didTimeout func() bool, err error) 25](#_Toc26975)

[5.23 func send(ireq \*Request, rt RoundTripper, deadline time.Time) (resp \*Response, didTimeout func() bool, err error) 25](#_Toc6443)

[5.24 func setRequestCancel(req \*Request, rt RoundTripper, deadline time.Time) (stopTimer func(), didTimeout func() bool) 25](#_Toc31518)

[5.25 func basicAuth(username, password string) string 25](#_Toc3500)

[5.26 func Get(url string) (resp \*Response, err error) 26](#_Toc18303)

[5.27 func (c \*Client) checkRedirect(req \*Request, via []\*Request) error 26](#_Toc13699)

[5.28 func (c \*Client) makeHeadersCopier(ireq \*Request) func(\*Request) 26](#_Toc2348)

[5.29 func Post(url string, contentType string, body io.Reader) (resp \*Response, err error) 26](#_Toc9348)

[5.30 func (c \*Client) Post(url string, contentType string, body io.Reader) (resp \*Response, err error) 26](#_Toc313)

[5.31 func PostForm(url string, data url.Values) (resp \*Response, err error) 26](#_Toc18895)

[5.32 func (c \*Client) PostForm(url string, data url.Values) (resp \*Response, err error) 27](#_Toc21274)

[5.33 func dbinit() 27](#_Toc20608)

[5.34 func (ns \*NullString) Scan(value interface{}) error 27](#_Toc1324)

[5.35 func (ns NullString) Value() (driver.Value, error) 27](#_Toc13955)

[5.36 func (n NullInt64) Value() (driver.Value, error) 27](#_Toc29344)

[5.37 func (db \*DB) maxIdleConnsLocked() int 27](#_Toc21328)

[5.38 func (db \*DB) SetMaxIdleConns(n int) 27](#_Toc8664)

[5.39 func (db \*DB) connectionCleaner(d time.Duration) 27](#_Toc5482)

[5.40 func (db \*DB) Stats() DBStats 28](#_Toc19047)

[5.41 func (db \*DB) maybeOpenNewConnections() 28](#_Toc13511)

[5.42 func (db \*DB) connectionOpener() 28](#_Toc28757)

[5.43 func (db \*DB) openNewConnection() 28](#_Toc18831)

[5.44 func (db \*DB) nextRequestKeyLocked() uint64 28](#_Toc26208)

[5.45 func (db \*DB) conn(ctx context.Context, strategy connReuseStrategy) 28](#_Toc29150)

[5.46 func (db \*DB) putConn(dc \*driverConn, err error) 28](#_Toc22340)

[5.47 func (db \*DB) putConnDBLocked(dc \*driverConn, err error) bool 28](#_Toc18014)

[5.48 func Marshal(v interface{}) ([]byte, error) 29](#_Toc13914)

[5.49 func Unmarshal(data []byte, v interface{}) error 29](#_Toc7420)

# 引言

## 编写目的

编写详细设计说明书是软件开发过程必不可少的部分，智慧养老数据交换软件详细设计是设计的第二阶段，这个阶段的主要任务是在智慧养老数据交换软件概要设计基础上，对概要设计中产生的功能模块进行过程描述，设计功能模块的内部细节，为编写源代码提供必要的说明。详细设计说明书同时也是为了更好得让项目组人员了解项目开发的信息。

## 项目背景

软件名称：智慧养老数据交换软件

软件开发平台：Go语言(Linux系统，兼容windows系统)

委托单位：

承担单位：

区“智慧养老”一体化综合服务管理信息平台由智慧养老综合数据中心、养老服务分析及监管系统、综合为老服务信息系统等组成，实行区级、街道级及社区为老服务机构三级联动，形成综合养老服务管理体系。

通过该项目建设，政府可以建立为老服务专项资金台账，并对养老工作进行统一监管评价，建设服务机构诚信体系，为政府智慧养老工作和统筹合理匹配养老资源提供科学、有效的数据支撑。社区老人可通过网页、移动终端、微信等，访问区为老服务系统，接收社区为老年提供的各类为老服务，并进行老年人间自助、互助的管理。

