**【综合监控历史库】**

**详细设计说明书**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号： | V1.0 | 文档编号： | NG\_TS\_TSDB\_DDS |
| 文档密级： | 内部公开 | 归属部门/项目： | 专业系统部 |
| 编写人： |  |  |  |

**版权信息**

本文件涉及之信息，属南京轨道交通系统工程有限公司所有。

未经南京轨道交通系统工程有限公司允许，文件中的任何部分都不能以任何形式向第三方散发。

网址：http://www.nanjingrail.com/

**文档修订记录**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **版本号** | **修订日期** | **修订人** | **修订说明** | **修订状态** | **审核日期** | **审核人** | **批准人** |
| V1.0 | 2018-4-29 |  |  | A |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

修订状态：A--增加，M--修改，D--删除

日期格式：YYYY-MM-DD

**目 录**

[1. 引言 1](#_Toc512946751)

[1.1. 说明 1](#_Toc512946752)

[1.2. 编写目的 1](#_Toc512946753)

[1.3. 背景 1](#_Toc512946754)

[1.4. 术语定义 1](#_Toc512946755)

[1.5. 参考资料 1](#_Toc512946756)

[2. 综合描述 2](#_Toc512946757)

[2.1. 系统概述 2](#_Toc512946758)

[2.2. 运行环境 2](#_Toc512946759)

[3. 系统模块设计 2](#_Toc512946760)

[3.1. 模块简介 2](#_Toc512946761)

[3.2. 核心数据模型 3](#_Toc512946762)

[3.2.1. 功能描述 3](#_Toc512946763)

[3.2.2. 模块类图 5](#_Toc512946764)

[3.2.3. 关键业务流程 6](#_Toc512946765)

[3.3. 数据冗余 17](#_Toc512946766)

[3.3.1. 功能描述 17](#_Toc512946767)

[3.3.1. 模块类图 17](#_Toc512946768)

[3.3.2. 关键业务流程 18](#_Toc512946769)

[4. 模块接口 20](#_Toc512946770)

[4.1. C API接口 20](#_Toc512946771)

[4.2. HTTP接口 22](#_Toc512946772)

[4.3. 用户管理接口 22](#_Toc512946773)

[4.4. 数据库管理接口 22](#_Toc512946774)

[4.5. 数据表管理接口 23](#_Toc512946775)

[4.6. 数据管理接口 23](#_Toc512946776)

[4.7. 数据冗余接口 23](#_Toc512946777)

[5. 用户界面详细设计 24](#_Toc512946778)

[5.1. 界面结构 24](#_Toc512946779)

[5.1.1. 界面结构或菜单结构 24](#_Toc512946780)

# 引言

## 说明

本详细设计主要描述的是在轨道交通综合监控系历史数据库子系统的解决方案。保障系统实时高效、可靠、安全的运行。

## 编写目的

本文档遵循《地铁综合监控系统技术规范》要求，依照《综合监控历史数据库需求规格说明书》和《综合监控历史数据库系统概要设计说明书》，对历史数据库的系统架构和模块进行详细设计和描述。

本部详细设计文档，作为历史数据库系统开发人员，以及相关系统开发人员的编码指导。代码编写人员需要依照本文档对系统和模块的设计要求，进行代码的编写和测试。

## 背景

轨道交通综合监控系统需要监控各种设备的状态，包括温度、湿度、电压、电流等等，并且需要每时每刻把监控的数据记录下来，用来做后续的场景回放及分析。按照每条线路10万个设备，每个设备每秒钟产生一条数据来计算的话，每秒钟将会产生10万条数据，每天将会有86.4亿条数据。如此巨大的数据量，如果采用传统的关系数据库系统进行存储，存储成本会非常的高，并且随着数据量的增加，传统关系数据库的查询速度会有很明显的下降。

所以，在综合监控系统中使用传统的关系数据库存储设备的历史数据不合理，需要针对业务数据的特征研发新的数据库用来存储历史数据。

## 术语定义

TSDB ： 基于时间序列的历史数据库

WAL ： 预写日志(Write-Ahead Logging)

TSM: 时间结构合并树(Time Structured Merge Tree)

Compactor: 数据压缩器

## 参考资料

【列举编写软件详细设计说明书时所直接参考的资料或其他资源。可能包括用户界面风格指导、合同、标准、系统需求规格说明、使用实例文档或相关软件需求规格说明、概要设计说明等。在这里，最好给出详细的信息，包括标题名称、作者、版本号、日期、出版单位或资料来源，以便于读者查阅这些文献。】

