ibear设计文档及开发手册

V.1.5.2.1

变更记录

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **变更序号** | **变更原因** | **变更页码** | **变更前版本号** | **变更后版本号** | **更改人** | **批准人** | **生效日期** | **备注** |
| **1** | **增加对多个字段是否为空判断的支持** |  | **1.0** | **1.1** | **王智** | **王智** | **16/12/19** |  |
| **2** | **增加相同key值的使用** |  | **1.1** | **1.2** | **王智** | **王智** | **16/12/20** |  |
| **3** | **优化自定义密码解析功能** | **3.5.17** | **1.2** | **1.3** | **王智** | **王智** | **16/12/21** |  |
| **4** | **增加动态库**  **对自定义密码解析功能的文档描述进行补充** | **3.5.17** | **1.3** | **1.3.1** | **王智** | **王智** | **16/12/23** |  |
| **5** | **Bug修改**  **对@lt,@gt的替换的bug；**  **select条件中包含两个及以上long、int类型导致查询不准的情况** |  | **1.3.1** | **1.3.1.1** | **王智** | **王智** | **17/4/17** |  |
| **6** | **增加游标的方式查询数据;**  **修改BySQL重载入参** | **3.5.8**  **3.5.9**  **3.5.10**  **3.5.11**  **3.5.12** | **1.3.1.1** | **1.3.2** | **王智** | **王智** | **17/4/25** |  |
| **7** | 增加resultNum值为空或0时，系统自动计算的功能；  增加了两个BySQL重载函数（无需传入resultNum）  增加默认库配置；  增加根据规则路由； | **3.5.9**  **3.5.11**  **3.1.1**  **3.1.2**  **3.5.19** | **1.3.2** | **1.3.3** | **王智** | **王智** | **18/8/8** |  |
| **8** | 增加数据缓存功能；  根据DSCI升级，进行相应升级； | **3.1.1**  **3.1.2**  **3.5.20**  **3.5.21** | **1.3.3** | **1.4** | **王智** | **王智** | **18/8/14** |  |
| **9** | 增加sql性能分析  增加“构造3” | **3.1.1**  **3.2.3**  **3.6** | **1.4** | **1.4.1** | **王智** | **王智** | **18/8/30** |  |
| **10** | 增加当“构造函数”中**bear\_db为空时，直接使用DSCI中db.cfg标签编号作为库标签** | **3.3.1**  **3.3.2**  **3.3.3** | **1.4.1** | **1.4.2** | **王智** | **王智** | **18/9/13** |  |
| 11 | 修复已知程序bug；  修改queryRoute，将入参char\*改为void\*；  新增DML重载函数，返回受影响行数；  增加ibear类的虚函数getRouteDef可用于子类继承重写；  增加可选择库进行提交/回滚的函数；  增加in绑定变量操作；  废弃init\_XxmlFile的重载函数；  增加测试模式； | 3.5.22  3.5.8  3.5.23  3.5.15  3.5.17  3.1.2  3.5.19  3.1.1 | 1.4.2 | 1.4.3 | 王智 | 王智 | 18/9/20 |  |
| 12 | 修改已知程序bug  优化预解析方式 | 无 | 1.4.3 | 1.4.4 | 王智 | 王智 | 18/12/20 |  |
| 13 | 修复make\_all中的bug | 无 | 1.4.4 | 1.4.4.1 | 王智 | 王智 | 19/02/19 |  |
| 14 | 增加获得sql db\_no的函数；  增加清空预解析序列的函数； | 3.5.26  3.5.27 | 1.4.4.1 | 1.4.4.2 | 王智 | 王智 | 19/03/15 |  |
| 15 | 增加获得sql db\_no的函数； | 3.5.28 | 1.4.4.2 | 1.4.4.3 | 王智 | 王智 | 19/03/16 |  |
| 16 | 增加绑定变量对lob的支持${a|lob}；  修改DML报错导致的core问题 | 3.1.2 | 1.4.4.3 | 1.4.4.4 | 王智 | 王智 | 19/04/12 |  |
| 17 | 去掉了对绑定变量为“空字符串”或“NULL”的限制 | 3.1.2 | 1.4.4.4 | 1.4.4.5 | 王智 | 王智 | 19/05/20 |  |
| 18 | 数据类型增加date，增加对数据类型作基本验证 | 3.1.2 | 1.4.4.5 | 1.4.4.6 | 王智 | 王智 | 19/05/24 |  |
| 19 | 增加对DMDB的重连;  增加queryForList和DML对lob的支持(增加了重载函数和xml配置);  废弃了部分API;  修改已知bug;  增加ibear\_tools工具; | 3.1.1  3.1.2  3.5.3  3.5.6  3.5.8  3.5.10  3.5.13  3.5.14  4.3 | 1.4.4.6 | 1.5 | 王智 | 王智 | 19/07/24 |  |
| 20 | 对连接mysql的bug修复；  增加对mysql结果集clear的方法； | 3.5.4 | 1.5 | 1.5.1 | 王智 | 王智 | 19/07/31 |  |
| 21 | 已知bug修补；  param节点新增len属性；  修改clearDatas | 3.1.2  3.5.4 | 1.5.1 | 1.5.2 | 王智 | 王智 | 19/08/22 |  |
| 22 | 已知bug修复； | 无 | 1.5.2 | 1.5.2.1 | 王智 | 王智 | 19/09/11 |  |