本软件的提出是为了结合各地社区已有的数据库系统，有效地提取采集到的智慧养老系统平台，并根据应用需要按一定的格式存入平台数据库。本软件拥有多协程并发处理数据，使接收数据和存储数据高速同步运行； Go语言开发，跨平台设计，使软件以二进制本地代码方式运行在各种操作系统中；无界面设计，充分利用系统资源，运行方便，调试容易等技术特点，大大降低错误率，节约时间和人力成本，迎合系统平台需求。因此，有较好的应用前景与商业价值。该软件运行于Linux系统平台。由于go语言的兼容性的特点，所以windows平台也能够很好的运行。

## 使用范围

本软件组的技术开发人员；对于部分软件使用人员可以了解软件内部的运行机制。

## 定义

数据库：数据库(Database)是按照数据结构来组织、存储和管理数据的建立在计算机存储设备上的仓库。  
 简单来说是本身可视为电子化的文件柜——存储电子文件的处所，用户可以对文件中的数据进行新增、截取、更新、删除等操作。  
 在经济管理的日常工作中，常常需要把某些相关的数据放进这样的“仓库”，并根据管理的需要进行相应的处理。  
 举例如下，企业或事业单位的人事部门常常要把本单位职工的基本情况(职工号、姓名、年龄、性别、籍贯、工资、简历等)存放在表中，这张表就可以看成是一个数据库。有了这个"数据仓库"我们就可以根据需要随时查询某职工的基本情况，也可以查询工资在某个范围内的职工人数等等。这些工作如果都能在计算机上自动进行，那我们的人事管理就可以达到极高的水平。此外，在财务管理、仓库管理、生产管理中也需要建立众多的这种"数据库"，使其可以利用计算机实现财务、仓库、生产的自动化管理。  
 目前在各社区使用的老人数据大都使用oracle系统，有部分使用mysql以及微软的sql server，所以需要统一在系统平台的大数据新sql数据库中，必须使用统一的接口来兼容所有的其他数据库。

新SQL语言：NewSQL系统虽然在的内部结构变化很大，但是它们有两个显着的共同特点：(1)它们都支持关系数据模型，(2) 它们都使用SQL作为其主要的接口。已知的第一个NewSQL系统叫做H-Store，它是一个分布式并行内存数据库系统。目前NewSQL系统大致分三类：

新架构

第一类型的NewSQL系统是全新的数据库平台，它们均采取了不同的设计方法。它们大概分两类：

(1) 这类数据库工作在一个分布式集群的节点上，其中每个节点拥有一个数据子集。 SQL查询被分成查询片段发送给自己所在的数据的节点上执行。这些数据库可以通过添加额外的节点来线性扩展。现有的这类数据库有： Google Spanner, VoltDB, Clustrix, NuoDB.

(2) 这些数据库系统通常有一个单一的主节点的数据源。它们有一组节点用来做事务处理，这些节点接到特定的SQL查询后，会把它所需的所有数据从主节点上取回来后执行SQL查询，再返回结果。

SQL引擎

第二类是高度优化的SQL存储引擎。这些系统提供了MySQL相同的编程接口，但扩展性比内置的引擎InnoDB更好。这类数据库系统有：TokuDB, MemSQL。

透明分片

这类系统提供了分片的中间件层，数据库自动分割在多个节点运行。这类数据库包扩：ScaleBase，dbShards, Scalearc。

Socket：网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个socket。

建立网络通信连接至少要一对端口号(socket)。socket本质是编程接口(API)，对TCP/IP的封装，TCP/IP也要提供可供程序员做网络开发所用的接口，这就是Socket编程接口；HTTP是轿车，提供了封装或者显示数据的具体形式；Socket是发动机，提供了网络通信的能力。

Socket的英文原义是“孔”或“插座”。作为BSD UNIX的进程通信机制，取后一种意思。通常也称作"套接字"，用于描述IP地址和端口，是一个通信链的句柄，可以用来实现不同虚拟机或不同计算机之间的通信。在Internet上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个Socket，并绑定到一个端口上，不同的端口对应于不同的服务。Socket正如其英文原义那样，像一个多孔插座。一台主机犹如布满各种插座的房间，每个插座有一个编号，有的插座提供220伏交流电， 有的提供110伏交流电，有的则提供有线电视节目。 客户软件将插头插到不同编号的插座，就可以得到不同的服务