# 综合描述

## 系统概述



## 运行环境

|  |  |
| --- | --- |
| **类别** | **标准配置** |
| 计算机硬件 | Intel Xeon 12核CPU 64GB内存/1000GB硬盘,千兆网卡 |
| 软件 | CentOS 7 x64或Windows 7以上版本 |
| 网络通信 | TCP/IP协议栈支持，百兆以上带宽 |

# 系统模块设计

## 模块简介

|  |  |
| --- | --- |
| **核心数据模型** | |
| 模块名称 | 功能简述 |
| *Cache* | *内存缓存，加速数据访问* |
| *WAL* | *预写日志，保证数据完整性* |
| *TSM* | *TSM文件组织和管理* |
| *Compactor* | *数据压缩，减少磁盘空间占用* |
| **数据冗余** | |
| 模块名称 | 功能简述 |
| *心跳* | *接入集群管理服务* |
| *主从复制* | *双机热备，实时复制数据* |
| *备份还原* | *增量备份和全量备份* |
| *级联同步* | *数据级联上报汇总* |

## 核心数据模型

### 功能描述

TSDB整体结构可分为两部分：

* 元数据存储

元数据存储主要用于存储数据库的数据库结构、表结构等信息。

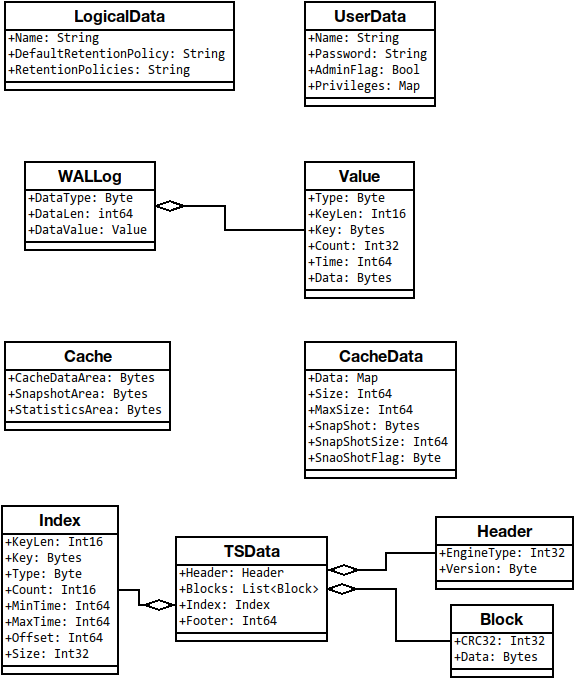


* 数据存储

数据存储部分主要由Cache模块、WAL存储模块、TSDATA存储模块、数据压缩模块、文件管理模块构成，整体结构如下



### 模块类图



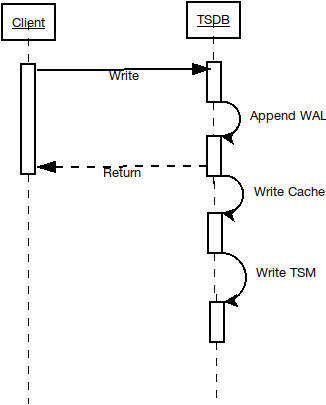
### 关键业务流程

#### 数据写入流程



流程描述如下：

1. TSDB通过网络从客户端接收数据；
2. TSDB将接收的数据通过追加方式写入WAL文件，同时将数据复制到Cache中；
3. 当WAL文件大小达到预设值后，TSDB会创建新的WAL文件，原先的WAL文件的属性将变为只读，并将该文件对应的Cache中的内容经过排序和压缩后存放到TSDATA文件中，同时在内存中添加这部分数据的索引信息；
4. 删除原先的WAL文件。