目录

1. ibear简介 7

2. ibear设计图 9

2.1 程序结构设计 9

2.2 类图设计 10

2.3 流程图设计 11

3. ibear工具类管理 13

3.1 配置文件 13

3.2 关键结构体 16

3.3 构造函数 16

3.4 属性 18

3.5 函数 18

3.6 性能日志 30

4. ibear环境搭建及DEMO 32

4.1 环境搭建 32

4.2 demo使用 32

4.3 ibear\_tools工具 32

## ibear简介

**驱动力**

电信行业业务较为庞大，业务模型设计更是十分复杂，而c++中却缺少一个能对sql进行统一管理的框架，正是基于此考虑，模仿java中的ibatis框架开发出一个c++版本，部署上只需要简单的配置即可使用，目前支持hp-ux和linux版本；  
  
**软件介绍：**通过xml配置和简单的API调用，进行数据库操作的开发；  
支持云化（支持根据集群进行自动路径配置）；  
支持在代码中直接进行sql开发，而不通过xml（开发更加灵活）；  
数据库链接部分调用DSCI接口（支持oracle，mysql，dmdb，mdb等跨库操作）；  
xml解析部分调用xml2公共库；  
  
**软件优势：**业务中的sql全部配置在xml中，方便业务梳理，同时方便管理；  
使得程序更加灵活，适配多省版本时，直接加载该省的业务xml即可，无需修改代码；  
有业务变更需要修改sql时，只需同步新的sqlxml而无需编译程序；  
对sql的预解析，执行，返回等操作进行统一管理，例如各种数据库异常处理等，外部系统无需关注；  
可根据需求，定制ERROR\_CODE（配置有专属的ibear\_err\_define.h，可自定义当中的errcode）；  
命名规则浅显易懂，且均采用工具名称作为开头，避免和其他类库冲突；

由于xml解析等繁琐工作均在程序启动时完成，程序运行过程中对工具的调用基本没有耗时，预解析时判断该sql是否已经经历过预解析，如果已经预解析过，则直接执行。所以工具在性能上不存在任何损耗；

在程序对工具调用过程中，采用高效的key-value算法，根据唯一的sql名称，找到对应的sql；

**第三方工具说明：**DSCI由王金山提供（用于数据库基础操作）  
xml2为开源工具（用于xml解析）

**目前使用情况**安徽移动账务后台进行使用；

山西移动实时营销事件中心使用；

黑龙江移动账管后台程序使用；

新版本信控中进行使用；

由于该工具没有使用限制，以及后续c++开发中会持续进行使用

## ibear设计图

### 程序结构设计

前台部分（API部分）：

该部分和后台部分独立开来，中间通过私有接口进行调用，相互之间的逻辑变更不会导致前台对外提供的服务变更；

后台部分（xml解析、基础数据库操作封装）：

Xml解析部分为单独的公共函数完成；

对基础数据库操作（DSCI）进行了二次封装，接口仅对前台部分提供；

后台部分的任何修改，不会导致前台的API口径变更；

**

*2.1程序结构图*

### 类图设计

1、Ibear作为整个工程的核心类，对外：提供初始化、增删改查等操作接口；

对内，调用各个功能类、结构体，完成接口工作；

其中Ibear中方法及部分属性为公有；

2、stSqlMap---存放解析前后的sql，key，stSQLParam，公有类，外部可以访问；

3、stSQLParam----存放sql中变量的属性名、属性值、是否可空、字段类型等，私有类，外部无法访问；

4、stSettings---存放sqlMapConfig.xml的配置元素，如：是否自动提交等，私有类，外部无法访问；

5、stSqlMapConfig---存放整个sqlMapConfig内容，包含stSettings以及各个sqlxml的配置路径，私有类，外部无法访问；



*2.2类图*

### 流程图设计

初始化流程：

1. 对ibear类进行初始化，ibear内部会对xml进行解析，并将解析后的各个值存放于各个功能类当中；
2. 调用DSCI接口，对数据库链接进行初始化，初始化结束；



*2.3初始化流程图*

调用API流程：

1. 调用ibear中的sql处理函数，并传入sql对应的key，以及sql参数值（以map的方式传入）；
2. 根据key找到对应的sql，根据入参map进行sql参数赋值（参数解析方式非硬解析方式）
3. 调用封装后的预解析函数，并将该sql存放于预解析结构体中（该步骤可以动态控制sql的预解析情况，已经做过预解析的sql不会再次进行预解析操作；如果没有调用到的sql，即使在xml中配置，也不会进行预解析操作）；
4. 执行sql，并返回结果，如果报错，会返回errcode（分为数据库error\_code和自定义no\_error）以及sql\_error；