Web restful接口：REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之间是无状态的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。

在服务器端，应用程序状态和功能可以分为各种资源。资源是一个有趣的概念实体，它向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个唯一的地址。所有资源都共享统一的接口，以便在客户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。

## 参考资料

* 《黄浦区养老机构信息手册》
* 《关于推进本市“十二五”期间养老机构建设的若干意见》（沪府[2012]105号文）
* 《关于加快发展养老服务业的若干意见》
* 《黄浦区养老设施布局专项规划》

# 软件与其他系统的关系

## 系统整体概况

为了全面推进智慧养老服务工作，以需求为导向，利用信息技术，搭建黄浦区养老服务统计分析系统，全面掌握全区老年人基本信息，统计和分析老年人历年分布情况，为领导养老服务决策提供数据支撑。

黄浦区养老服务统计分析系统包括区老年人基本信息库和区老年人统计分析系统。

1、区老年人基本信息库

采用区实有人口信息资源库共享服务接口，调用黄浦区内实有户籍老年人信息，包括人户一致以及户在并居住在黄浦的老年人员信息。老年人基本信息当前库，按周频度更新。历史数据调用自2013年1月至2017年7月以来全区户籍60岁以上老年人数据。

2、区老年人统计分析系统

（1）信息管理

利用技术手段，实现数据的获取、加工、清洗、筛选和校验，确保数据的完整性和一致性，主要由数据采集、数据导入、数据校验、数据交换、数据比对组成。

（2）综合查询

通过快速查询、按条件综合查询老年人基本信息。

（3）统计分析

依托老年人基本信息库，实现数据的归集、共享、分析，并按层次、多维度、图表结合、丰富多样化的数据展现。利用智能专业的统计分析工具，按照街道、年龄段、性别、时间等条件，对老年人数据进行同比式、环比式、金字塔、穿透式的统计分析。

## 接口系统概况

本软件接口包括以下5个主要接口:

1. 对外链接各类数据库驱动接口
2. 对内网络socket通信的接受接口
3. 对内网络socket通信的发送接口
4. 统一的restful的web通信接口
5. 内置隐藏的数据通用json接口

## 软件与接口系统关系

软件通过内外的接口，形成互相通信的模块架构，使用数据库数据交换的统一协议管理，并实现高效数据交换的能力，从而达到数据的有效性和高可用性。

# 软件总体设计

## 软件需求概括

本软件采用传统的软件开发生命周期的方法，采用自顶向下，逐步求精的结构化的软件设计方法。

本软件主要有以下几方面的功能

(1) 能兼容驱动各类数据库

(2) 利用socket通信实现数据库驱动网络化

(3) 数据库获取数据，统一使用json格式进行网络传输

(4) 利用socket获取json格式数据后，以restful接口以web方式发送

(5) web接口也已json格式数据统一交换数据

## 定义

本项目定义为一个典型的数据交换软件。它将实现高效采集各类养老数据库信息和有效数据交换的无缝对接，以实现养老数据的大数据分析的基础架构。

## 需求概述

### 功能性需求分析

通过对养老大数据交换入库实际业务的调查研究，结合软件开发的业务需求，数据交换软件可分为以下五个功能：

1.提取已有养老数据信息

2.添加新的养老数据信息

3.兼容各类数据库驱动

4.网络通信使用json格式。

5.对外实现web的restful接口。

### 非功能性需求分析

虽然功能性需求是系统需求中最重要的部分，但它仍不能体现系统需求的全貌。为确保最终质量，还需要额外满足系统在功能以外的需求。非功能性需求会对功能性需求的定义产生很大影响，体现系统易用程度。重视功能性需求，忽视非功能性需求的系统容易给用户带来不便，往往被束之高阁。

数据交换软件的非功能性需求主要涵盖适用性、易用性等六个层面的内容：

1. 适用性

这部分要求主要从客户使用的软件资源现状、硬件资源现状以及用户工作便捷性三方面提出的，下面分别进行详细说明。

智慧养老信息化程度较高，除了数据交换库软件外，还有其他的平台软件，需要与其他的软件能够友好的共存在平台系统中，利用平台的架构原理，使本软件能够发挥自身的功能。因此要求数据交换软件与已平台能有良好的兼容性，尽量减小服务器端的负担。