#### 数据读取流程

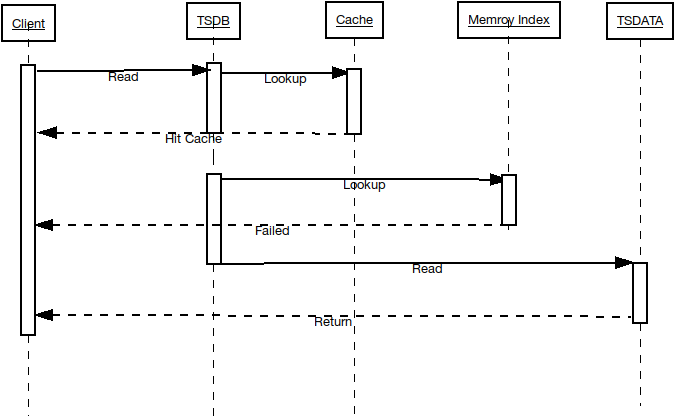
TSDB数据读取流程如下：



流程描述如下：

1. TSDB接收客户端的查询指令；
2. TSDB根据指令中的Key及约束条件在Cache中查找，如果查询成功，直接返回结果；
3. 如果在Cache中未查询成功，TSDB会从MemoryIndex中查询；
4. 如果TSDB在MemoryIndex中查询成功，TSDB会根据MemoryIndex提供的信息从TSDATA文件中读取数据，解压后返回查询结果；

如果TSDB在MemoryIndex中未查询成功，则本次查询失败；



#### 数据压缩流程

为达到较快的压缩和解压速度，同时减少对CPU的消耗，TSDB针对不同的数据类型采用不同的压缩算法。

##### 整型数据压缩

1、压缩流程描述



首先使用ZigZag算法进行编码，如果编码后的值小于 (1 << 60 ) - 1，使用simple8b算法；

如果大于(1 << 60 ) - 1，则不执行不压缩操作。

2、ZigZag算法描述

 ZigZag这个算法使用的基础就是认为在大多数情况下，我们使用的数字都是不大的数字。其编码过程如下：

1）获取int64类型输入X；

2）对X执行左移1位的操作，得到X1；

3）对X执行右移63位的操作，得到X2；

4）对X1和X2执行异或运算，得到ZigZag编码结果；

从编码过程可以看出，该算法的原理是将标志位后移至末尾，如果是负数则保留符号位移过来的1，非负数直接为0（异或操作），去掉编码中多余的前导0，则可以使用更少的字节来存储数据，从而达到压缩效果。

比如int64类型的数字1，其标志位为0，用二进制表示时前面会有63个0，最后一位才是1，执行位移操作后，X1为2，X2为0，执行异或操作后的值为2，前面有62个0，

去掉前面多余的0，仅用最后8位数表示，则编码后的数据为： 00000010 。

标志位后移主要是为了处理负数，比如int64类型的数字 -1 ，其标志位为1，用二进制表示时两端各有一个1，中间有62个0，执行位移操作后，X1为0xfffffffffffffffe，X2为0xffffffffffffffff，执行异或操作后的值为1，前面有62个0，去掉前面多余的0，仅用最后8位数表示，则编码后的数据为： 00000001 。

小整数对应的ZigZag码字短，大整数对应的ZigZag码字长。在特定的场景下，比如，要传输的整数为大整数居多，ZigZag编码的压缩效率就不理想了。而综合监控中的业务数据大部分是小数据，所以非常适合使用该算法。

该算法的解码过程如下：

 1）获取ZigZag编码结果V；

2）对V执行右移1位的操作，得到结果V1；

3）将V与1相与，得到中间值，将中间值左移63位，然后右移63位，得到结果V2；

4）对V1和V2执行异或操作，得到结果X；

3、Simple8b算法描述

Simple8b算法是64位算法，实现将多个整型数据（在 0 和 1<<60 - 1 之间）压缩到一个64位的存储结构中。

其中前4位为选择器，后面60位用于存储数据，数据使用下表进行编码：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Selector | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Bits | 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 12 | 15 | 20 | 30 | 60 |
| N | 240 | 120 | 60 | 30 | 20 | 15 | 12 | 10 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Wasted  Bits | 60 | 60 | 0 | 0 | 0 | 0 | 12 | 0 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |

编码过程如下：



这里以30个3的数组为例描述下编码过程：

1）该数组中的最大数据为3，可以使用2位二进制表示。

2）从表中可以看出，Selector等于0、1、2时都不满足编码条件，当Selector为3时，条件满足：每2个bit存储一个数据，可以存储30个数据。

3）最终数据由选择器和存储数据组成。

其中前4个bit数据为： 0011

后面存储了30个3，则后面60个bit数据为：0xfffffffffffffff

两部分数据合并在一起表示：0x3fffffffffffffff

因此，30个3使用该算法压缩后可表示为： 0x3fffffffffffffff

如果上面的30个3都使用int64进行存储，该算法的压缩后占用空间为原来的 3.3%；如果上面的30个3都使用int32进行存储，该算法的压缩后占用空间为原来的 6.7%；如果上面的30个3都使用int8（即一个Byte）进行存储，该算法的压缩后占用空间为原来的 26.7% 。