*2.2调用流程图*

## ibear工具类管理

### 配置文件

#### sqlMapConfig.xml

|  |  |  |  |
| --- | --- | --- | --- |
| **属性名** | **名称** | **值** | **备注** |
| **cacheModelsEnabled** | **sql缓存** | **true默认启动缓存；**  **false默认不做缓存；** | **缓存开关** |
| **testModel** | **测试模式** | **true开启测试模式；**  **false关闭测试模式；** | **默认关闭；**  **开启测试模式，所有DML操作将写入文件，并自动执行回滚，日志路径：performanceLogPath** |
| **autoCommit** | **自动提交标示** | **0、手动提交；1、自动提交** |  |
| **defaultDBNum** | **默认库标示** | **数据库标签**  **当data.xml中sql的db\_no为空时生效；** |  |
| **autoClearEnabled** | **自动清理策略** | **TransationClear：事务清理（提交or回滚时清理）；**  **SelfClear：手工清理；** | **事务自动调用：**  **ClearAllMemData**  **手工可调用：**  **ClearAllMemData;**  **ClearTableMemData** |
| **maxOverflaw** | **最大缓存大小** | **配置缓存的最大值，单位Mb** | **默认256M** |
| **performanceAnalyzeEnabled** | **性能分析** | **true：启动开关**  **false：关闭开关** | **需要部署ilog、ifile工具，否则无法使用** |
| **performanceLogPath** | **日志输出路径** | **测试模式日志和性能日志均使用该路径** | **路径下要有temp和runlog目录** |
| **performanceThreshold** | **sql耗时阈值** | **单位微妙** | **十万分之一秒** |
| **reConnect** | **重连** | **true：开启**  **false：关闭** | **只针对DMDB做重连；**  **默认关闭，打开时会自动重连两次；** |
| **resource** | **关联xml** | **配置sql语句xml路径，可多个** |  |

格式如下：

<?xml version="1.0" encoding="UTF-8"?>

<sqlMapConfig>

<settings cacheModelsEnabled="true"

testModel="true"

autoCommit="0"

defaultDBNum="1"

autoClearEnabled="TransationClear"

maxOverflaw="256"

performanceAnalyzeEnabled="true"

performanceLogPath="./"

performanceThreshold="1000"/>

<sqlMap resource="./data1.xml" />

<sqlMap resource="./data2.xml" />

</sqlMapConfig>

#### data.xml

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **父节点** | **属性名** | **名称** | **值** | **可空** | **备注** |
| **sqlmap** | **select** | **查询** |  |  | **详见示例1** |
| **delete** | **删除** |  |  |  |
| **insert** | **插入** |  |  |  |
| **update** | **更新** |  |  |  |
| **select** | **id** | **Sql标示** | **唯一获取该sql的id** | **否** |  |
| **parameterClass** | **入参类型** | **目前只支持map** | **否** |  |
| **resultNum** | **出参个数** | **Sql语句列个数，只有select需要；系统会自动计算列个数；** | **是** | **当遇到复杂sql导致无法正确计算列数时，需要手工填写该值；** |
| **db\_no** | **库标示** | **数据库标签**  **填写初始化bear\_db时的数组下标（从0开始）** | **是** | **为空时，使用defaultDBNum赋值;**  **为?时,使用queryRoute赋值;** |
| **memType** | **缓存类型** | **false：不缓存**  **true：缓存**  **forever：永久缓存** | **是** | **cacheModelsEnabled为true时默认生效；**  **否则默认不生效；** |
| **testModel** | **测试模式** | **true开启测试模式；**  **false关闭测试模式；** | **是** | **sqlMapConfig.xml中的testModel为true时，该配置生效；默认开启；** |
| **param** | **输出参数** | **子节点** | **是** | **Lob类型需要该子节点**  **详见示例5** |
| **param** | **name** | **参数名** | **sql字段名** | **否** |  |
| **type** | **参数类型** | **sql字段类型** | **否** | **目前支持clob/blob** |
| **len** | **参数长度** | **sql字段长度** | **是** | **默认1024** |
| **select** | **sql内容** | **Sql语句** | **配置通用sql：**  **对传入的变量值透传；** | **否** | **xml特殊字符处理：**  **<对应@lt**  **>对应@gt**  **sql结尾请勿加";"** |
| **sql内容** | **#value#** | **硬解析** | **直接用value对sql进行替换** |  | **详见示例2** |
| **$E[and xxx=y]** | **空判断** | **若不对y赋值，则删除[]中的内容；** |  | **详见示例3** |
| **${value|type|num}** | **软解析** | **type枚举值：int、long、char、blob、clob/lob、date**  **num可为空**  **type为blob时num必填** |  | **数据类型做基本验证：**  **验证date格式，支持类型如下：YYYYMMDDHHMISS**  **YYYY-MM-DD HH:MI:SS**  **YYYYMMDD**  **YYYY-MM-DD**  **详见示例4** |
| **$L[xxxx]** | **一组数据的软解析** | **一组数据的软解析，建议在in ()元素个数不确定的情况下适用；** |  | **详见示例6** |

**示例1：**

<?xml version="1.0" encoding="UTF-8"?>

<sqlMap namespace="test">

<select id="key" parameterClass="Map" resultNum="4" db\_no="1" memType="false" testModel=”false”>

select ACCT\_ITEM\_CODE, payed\_prepay,payed\_later,status

from act\_payedowe\_#MONTH#

where contract\_no= ${CONTRACT\_NO|long}

$E[ and RUN\_CODE in ($L[runcode])]

and ROWNUM @lt 2

</select>

</sqlMap>

**示例2：**

<select id="key" parameterClass="Map" db\_no="1">

select a,b,c,d,e from abc\_#MONTH#

</select>

**示例3：**

<select id="key" parameterClass="Map" db\_no="1">

select a,b,c,d,e from abc where a=${A|long}

$E[ and b={B|char}]

</select>

**示例4：**

<insert id="INSERT\_BLOB" parameterClass="Map" db\_no="1">

insert into abc(a,b,c,d,e) values(${A|char},${B|long},${C|date},${D|blob|128},${E|clob|200})