智慧养老平台由于工作人员数量有限，使用本系统的用户工作较为繁忙、不能出差错。为方便用户使用，软件设计舍弃了功能界面，实现自动处理功能，易于用户操作，这要求软件一方面无操作界面，另一方面保持与功能需求相一致的操作流程。

1. 易用性

易用性是提高用户满意度的要素之一。因此用户界面设计要以易于理解、可操作为基本设计目标。考虑到用户实际工作状态，本软件设计为无操作界面软件。

1. 可用性

系统的职责是保证所有数据交换系统接口数据都能准确无误地提取并存入数据库，所以系统要保证7x24小时可用，不能因为某些操作原因造成系统崩溃或不可用。

1. 可靠性

可靠、稳定的软件能够获得客户信任，软件出错频率、故障频率、故障影响程度、故障恢复期长短都是评价软件可靠性的重要因素，因此值得重视。本软件要确保每日正常运作，并实现功能模块的相互独立，以避免某个功能的故障波及到其他功能的正常使用。同时，软件有很好的数据保护措施，防止受到故障的损害，故障要求一天内恢复，软件出现问题时，要求给予用户易理解易操作的提示。

1. 安全性

由于本软件涉及养老人员的生命财产的安全，因此要求本软件拥有较高的安全性，一方面防止软件访问接口系统账户信息泄露造成越权访问和恶意篡改数据，另一方面要杜绝伪造标签进入区域。为避免软件开发者无意地进行数据操作，接口系统必须实现权限控制，开发者只能在权限范围内进行数据访问与操作。

1. 维护、扩展方便

养老的人员管理模式会随着新形势的需要进行变更，因而软件拥有高可维护性才能满足随时根据监管场所的新情况调整部门、人员或管理模式的需求，因此要尽可能简化维护功能，减轻维护工作量。

后期软件投入运营，存在用户提出新业务需求的可能性，为适应这种情况，应尽可能使功能模块保持相互独立，便于日后进行功能扩展。

## 条件与限制

### 研发硬件配置：

本系统开发设计所用的硬件是实验室电脑。其配置：

|  |  |
| --- | --- |
| CPU | Intel(R) i5 cpu |
| 主板 | Intel(R) 82865G |
| 内存 | 8GB |
| 显卡 | Intel(R) Q33 Express Chipset Family |
| 显示器 | HP L1710 17寸液晶显示器 |
| 硬盘 | 日立 HDS721616PLA380 80G |
| 网卡 | Realtek RTL8139 Family PCI Fast Ethernet NIC#2 |
| 系统 | ubuntu Linux16.04 |

表1 研发硬件配置表

### 客户使用配置

**服务端**

硬件：CPU：Intel xen e2604 cpu 内存：32G 硬盘：300G 网卡：1000M

软件：操作系统采用 ubuntu Linux16.04

**客户端**

CPU：Intel Core(TM)2 Duo Processor 内存：1G 硬盘：80G 网卡：1000M

IP网络环境

### 研发语言及编译器

本系统开发所用的语言是Go语言，开发的IDE工具LiteIDE。

### 软件支持工具

开发使用VIM软件工具。

## 软件的设计思路与原则

数据交换软件的主要目的是最大限度地借助现代信息技术来养老的信息化管理，减少人工干预，提高系统准确率，减少管理成本。目前养老的人员管理还是人工操作，养老人员达到一定数量容易出错，因此采用信息化管理手段应用本软件后可快速、精准地实现养老人员信息投放等功能，提高养老管理效率，最终方便对养老进行科学化、信息化管理。建立支撑养老管理需求的数据交换软件，对于实现该养老数据利用效果最大化来说至关重要。所以，必须认真收集、分析系统需求，来保障相关业务需求的满足。