由上面的例子可以看出，该算法针对使用int64和int32存储数据的场景压缩效果是比较明显的，如果存储数据的范围波动比较大，需要使用64位或32位的int进行存储，但大部分数据的绝对值比较小（比如可以使用一个字节存储），则使用该算法的压缩效果比较明显。而综合监控系统中的数据刚好满足上述条件，因此作为存储系统的历史数据库，可以使用该算法对最终数据进行压缩后存储，节省磁盘空间。

该算法的解码过程如下：



1）首先获取压缩数据V的前4个bit作为Selector的值；

2）如果Selector的值大于或等于16，直接出错返回；

3）如果Selector的值小于16，执行解码操作：根据不同的Selector值选取不同的解码规则进行解码操作。

##### 时间戳类型数据压缩

时间戳为独立的数据类型，并且具有一定的规律可循，在TSDB中，针对时间戳先执行排序操作后使用差分编码算法进行编码，然后再根据编码结果采用不同的算法。

1、压缩流程描述



解释如下：

1）根据输入的原始数组arrValues计算出差值数组deltaValues；

2）如果差值数组的所有值相同，使用RLE编码算法；

3） 如果差值数组的所有值不同，并且差值数组的最大值大于（1 << 60）- 1，使用Raw编码算法；

4）如果差值数组的所有值不同，并且差值数组的最大值不大于（1 << 60）- 1，使用Packed编码；

2、编码算法描述

1）RLE编码算法描述

使用该算法的前提是差值数组的所有数值都相同。使用该算法进行编码时，其存储结构如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EncodeType | Divisor | Timestamp | DeltaValue | N |
| 4 bits | 4 bits | 8 Bytes | 8 Bytes | 8 Bytes |

解释如下：

EncodeType ： 记录编码类型，占4个bit；

Divisor ：记录除数的log10值，占4个bit；

Timestamp ： 记录第一个时间戳的值；

DeltaValue ： 记录第一个差值；

N ： 重复次数；

该算法的核心思想是记录数据的重复次数，其存储结构的第一个字节的高4位用于记录该存储结构使用了RLE编码，后4位记录除数的log10值。由于差值数组是相对原始数组的第一个数据计算的，所以原始数组的第一个值（第一个时间戳）必须记录，即上述结构中的Timestamp字段。差值数组的所有值都相同，所以可以在存储结构中可以记录第一个差值和重复次数，即上述结构中的DeltaValue字段和N字段。

2）Raw编码算法描述

使用该算法的前提是差值数组的最大值大于（1 << 60）- 1。使用该算法进行编码时，其存储结构如下：

|  |  |
| --- | --- |
| EncodeType | RawData |
| 4 bits | N Bytes |

解释如下：

EncodeType ：编码类型，和其它结构兼容，第一个字节的前4个bit用于记录编码类型；

RawData ： 原始数组的数据；

该算法数据没有压缩，反而增加了一个字节。为了和其它结构兼容，第一个字节的前4个bit用于记录当前存储的数据使用的是Raw编码类型。

3）Packed编码算法描述

使用该算法的前提是在差值数组的所有数值均不同，并且差值数组中数据的最大值不大于（1 << 60）- 1 。使用该算法进行编码时，其存储结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| EncodingType | Divisor | Timestamp | Simple8bData |
| 4 bits | 4 bits | 8 Bytes | N Bytes |

解释如下：

EncodeType ：记录编码类型，占4个bit；

Divisor ：记录除数的log10值，站4个bit；

Timestamp ：记录第一个时间戳的值；

Simple8bData ：差值数组使用Simple8b算法编码后的结果；

该算法首先使用差值编码对原始数据进行编码，将编码后的值除于最大共同除数Divisor（10的倍数或1），使差分数组的值尽量缩小。然后将差值数组使用Simple8b算法进行编码，进一步提高压缩效果。