</insert>

**示例5：**

<select id="SELECT\_BLOB" parameterClass="Map" db\_no="1">

<param name="d" type="blob" len="128"/>

<param name="e" type="clob"/>

select a,b,c,d,e from abc where a=${A|char}

</select>

</sqlMap>

**示例6：**

<select id="key" parameterClass="Map" db\_no="1">

select a,b,c,d,e from abc where a in ($L[runcode])]

</select>

当sql使用$L[runcode]时，程序会认为以runcode开头的key是一组数据；

inParams.insert(pair<string,string>(string("runcode3"),x));

inParams.insert(pair<string,string>(string("runcodea"),y));

inParams.insert(pair<string,string>(string("runcode\_"),y));

### 关键结构体

|  |  |
| --- | --- |
| **结构体** | **说明** |
| **bear\_db** | **存储一组数据库连接标签；数组从0开始，值对应DSCI中的db.cfg；后续xml中配置该数组下标；** |
| **stSettings** | **存储sqlMapConfig.xml中存储的值** |
| **stSqlMap** | **存储data.xml中的数据** |
| **stDBSection** | **存储用户名、密码、库名** |

### 构造函数

#### 构造1

|  |  |
| --- | --- |
| **函数原型** | **ibear(bear\_db \*pInfo,map<string,stSqlMap\*> sqlconfs,stSettings \*set);** |
| **功能说明** | **构造函数，通过pInfo找到程序的数据库dsc编码并做初始化** |
| **参数说明** | **bear\_db \*pInfo ：存储数据库标签的数组；若为NULL，则直接使用DSCI/cfg/db.cfg中的标签：例DB03对应为3；**  **map<string,stSqlMap\*> sqlconfs ：data.xml配置内容**  **stSettings \*set：sqlMapConfig.xml配置内容** |
| **返回值** | **ibear** |
| **备注** | **需要在程序中用到xml文件中的内容时使用该构造函数；**  **需要先进行xml初始化才能使用该构造函数；** |
| **实例** | **map<string,stSqlMap\*> sqlconfs;**  **stSettings \*settings = new stSettings();**  **init\_XxmlFile(filename, host\_id, &sqlconfs,settings);**  **bear\_db \* pInfo = new bear\_db();**  **pInfo ->db\_id[0] = 1;**  **pInfo ->db\_id[1] = 2;**  **ibear \*br = new ibear(pInfo,sqlconfs, settings);**  **or**  **ibear \*br = new ibear(NULL,sqlconfs, settings);** |

#### 构造2

|  |  |
| --- | --- |
| **函数原型** | **ibear(char \*fileName,bear\_db \*pInfo);** |
| **功能说明** | **构造函数，通过xml文件名，和pInfo找到数据库并做初始化** |
| **参数说明** | **char \*fileName ：sqlMapConfig.xml的路径**  **bear\_db \*pInfo：存储数据库标签的数组；若为NULL，则直接使用DSCI/cfg/db.cfg中的标签：例DB03对应为3；** |
| **返回值** | **ibear** |
| **备注** |  |
| **实例** | **bear\_db \* pInfo = new bear\_db();**  **pInfo ->db\_id[0] = 1;**  **pInfo ->db\_id[1] = 2;**  **ibear \*br = new ibear(filename,pInfo);**  **or**  **ibear \*br = new ibear(filename,NULL);** |

#### 构造3

|  |  |
| --- | --- |
| **函数原型** | **ibear(char \*fileName,bear\_db \*pInfo ,char \*host\_id);** |
| **功能说明** | **构造函数，通过xml文件名，和pInfo找到数据库并做初始化** |
| **参数说明** | **char \*fileName ：sqlMapConfig.xml的路径**  **bear\_db \*pInfo：存储数据库标签的数组；若为NULL，则直接使用DSCI/cfg/db.cfg中的标签：例DB03对应为3；**  **char \*host\_id：集群名称** |
| **返回值** | **ibear** |
| **备注** | **需要在配置中使用${host\_id}替换集群号时，使用该构造函数；** |
| **实例** | **bear\_db \* pInfo = new bear\_db();**  **pInfo ->db\_id[0] = 1;**  **pInfo ->db\_id[1] = 2;**  **ibear \*br = new ibear(filename,pInfo,host\_id);**  **or**  **ibear \*br = new ibear(filename,NULL,host\_id);** |

### 属性

|  |  |  |
| --- | --- | --- |
| **属性名** | **类型** | **说明** |
| **m\_Sqlconfs** | **map<string,stSqlMap\*>** | **解析后的xml，存放全部sql及其属性值等内容** |
| **m\_dbDsc** | **IBEAR\_DB\_INFO** | **数据库信息** |
| **sql\_error** | **char** | **错误信息** |
| **no\_error** | **char** | **自定义errorcode** |
| **error\_code** | **int** | **数据库返回errorcode** |

### 函数

#### getDBP

|  |  |
| --- | --- |
| **函数原型** | **void getDBP(int db\_no,void \*dbp);** |
| **功能说明** | **获得数据库连接句柄** |
| **参数说明** | **int db\_no：数据库标签**  **void \*dbp：数据库连接句柄** |
| **返回值** | **无：** |
| **备注** |  |