本系统目标对象为养老管理人员，即以人员为中心，围绕人员管理数据的获取、分析、提取、存储等事件进行软件的设计与管理。

为实现以上功能，设计软件时应参照以下几项原则：

1. 数据交换精准原则：软件实现将数据交换数据纳入管理，并尽可能简化分析处理程序，提高交换数据的效率与精准度。
2. 系统操作简便原则：考虑最便于用户使用，采用无界面的方式，对用户几乎零打扰。
3. 技术先进性原则：技术上使用较领先且成熟的软件开发技术。考虑采用Go Socket技术，配以高效开发的Interface接口技术。体系结构采用目前流行的C/S。
4. 软件设计开发规范化原则：遵循软件工程的要注进行系统设计、开发；且设计、开发时采用模块化结构思想，以降低代码耦合度，提高代码重用性和系统的可扩展性。
5. 数据安全原则：确保软件运维安全、稳定，避免丢失数据的情况发生。软件数据定期备份。
6. 功能扩展性原则：软件当前针对定位设备采集到的数据进行获取、分析、提取、存取，将来随着管理需求的变化，必将要求软件功能也随之改变，因此在设计系统时要考虑代码的复用，即尽量模块化，并编写文档对其进行详细描述，以适应未来系统功能的扩展。

# 数据库驱动兼容系统设计

数据库驱动兼容系统设计是否科学影响到系统的查询速度及扩展性等性能。这里首先介绍了本软件数据库驱动兼容系统的设计原则，并进行了总体结构设计。然后提出了数据库驱动兼容概念设计，并依据概念设计中提及的E-R模型思想与第三范式的概念，对整个软件进行了结构设计。

## 设计原则

1. 软件的适应性及可维护性

软件要保证技术上的实现可行以及可扩展性，并具备满足业务变化的适应性，以满足实际业务的发展需要。

1. 技术的先进性和成熟性

软件设计一方面要考虑使用较领先的技术，保证技术发展的延续性，另一方面要兼顾工具的成熟性。

1. 软件的可靠性和安全性

软件涉及养老人员的生命财产安全，要求运行稳定，能够提供可靠的备份方案。软件设计中，注意数据的保护和隔离，包括业务操作权限、数据安全机制，数据访问权限等。

1. 软件的易维护性和可扩展性

软件要提供完善的维护工具，支持业务升级。考虑到业务发展需要，软硬件选型要考虑采用具有良好兼容性和可扩展性的产品。

## 数据库驱动兼容概念设计

在数据库驱动兼容设计过程中，第一步要完成概念设计，因此它成为了整个数据库设计的关键。概念设计有自顶向下、自底向上、逐步扩张、混合策略这四种常用方法。其中使用最频繁的是自底向上方法，即先规定各局部应用的概念结构，然后进行集成，形成全局概念结构；

概念设计阶段，通常使用E．R模型描述概念数据模型，它由实体、联系、属性三要素构成。实体是指客观存在并可彼此区别的事物；联系表示实体集之间的干系；属性是指实体或联系所具有的每一个特性。E．R模型接近于人的思维，便于理解；而且与计算机无关，用户也容易接受。

归并E．R图的关键点是消除各分E．R图中存在的属性冲突、命名冲突及结构冲突这三类冲突。属性冲突指属性值类型、取值范围等不一致；命名冲突指相同名字意义不同或不同名字意义相同；结构冲突是指在不同的应用程序相同的对象有不同的抽象，或是在不同分E-R图中拥有不同的属性个数、排列顺序或类型。解决办法是按照应用的语义对实体关联的类型进行统一调整。

## 数据库兼容驱动结构设计

根据数据库驱动兼容概念设计思想，分析得到本软件涉及的参与对象养老数据记录。一般情况下，每一个养老对象可以映射成一张数据库表，而参与对象属性映射为表中的一个字段。但是也可为了迎合更高范式的要求，适当增加数据库表减少数据冗余。下面按照该规则，首对这两个实体进行分析。每一个兼容数据的驱动都分类为单一的驱动层，例如MySQL驱动，选用go语言的mymysql驱动sdk，把驱动的操作转化成socket的json格式文本命令。