##### 布尔型数据压缩

布尔类型只有2个值，只占1位数据，采用简单的位数据打包策略。

##### 浮点型数据压缩

根据浮点数的存储特征，历史数据库软件使用XOR算法对数据进行压缩存储，其存储格式如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **Flag** | **FirstValue** | **DeltaArray** | **NaN** |
| 8 bits | 64 bits | N bits | 64 bits |

解释如下：

Flag ： 数据压缩标记。

FirstValue ： 第一个数据。

DeltaArray ：压缩后的后续数据。

NaN ：结束标志。

下面描述下浮点数的存储格式及XOR算法。

**1、IEEE754标准的浮点数存储格式**

32位浮点数存储格式：

|  |  |  |
| --- | --- | --- |
| Sign | Exponent | Fraction |
| 符号位  （1 bit） | 阶码  （8 bits） | 有效数字  （23 bits） |

这种格式的非0浮点数的真值为：

64位浮点数存储格式：

|  |  |  |
| --- | --- | --- |
| Sign | Exponent | Fraction |
| 符号位  （1 bit） | 阶码  （11 bits） | 有效数字  （52 bits） |

这种格式的非0浮点数的真值为：

**2、XOR算法描述**

该算法是结合遵循IEEE754标准的浮点数存储格式的数据特征设计的特定算法，数据编码过程如下：

1）第一个值不压缩（记录为v0）；

2）计算后续值v与第一个值v0的异或值vDelta；

3）如果vDelta为0（即：v与v0的值相同），接下来的一个bit存储一个0（占用一个bit）；

4）如果vDelta不为0（即：v与v0的值不相同），接下来的一个bit存储一个1（占用一个bit），然后根据vDelta的值分以下两种情况进行处理：

如果重置前导值或尾数存储空间更优化，则按如下流程处理：

a）接下来的一个bit写入1；

b）接下来的 5 个 bit 写入vDelta值（二进制表示）中前导0的个数leading；

c）接下来的 6 个 bit 写入vDelta值（二进制表示）中有效位大小sigbits；

d）将vDelta值（二进制表示）右移去掉后面多余的0（长度前面有效数字已经标记过）得到vDelta2，写入vDelata2的值（仅有效长度）；

如果重置前导值或尾数存储空间没有达到更优效果，则之前使用之前的参数，按如下流程处理：

a）接下来的一个bit写入0；

b）将vDelta值（二进制表示）右移去掉后面多余的0（长度前面有效数字已经标记过）得到vDelta2，写入vDelata2的值（仅有效长度）；

##### 字符串型数据压缩

历史数据库软件中存储的字符串类型的数据使用Snappy算法进行压缩。Snappy的目标不是最大限度压缩或者兼容其他压缩格式，而是旨在提供高速压缩速度和合理的压缩率。