#### queryForList

|  |  |
| --- | --- |
| **函数原型** | **int queryForList(char \*sql\_key ,map<string,string> inParams, vector<vector<string> > \*list)** |
| **功能说明** | **执行select，获得多行结果** |
| **参数说明** | **char \*sql\_key ：sql唯一标示**  **map<string,string> inParams：存储数据库标签的数组**  **vector<vector<string> > \*list：出参，结果集列表** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数，不支持lob出参类型** |

#### queryForList\*

|  |  |
| --- | --- |
| **函数原型** | **int queryForList(char \*sql\_key ,map<string,void\*> inParams, vector<vector<void\*> > \*list)** |
| **功能说明** | **执行select，获得多行结果** |
| **参数说明** | **char \*sql\_key ：sql唯一标示**  **map<string,void\*> inParams：存储数据库标签的数组**  **vector<vector<void\*> > \*list：出参，结果集列表** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数，支持全部出参类型，1.5版本新增函数，使用后需要clearDatas清理list内存** |

#### clearDatas\*

|  |  |
| --- | --- |
| **函数原型** | **void clearDatas (vector<vector<void\*> > &mList)** |
| **功能说明** | **对queryForList的结果集进行内存清理** |
| **参数说明** | **vector<vector<void\*> > &mList：结果集** |
| **返回值** | **无** |
| **备注** | **使用queryForList参数为void\*的重载函数时，必须执行该函数释放不再使用的结果集，否则会导致程序内存泄漏；** |

#### clearDatas\*

|  |  |
| --- | --- |
| **函数原型** | **void clearDatas (vector<void\*> &mList)** |
| **功能说明** | **对queryForObject的结果集进行内存清理** |
| **参数说明** | **vector<void\*> &mList：结果集** |
| **返回值** | **无** |
| **备注** | **使用queryForObject/fetchRow参数为void\*的重载函数时，必须执行该函数释放不再使用的结果集，否则会导致程序内存泄漏；** |

#### ~~queryForList（已过时）~~

|  |  |
| --- | --- |
| **函数原型** | **int queryForList(const char \*sql\_key,map<string,string> &inParams,vector<vector<string> > \*list,int num);** |
| **功能说明** | **执行select，获得多行结果**  该函数在1.5版本过时(编译会有警告)；  出参包含lob字段时，可使用xml中的<param name=”xxx” type=”clob”>来获得，后续版本将会废弃 |
| **参数说明** | **char \*sql\_key ：sql唯一标示**  **map<string,string> inParams：存储数据库标签的数组**  **vector<vector<string> > \*list：出参，结果集列表**  **int num：lob字段所在的列（从1开始计算）** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数，在1.5版本中已过时** |

#### queryForList

|  |  |
| --- | --- |
| **函数原型** | **Int queryForList(const char \*sql\_key, map<string,string> &inParams);** |
| **功能说明** | **执行select，配合fetchRow使用** |
| **参数说明** | **char \*sql\_key ：sql唯一标示**  **map<string,string> &inParams：存储数据库标签的数组** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数，不支持lob出参类型，不支持缓存** |

#### queryForList\*

|  |  |
| --- | --- |
| **函数原型** | **Int queryForList(const char \*sql\_key, map<string,void\*> &inParams);** |
| **功能说明** | **执行select，配合fetchRow使用** |
| **参数说明** | **char \*sql\_key ：sql唯一标示**  **map<string,void\*> &inParams：存储数据库标签的数组** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数，支持全部出参类型，不支持缓存，1.5版本新增函数，使用后需要clearDatas清理list内存** |

#### fetchRow

|  |  |
| --- | --- |
| **函数原型** | **int fetchRow(const char \*sql\_key,vector<string> \*out\_row);** |
| **功能说明** | **执行select，获得游标所在行数据，单行数据** |
| **参数说明** | **char \*sql\_key ：sql唯一标示**  **vector<string> \*out\_row：出参，结果集** |
| **返回值** | **0：成功；**  **1：**游标已经到最后  **-1：失败；** |
| **备注** | **必须先使用3.5.6/3.5.5，才能使用该函数**  **不支持lob，不支持缓存** |
| **实例** | vector<string> v\_row;  iRet = m\_Ibear->queryForList("SEL\_TEST\_LISTS",inParams);  while (1)  {  *//每次调用返回一行，入参需要对应sqlkey，出参为单行数据*  iRet = m\_Ibear->fetchRow("SEL\_TEST\_LISTS",&v\_row);  if (iRet < 0)  return iRet;  else if (iRet == 1)  break;  printf("\n[%d]%s\n",++i,v\_row[0].c\_str());  } |

#### fetchRow\*

|  |  |
| --- | --- |
| **函数原型** | **int fetchRow(const char \*sql\_key,vector<void\*> \*out\_row);** |
| **功能说明** | **执行select，获得游标所在行数据，单行数据** |
| **参数说明** | **char \*sql\_key ：sql唯一标示**  **vector<void\*> \*out\_row：出参，结果集** |
| **返回值** | **0：成功；**  **1：**游标已经到最后  **-1：失败；** |
| **备注** | **必须先使用3.5.6/3.5.5，才能使用该函数**  **支持全部出参类型，不支持缓存，1.5版本新增函数，使用后需要clearDatas清理out\_row内存** |
| **实例** | vector<void\*> v\_row;  iRet = m\_Ibear->queryForList("SEL\_TEST\_LISTS",inParams);  while (1)  {  *//每次调用返回一行，入参需要对应sqlkey，出参为单行数据*  iRet = m\_Ibear->fetchRow("SEL\_TEST\_LISTS",&v\_row);  if (iRet < 0)  return iRet;  else if (iRet == 1)  break;  printf("\n[%d]%s\n",++i,(char\*)v\_row[0]);  } |