## 总体结构和模块接口设计

以下假定需要兼容的数据库为MySQL数据库，其他数据库只需要替换MySQL接口层为其他数据库接口层即可。

从数据角度：MySQL接口层->Go语言、MySQL支持库->多协议数据存储层

从通信角度：http接口层->数据协议层->数据分析层

而最终，数据编码层建立在多协议数据存储层和数据分析层之上。

系统整体结构框架如图：

Json

通信

协议

Web服务器restful接口层

网络socket通信层

Oracle驱动层

Sqlserver驱动层

。。。

Mysql驱动层

图1 系统整体结构框架图

## 模块功能逻辑关系

软件共包括web通讯模块、JSon数据解码模块、数据分析模块和数据入库模块四个主模块及JSon库、web库和MySQL接口库三个支撑类模块。

系统详细的模块信息所示：

|  |  |
| --- | --- |
| 主模块 | |
| 模块名称 | 功能简述 |
| Web通讯模块 | 对外web的json的数据接口 |
| JSon数据解码模块 | JSon数据解码 |
| 数据分析模块 | 提取数据 |
| 数据入库模块 | 数据按一定格式保存至平台数据库中 |
| 支撑类模块 | |
| 模块名称 | 功能简述 |
| JSon库 | 提取Json格式的数据 |
| Web库 | 前端设备进行web协议的数据通讯 |
| MySQL接口库 | 将数据存入MySQL数据库 |