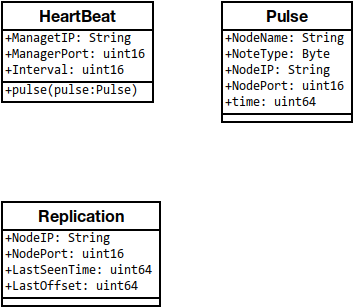
## 数据冗余

### 功能描述

为了保证数据的完整性，TSDB系统支持多个纬度的数据冗余。



### 模块类图



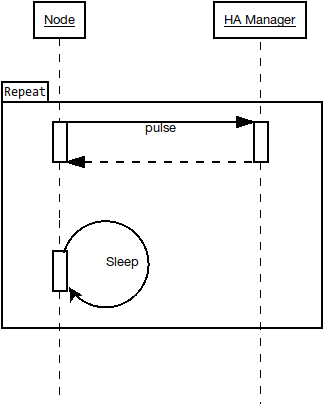
### 关键业务流程

#### 心跳流程

心跳由集群节点，定期向集群管理服务发送



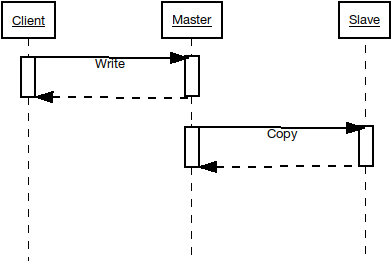
时序图如下



#### 主从复制流程



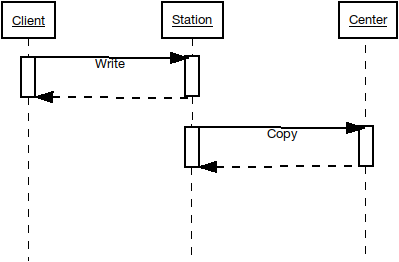
时序图如下



#### 级联同步

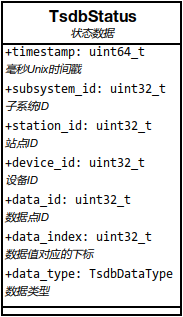


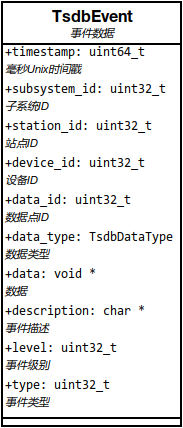
时序图如下

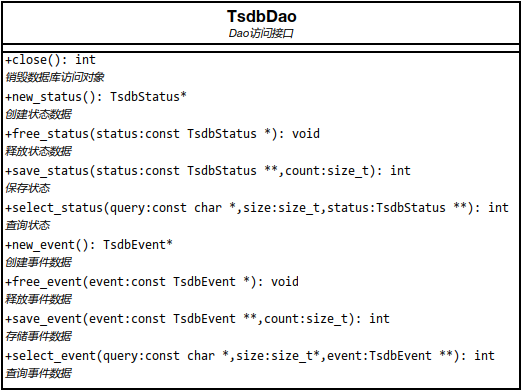


# 模块接口

## C API接口







## HTTP接口

TSDB提供基于HTTP协议的接口，实现对数据的管理，具体如下：

* 用户认证接口

认证数据库用户信息，保障数据安全。

* 数据写入接口

向数据库写入数据，支持单条数据写入和批量数据写入两种模式。

* 数据查询接口

从数据库中查询数据，支持分页功能。

## 用户管理接口

TSDB提供用户管理相关接口，具体如下：

* 添加用户接口

用于添加数据库用户，该功能需要管理员权限才能操作。

* 删除用户接口

用于删除数据库用户，该功能需要管理权限才能操作。

* 修改用户密码

用于修改数据库用户对应的密码信息，修改前需对用户信息进行认证。该功能对系统管理员和当前用户有效。

* 更新用户权限

用户权限的类别有：无操作权限、数据读取权限、数据写入权限、所有权限。

针对权限的操作有：请求权限、授予权限、撤销权限。

## 数据库管理接口

TSDB提供数据库管理接口，具体如下：

* 创建数据库接口

用于创建数据库，该功能需要管理员权限才能执行。

* 删除数据库接口

用于删除数据库，该功能需要管理员权限才能使用。

* 查询数据库列表接口

用于查看当前TSDB中有那些数据库，该功能所有用户都可使用。

* 数据库切换接口

用于改变当前工作数据库的接口。

## 数据表管理接口

历史数据库软件查询语言支持对数据表进行管理的操作，具体如下：

* 建表接口

数据表建立接口，用于创建数据表。

* 表结构更新接口

表内字段新增操作和表内字段删除操作等。

* 删表接口

删除数据表的接口。

* 查询数据表列表接口

查询当前数据库中有那些表。

## 数据管理接口

TSDB提供数据管理接口，具体如下：

* 数据写入接口

用于向数据表中写入数据。

* 数据更新接口

用于更新数据表中的数据。

* 数据删除接口

用于删除数据表中的数据，支持删除满足条件的数据，支持删除整张表的数据。

* 数据查询接口

用于查询数据表中满足条件的数据，支持排序操作、支持条件查询、支持分页操作。

## 数据冗余接口

* 心跳接口，

用于跟集群管理服务交互

* 主备切换接口

接收集群管理服务的命令，切换成master或者slave模式

* 全量备份接口

全量备份整个数据库

* 增量备份接口

基于某个时间点做备份

* 全量备份文件导入接口

用全量备份文件，整体替换当前的数据库

* 增量备份文件导入接口

将增量备份文件数据导入至当前数据库

# 用户界面详细设计

## 界面结构

### 界面结构或菜单结构



可分为如下几个部分：

* 菜单栏

数据库的管理工具的集合，包括对数据库连接的管理，数据查询语句的编辑，数据导入导出工具等；

* 工具栏

数据库快捷管理工具的集合。

* 连接管理区

以树形控件的形式显示数据库连接、数据库、数据表之间的关系。

* 数据查询区

编辑数据库查询语句。

* 数据显示区

显示数据库查询结果。

* 查询日志显示区

显示数据库查询日志。