#### queryForObject

|  |  |
| --- | --- |
| **函数原型** | **int queryForObject (char \*sql\_key,map<string,string> inParams, vector<string> \*object);** |
| **功能说明** | **执行select，获得单行结果** |
| **参数说明** | **char \*sql\_key ：sql唯一标示；**  **map<string,string> inParams：存储数据库标签的数组；**  **vector<string> \*object：出参，结果集；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数，不支持lob出参类型** |

#### queryForObject\*

|  |  |
| --- | --- |
| **函数原型** | **int queryForObject (char \*sql\_key,map<string,void\*> inParams, vector<void\*> \*object);** |
| **功能说明** | **执行select，获得单行结果** |
| **参数说明** | **char \*sql\_key ：sql唯一标示；**  **map<string,void\*> inParams：存储数据库标签的数组；**  **vector<void\*> \*object：出参，结果集；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数，支持全部出参类型，1.5版本新增函数，使用后需要clearDatas清理object内存** |

#### DML

|  |  |
| --- | --- |
| **函数原型** | **int DML(char \*sql\_key,map<string,string> inParams);** |
| **功能说明** | **执行update，delete，insert** |
| **参数说明** | **char \*sql\_key ：sql唯一标示；**  **map<string,string> inParams：存储数据库标签的数组；** |
| **返回值** | **0：成功；**  **1:受影响行为0；**  **-1：失败；** |
| **备注** | **不支持lob类型，不支持缓存** |

#### DML

|  |  |
| --- | --- |
| **函数原型** | **int DML(char \*sql\_key,map<string,string> inParams,int \*rownum);** |
| **功能说明** | **执行update，delete，insert，输出受影响行数** |
| **参数说明** | **char \*sql\_key ：sql唯一标示；**  **map<string,string> inParams：存储数据库标签的数组；**  **int \*rownum：返回受影响行数** |
| **返回值** | **0：成功；**  **1:受影响行为0；**  **-1：失败；** |
| **备注** | **不支持lob类型，不支持缓存** |

#### DML\*

|  |  |
| --- | --- |
| **函数原型** | **int DML(char \*sql\_key,map<string,void\*> inParams);** |
| **功能说明** | **执行update，delete，insert** |
| **参数说明** | **char \*sql\_key ：sql唯一标示；**  **map<string,void\*> inParams：存储数据库标签的数组；** |
| **返回值** | **0：成功；**  **1:受影响行为0；**  **-1：失败；** |
| **备注** | **支持全部类型，不支持缓存，1.5版本新增函数** |

#### DML\*

|  |  |
| --- | --- |
| **函数原型** | **int DML(char \*sql\_key,map<string,void\*> inParams,int \*rownum);** |
| **功能说明** | **执行update，delete，insert，输出受影响行数** |
| **参数说明** | **char \*sql\_key ：sql唯一标示；**  **map<string,void\*> inParams：存储数据库标签的数组；**  **int \*rownum：返回受影响行数** |
| **返回值** | **0：成功；**  **1:受影响行为0；**  **-1：失败；** |
| **备注** | **支持全部类型，不支持缓存，1.5版本新增函数** |

#### queryForListBySQL

|  |  |
| --- | --- |
| **函数原型** | **int queryForListBySQL(int db\_no\_in, int res\_num, char \*sql\_in, map<string,string> &inParams,vector<vector<string> > \*list)** |
| **功能说明** | **自定义sql，执行select，获得多行结果** |
| **参数说明** | **db\_no\_in：数据库标签；**  **int res\_num：sql列个数；**  **char \* sql\_in：自定义sql语句；**  **map<string,string> &inParams：存储数据库标签的数组；**  **vector<vector<string> > \*list：出参，结果集列表** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **函数重载，不支持lob类型，不支持缓存** |

#### queryForListBySQL

|  |  |
| --- | --- |
| **函数原型** | **int queryForListBySQL(int db\_no\_in,char \*sql\_in,map<string,string> &inParams,vector<vector<string> > \*list);** |
| **功能说明** | **自定义sql，执行select，获得多行结果** |
| **参数说明** | **db\_no\_in：数据库标签；**  **char \* sql\_in：自定义sql语句；**  **map<string,string> &inParams：存储数据库标签的数组；**  **vector<vector<string> > \*list：出参，结果集列表；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **函数重载，不支持lob类型，不支持缓存** |

#### queryForObjectBySQL

|  |  |
| --- | --- |
| **函数原型** | **int queryForObjectBySQL(int db\_no\_in, int res\_num,char \*sql\_in, map<string,string> &inParams,vector<string> \*object);** |
| **功能说明** | **自定义sql，执行select，获得单行结果** |
| **参数说明** | **db\_no\_in：数据库标签；**  **int res\_num：sql列个数；**  **char \* sql\_in：自定义sql语句；**  **map<string,string> &inParams：存储数据库标签的数组；**  **vector<string> \*object：出参，结果集；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **函数重载，不支持lob类型，不支持缓存** |