表2 系统详细的模块信息表

模块内部关系结构如下图所示：



图2 系统模块内部关系图

## 设计和描述

本软件的主要功能是数据的提取、分析和存储的功能，软件重点是实现与各类数据库的通信和数据分析。

## 软件流程图

应用软件

接受数据

发送数据操作命令

web接口反馈

Web接口接受

分析json文本，转成web接口

分析处理命令，转成json格式

启动socket连接

连接成功？

发送json命令

Socket发送json文本

Socket服务接受

反馈数据转成json格式

Json命令转成数据库驱动命令

数据库驱动执行命令，反馈数据

# 软件函数及结构描述

## func main()

软件的主函数，实现软件的主框架的主循环。

## func webserver()

将获取到的数据根据http协议进行解码，并启动web服务，实现web接口功能

## func netsocket()

在tcpip网络中实现socket通信，利用socket的接口应用实现网络数据交换。

## func dbblock()

兼容各类数据库封装的数据模块。本文例子都使用mysql。

## func (web \*WEB)listen(conn \*net.Conn)

Web服务器启动，使用此函数实现稳定的监听，保证web接口的有效使用

## func (web \*WEB)recv(conn \*net.Conn)

Web服务器的接受功能，通过web的服务器的conn接口，实现web接受数据功能

## func(web \*WEB)send(conn \*net.Conn)

Web服务器的发送共能，通过web服务器的conn接口，实现web的发送数据功能

## func(web \*WEB)stop(conn \*net.Conn)

启动web服务器后，停止web服务，并清扫内存残留。

## func (d \*Dialer) deadline(ctx context.Context, now time.Time) (earliest time.Time)

Socket通信建立后，设置socket链接的超时时间，切断上下文内容，反馈切断上下文的最早的时间。

## func (d \*Dialer) resolver() \*Resolver

解析网络socket通信中的域名服务

## func parseNetwork(ctx context.Context, network string, needsProto bool) (afnet string, proto int, err error)

处理socket通信中，网络上下文内容为可读字串，反馈网络名称协议和出错信息。

## func (r \*Resolver) resolveAddrList(ctx context.Context, op, network, addr string, hint Addr) (addrList, error)

解析网络socket通信中的域名服务，但是根据上下文进行批量操作，反馈地址序列和出错信息。

## func Dial(network, address string) (Conn, error)

Socket的网络连接，主动发起连接。反馈socket的连接和出错信息。

## func (d \*Dialer) Dial(network, address string) (Conn, error)

在socket连接对象中，主动发起连接。反馈socket的连接和出错信息。

## func (d \*Dialer) DialContext(ctx context.Context, network, address string)

根据上下文，在socket连接对象中，主动发起连接。反馈socket的连接和出错信息。

## func dialParallel(ctx context.Context, dp \*dialParam, primaries, fallbacks addrList) (Conn, error)

根据上下文，输入网络参数，地址列表，并行的发起连接，反馈socket的连接和出错信息。

## func dialSingle(ctx context.Context, dp \*dialParam, ra Addr) (c Conn, err error)

根据上下文，输入网络参数，发送单一连接，反馈socket连接和出错信息。

## func (k \*contextKey) String() string { return "net/http context value " + k.name }

在上下文的对象中，解析出可读字符串。

## func hasPort(s string) bool { return strings.LastIndex(s, ":") > strings.LastIndex(s, "]") }

根据输入字串，获取url字串中的端口port的数值。

## func isASCII(s string) bool

判断是否为ascii的字串，对输出为true，不对输出为false

## func refererForURL(lastReq, newReq \*url.URL) string

获取web通信中的refer的url，输出为字串。

## func (c \*Client) send(req \*Request, deadline time.Time) (resp \*Response, didTimeout func() bool, err error)

在客户端的对象中，发送请求，并根据超时时间限制发送时间，反馈成功为true，不成功为false，并在反馈输出反应对象。

## func send(ireq \*Request, rt RoundTripper, deadline time.Time) (resp \*Response, didTimeout func() bool, err error)

在客户端的对象中，发送请求，并根据超时时间限制发送时间，反馈成功为true，不成功为false，并在反馈输出反应对象。发送循环次数有设置，根据输入所决定。

## func setRequestCancel(req \*Request, rt RoundTripper, deadline time.Time) (stopTimer func(), didTimeout func() bool)

设置请求取消，反馈停止时间

## func basicAuth(username, password string) string

网络通信中认证，输入用户名和密码，输出sha256的数值

## func Get(url string) (resp \*Response, err error)

在socket通信中，发送get命令，获取响应。

## func (c \*Client) checkRedirect(req \*Request, via []\*Request) error

在客户端对象中，通过请求，检查重定向。

## func (c \*Client) makeHeadersCopier(ireq \*Request) func(\*Request)

备份制作一份数据包头内容。

## func Post(url string, contentType string, body io.Reader) (resp \*Response, err error)

Web通信中发送post命令。

## func (c \*Client) Post(url string, contentType string, body io.Reader) (resp \*Response, err error)

在客户端对象中，发送post命令。

## func PostForm(url string, data url.Values) (resp \*Response, err error)

Web通信中发送post命令，命令的格式为html中的格式。

## func (c \*Client) PostForm(url string, data url.Values) (resp \*Response, err error)

在客户端对象中，发送post命令，命令的格式为html中的Form格式。

## func dbinit()

数据库操作之前的初始化。

## func (ns \*NullString) Scan(value interface{}) error

扫描数据库，获取数据库结构。

## func (ns NullString) Value() (driver.Value, error)

根据驱动字串，获取驱动属于哪个数据库的操作。

## func (n NullInt64) Value() (driver.Value, error)

根据驱动数值，获取驱动属于哪个数据库的操作。

## func (db \*DB) maxIdleConnsLocked() int

获取数据库的最大连接锁定的数值。

## func (db \*DB) SetMaxIdleConns(n int)

设置数据库最大空闲连接数。

## func (db \*DB) connectionCleaner(d time.Duration)

在数据库对象中，清理连接。

## func (db \*DB) Stats() DBStats

获取数据库的连接状态信息。

## func (db \*DB) maybeOpenNewConnections()

在数据库对象中，打开新连接。

## func (db \*DB) connectionOpener()

在数据库对象中，获取打开的连接。

## func (db \*DB) openNewConnection()

数据库打开新连接。

## func (db \*DB) nextRequestKeyLocked() uint64

在数据库对象中，下次请求，并锁定。

## func (db \*DB) conn(ctx context.Context, strategy connReuseStrategy)

根据上下文和策略获取数据库的连接。

## func (db \*DB) putConn(dc \*driverConn, err error)

在驱动对象中，把各类数据库的驱动放置在统一的连接中。

## func (db \*DB) putConnDBLocked(dc \*driverConn, err error) bool

在驱动对象中，把各类数据库的驱动放置在统一的连接中并锁定连接。

## func Marshal(v interface{}) ([]byte, error)

json格式化字串，实现标准的json对象字符串。

## func Unmarshal(data []byte, v interface{}) error

把json字符串解码成json对象结构。