#### queryForObjectBySQL

|  |  |
| --- | --- |
| **函数原型** | **int queryForObjectBySQL(int db\_no\_in, char \*sql\_in, map<string,string> &inParams, vector<string> \*object);** |
| **功能说明** | **自定义sql，执行select，获得单行结果** |
| **参数说明** | **db\_no\_in：数据库标签；**  **char \* sql\_in：自定义sql语句；**  **map<string,string> &inParams：存储数据库标签的数组；**  **vector<string> \*object：出参，结果集；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **函数重载，不支持lob类型，不支持缓存** |

#### DMLBySQL

|  |  |
| --- | --- |
| **函数原型** | **int DMLBySQL(int db\_no\_in, char \*sql\_in, map<string,string> &inParams);** |
| **功能说明** | **自定义sql，执行update，delete，insert** |
| **参数说明** | **db\_no\_in：数据库标签；**  **char \* sql\_in：自定义sql语句；**  **map<string,string> &inParams：存储数据库标签的数组；** |
| **返回值** | **0：成功；**  **1:受影响行为0；**  **-1：失败；** |
| **备注** | **不支持缓存，不支持lob类型，不支持缓存** |

#### Commit

|  |  |
| --- | --- |
| **函数原型** | **int Commit();** |
| **功能说明** | **提交** |
| **参数说明** | **无；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** |  |

#### Commit

|  |  |
| --- | --- |
| **函数原型** | **int Commit(int n);** |
| **功能说明** | **按库提交** |
| **参数说明** | **int n：库标示** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** |  |

#### Rollback

|  |  |
| --- | --- |
| **函数原型** | **int Rollback();** |
| **功能说明** | **回滚** |
| **参数说明** | **无；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** |  |

#### Rollback

|  |  |
| --- | --- |
| **函数原型** | **int Rollback(int n);** |
| **功能说明** | **按库回滚** |
| **参数说明** | **int n：库标示** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** |  |

#### Quit

|  |  |
| --- | --- |
| **函数原型** | **void Quit();** |
| **功能说明** | **断开连接** |
| **参数说明** | **无；** |
| **返回值** | **无；** |
| **备注** |  |

#### ~~init\_XxmlFile（废弃）~~

|  |  |
| --- | --- |
| **函数原型** | **int init\_XxmlFile (char \*fileName,map<string,stSqlMap\*> \*sqlconfs, stSettings \*settings);** |
| **功能说明** | **初始化xml** |
| **参数说明** | char \*filename：sqlMapConfig路径；  map<string,stSqlMap\*> sqlconfs ：data.xml配置内容  stSettings \*set：sqlMapConfig.xml配置内容 |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **废弃，由3.5.20替代init\_XxmlFile**  **若无需host\_id,传NULL即可** |

#### init\_XxmlFile

|  |  |
| --- | --- |
| **函数原型** | **int init\_XxmlFile(char \*fileName,char \*host\_id,map<string,stSqlMap\*> \*sqlconfs,stSettings \*settings);** |
| **功能说明** | **初始化xml** |
| **参数说明** | **char \*filename：sqlMapConfig路径；**  **char \*host\_id：集群名称**  **map<string,stSqlMap\*> sqlconfs ：data.xml配置内容**  **stSettings \*set：sqlMapConfig.xml配置内容** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **重载函数**  **配合“3.2.1构造1”使用**  **支持xml的resource中{host\_id}替换为入参的host\_id（多使用于集群名）** |

#### Custom\_decryption

|  |  |
| --- | --- |
| **函数原型** | **int Custom\_decryption(void (\*getDBSection)(char \*,stDBSection \*),char \*dbnameKey,stDBSection \*dbsection);** |
| **功能说明** | **自定义解密函数** |
| **参数说明** | **void (\*getDBSection)(char \*,stDBSection \*)：自定义解密函数**  **char \*dbnameKey：出参，返回dsci密码文件名**  **stDBSection \*dbsection：出参，返回用户、密码、库名** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **回调函数，由ibear\_db\_pwd调用** |
| **实例** | ***//重写解密规则***  **void getDBSection(char \*dbnameKey,stDBSection \*dbsection)**  **{**  **strcpy(dbsection->sDBName,"BILLCOLLDEV");**  **strcpy(dbsection->sDBUser,"DBACCOPR");**  **strcpy(dbsection->sDBPwd,"Lf5k3rcf");**  **}**  ***//初始化ibear\_db\_pwd类***  **ibear\_db\_pwd \*idp = new ibear\_db\_pwd();**  ***//初始化stDBSection类***  **stDBSection \*dbs = new stDBSection();**  ***//调用Custom\_decryption函数，入参为“自定义函数”***  **if (0 != idp->Custom\_decryption(getDBSection,key,dbs))**  **{**  **return -1;**  **}** |

#### queryRoute

|  |  |
| --- | --- |
| **函数原型** | **int queryRoute(int (\*getRoute)(void \*,int \*), void \*rules,int \*db\_no);** |
| **功能说明** | **自定义路由函数** |
| **参数说明** | **int (\*getRoute)(void \*,int \*)：自定义路由函数**  **void \*rules：入参，选库规则；**  **int \*db\_no：出参，返回数据库标示（数组下标）；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **回调函数，由ibear调用** |
| **实例** | ***//重写路由规则***  **int getRoute(char \*rules,int \*db\_no){**  **\*db\_no = 1;**  **return 0;**  **}**  ***//初始化db\_no***  **int db\_no = 0;**  ***//初始化ibear***  **ibear \*br = new ibear(filename,pInfo);**  **br->queryRoute(getRoute,NULL,&db\_no);** |

#### getRouteDef

|  |  |
| --- | --- |
| **函数原型** | virtual int getRouteDef(void \*rules,int \*db\_no) |
| **功能说明** | **自定义路由虚函数，用于子类重写** |
| **参数说明** | **void \*rules：入参，选库规则；**  **int \*db\_no：出参，返回数据库标示；** |
| **返回值** | **0：成功；**  **-1：失败；** |
| **备注** | **虚函数，由ibear子类重写，跨库时自动调用**  **利用ibear子类重写路由函数来实现路由功能，sql需要路由时自动根据重写的函数进行路由判断，无需每次都手工调用queryRoute函数；** |
| **实例** | class myibear : public ibear {  myibear(char \*fileName,bear\_db \*pInfo):ibear(fileName,pInfo){};  ~myibear(){};  int getRouteDef(void \*rules,int \*db\_no);  **}**  **//实现**  int myibear::getRouteDef(void \*rules,int \*db\_no)  {  //根据rules进行路由规则判断，给db\_no赋值  \*db\_no=2;  }  //使用和ibear一致  myibear \*br = new myibear(filename,NULL);  若需要路由，则要注意：  1. xml中的db\_no=“?”  2.入参中要传入ROUTE\_KEY inParams.insert(pair<string,string>(string("ROUTE\_KEY"),toString(routeKey))); |

#### ClearAllMemData

|  |  |
| --- | --- |
| **函数原型** | **void ClearAllMemData()** |
| **功能说明** | **清理全部缓存数据** |
| **参数说明** | **无** |
| **返回值** | **无** |
| **备注** | **无** |

#### ClearTableMemData

|  |  |
| --- | --- |
| **函数原型** | **void ClearTableMemData(string table)** |
| **功能说明** | **按表清理缓存数据** |
| **参数说明** | **表名** |
| **返回值** | **无** |
| **备注** | **无** |

#### getDBNum

|  |  |
| --- | --- |
| **函数原型** | **char\* getDBNum(const char \*sql\_key)** |
| **功能说明** | **获取sql所在库的标签** |
| **参数说明** | **char \*sql\_key ：sql唯一标示** |
| **返回值** | **数据库标签** |
| **备注** | **无** |

#### clearPrepareSql

|  |  |
| --- | --- |
| **函数原型** | **void clearPrepareSql(int db\_no)** |
| **功能说明** | **按库标签清理预解析列表** |
| **参数说明** | **db\_no：数据库标签；** |
| **返回值** | **无** |
| **备注** | **无** |

#### getDBNum\_real

|  |  |
| --- | --- |
| **函数原型** | **int getDBNum\_real(const char \*sql\_key)** |
| **功能说明** | **获取sql所在库的标签（若db\_no为空，自动匹配默认库配置defaultDBNum）** |
| **参数说明** | **char \*sql\_key ：sql唯一标示** |
| **返回值** | **数据库标签** |
| **备注** | **无** |

### 性能日志

**明细日志**：ibear\_performance\_0\_PERFORMANCE\_20180830165415.log

type|sql [耗时]

type：MEM缓存；DB：查库；ERR：报错

**汇总日志**：ibear\_performance\_0\_STAT\_20180830165415.log

[type|sql]总耗时|执行次数|平均耗时|最大耗时|最小耗时

ps：涉及到的耗时单位均为微秒；

## ibear环境搭建及DEMO

### 环境搭建

**1．需要配置DSCI环境变量（若主机已经搭建了DSCI则忽略该步骤）：**

export DSCI\_PATH=dsci路径

export LD\_LIBRARY\_PATH=$DSCI\_PATH/lib:$LD\_LIBRARY\_PATH

到dsci/src目录下，执行sh make\_all（由于dsci和主机安装的数据库有关，若为多台云机搭建环境，无需重新编译）

*注：dsci详细信息请参见ibear\_all/dsci/doc/数据服务中心DSCI接口函数.pdf*

**2．需要配置IBEAR环境变量（如果使用静态库，无需配置环境变量）**

export IBF\_PATH=安装路径

export LD\_LIBRARY\_PATH=$IBF\_PATH/ibear\_all/ibear/lib:$LD\_LIBRARY\_PATH

**3．需要生成本地数据库密钥及数据库配置：**

ibear\_all/dsci/lib/dsci\_tools用于生成该框架需要使用的密钥文件**（若使用自定义解密方式，可忽略此步骤）**

ibear\_all/dsci/cfg/db.cfg用于配置需要链接的数据库信息；

**4．使用sql性能分析工具：**

**需要部署ifile、ilog工具，**

export LD\_LIBRARY\_PATH=$IBF\_PATH/ifile\_all/ifile/lib:$LD\_LIBRARY\_PATH

如果需要重新编译动态库：到ifile\_all/ifle/src目录下，执行sh make\_all

export LD\_LIBRARY\_PATH=$IBF\_PATH/ilog\_all/ilog/lib:$LD\_LIBRARY\_PATH

如果需要重新编译动态库：到ilog\_all/ilog/src目录下，执行sh make\_all

**详见ifile、ilog文档；**

### demo使用

编译demo：在demo路径下，执行make

运行demo：demo ./sqlMapConfig.xml

### ibear\_tools工具

编译后生成ibear\_tools，在ibear\_all/ibear/lib/目录下。

展示版本信息：ibear\_tools -v