业务封装技术BizWrapper开发指南和示例

本文先是介绍业务封装技术BizWrapper，并简要分析利用该技术带来的收益，然后提供开发指南和开发示例，本文旨在通过循序渐进和示例演示的方式，为BizWrapper的入门级开发者提供导引和降低心智负担。

一、业务封装技术BizWrapper技术是什么

BizWrapper是SinoDB业务封装技术的简称，BizWrapper是一种Server端功能扩展和性能提升方法，该方法支持开发和安装BizWrapper模块，并把模块以插件形式安装到SinoDB Server，进而提供给客户端（文中的客户端即访问数据库的应用软件）访问。

BizWrapper的物理表现称之为BizWrapper模块，其中包含SQL脚本，编译好的动态库或者jar文件。相较于存储过程等传统扩展方式，更为底层和靠近数据库内核，从而可以更大程度使用Server的扩展能力，并最大程度提高性能。

下文中BizWrapper技术、BizWrapper、BizWrapper模块和模块都会出现。

二、BizWrapper带来的主要收益

BizWrapper技术扩展数据库的主要收益如下：

1-更好的性能表现

- 自定义函数有利于更快的查询计划

  用户自定义函数是查询优化器友好的，这有利于生成更快的查询计划。

- 自定义函数有利于降低网络传输量
  当我们使用用户自定义函数时，被处理的对象不需要经过网络传输，从而有效降低网络传输量。
- 自定义索引有利于提升查询速度
  当数据是音频，图像等复杂数据的时候，可以对其特征进行索引，当查询和匹配图像时候，通过索引比对会比二进制图像比对的运行速度更快。

2-更简单的客户端应用开发

- BizWrapper模块易于组合
  在一个数据库实例里面，可以方便地组合处理各种数据类型的多个BizWrapper模块，然后用一个客户端应用程序集成所有数据。
  比如，一个自媒体需要集成视频，图像，音频或者文字数据到程序中，如果在服务端把每种数据的Bizwrapper模块组合起来处理各种数据类型，就可以简化客户端的开发。
- BizWrapper模块易于升级
  变更BizWrapper模块的时候，不需要重新编译整个客户端应用程序，整个变化由sinodb server管理和处理。

3-更好的事务控制、数据备份和恢复

BizWrapper模块安装和注册后，成为sinodb数据库的一个部件。这样以来，就获得了数据库备份，事务回滚和数据恢复等数据库基础设施的支持。

比如，我们可以安全的把存储此前以文件形式存储的数据，用智能大对象（Smart Large Objects）存储起来，从而获得这些额外的好处。

三、BizWrapper模块带来的其他收益

使用BizWrapper模块将带来如下附加收益：

1-生命周期控制

BizWrapper模块包含所有相关的特定功能的组合，使得我们安装，升级，卸载整个模块。

模块可以进行单独商业分发或者内部分发，可以独立于整个系统以补丁形式分发。

2-代码复用

BizWrapper模块间可以通过接口（Interface）相互引用，当模块A包含了模块B提供的接口时，系统自动建立模块A对模块B的依赖关系，要求模块B先于模块A安装（模块B可以自动安装），从而实现对模块B代码的复用。

四、前置能力训练之数据类型使用

1-常用类型

1.1-Boolean data

BOOLEAN

- 示范代码
  以下代码对BOOLEAN进行简单验证：

```
    create Table BooleanTable(col1  boolean);
    insert into BooleanTable values('F');
    insert into BooleanTable values('T');
    select * from BooleanTable;
```

1.2-Character data

CHAR(n) NVARCHAR(m,n)
NCHAR(n) VARCHAR(m,n)
VARCHAR(m) LVARCHAR(m)

- 示范代码
  以下代码对6种数据类型进行简单验证：

```
    create Table CharactorTable(col1 char(5), col2 nvarchar(5,1),col3 nchar(5),
col4 varchar(5,1), col5 varchar(255), col6 lvarchar(4096));
    insert into CharactorTable values('col1','col2','col3','col4','col5','col6');
    select * from CharactorTable;
```

1.3-Numeric data1

REAL(=SMALLFLOAT) FLOAT(n)
DOUBLE PRECISION INT8(=BIGINT)
INTEGER(=INT) SMALLINT
DECIMAL(p,s)(=dec) MONEY(p,s)

- 示范代码
  以下代码对8种数据类型进行简单验证：

```
    create Table Numberic1Table(col1 real, col2 float,col3 DOUBLE PRECISION , col4
int8, col5 integer, col6 smallint, col7 dec(10,2), col8 money(18,2));
    insert into Numberic1Table
values(1.23,1.234,1.2345,1234456780,12344567,32767,1.23,12345678901.23);
    select * from Numberic1Table;
```

1.4-Numeric data2

SERIAL, SERIAL8(=BIGSERIAL)

- 示范代码
  以下代码对2种数据类型进行简单验证：

```
    create Table Numberic2Table(id serial);
    insert into Numberic2Table values(0);
    select * from Numberic2Table;
    create Table Numberic2Table2(id serial8);
    insert into Numberic2Table2 values(0);
    select * from Numberic2Table2;
```

1.5-Date, time, and interval data

DATE DATETIME INTERVAL

- 示范代码
  以下代码对3种数据类型进行简单验证：

```
    export DBDATE=Y4MD- # "YYYY-MM-DD"
    export DBTIME='%Y-%m-%d %H:%M:%S.%F5' # "YYYY-MM-DD hh:mm:ss.fffff"
```

```
    create Table DtiTable(col1 date, col2 datetime year to fraction(5),col3
interval hour(5) TO fraction(5));
    insert into DtiTable values('2024-07-01','2024-07-01
10:11:12.12345','10:11:12.12345');
    select * from DtiTable;
```

1.6-Simple Large Object

TEXT BYTE

- 示范代码
  以下代码对2种数据类型进行简单验证：

```
    echo 'ABC DEF' > /tmp/text.load
    echo '41424320444546' >/tmp/byte.load
```

```
    create Table TextTable(col1 text);
    load from '/tmp/text.load' insert into TextTable;
    select * from TextTable;
    unload to '/tmp/text.un' select * from TextTable;
    create Table ByteTable(col1 byte);
    load from '/tmp/byte.load' insert into ByteTable;
    select * from ByteTable;
    unload to '/tmp/byte.un' select * from ByteTable;
```

1.7-Smart Large Object

CLOB BLOB

- 示范代码
  以下代码对2种数据类型进行简单验证：

```
    onstat -g cfg|grep SBSPACENAME #确认SBSPACENAME 有值
    echo 'ABC DEF' > /tmp/clob.txt
    echo -n -e '\x01\x02\x03\x04\x05\x06' > /tmp/blob.bin
```

```
    create Table ClobTable(col1  clob);
    insert into ClobTable values (filetoclob('/tmp/clob.txt','client'));
    select * from ClobTable;
    select lotofile(col1, '/tmp/clob.out','client') from ClobTable; -- ex.
/tmp/clob.out.00000000663c9cdd
    create Table BlobTable(col1  blob);
    insert into BlobTable values (filetoblob('/tmp/blob.bin','client'));
    select * from BlobTable;
    select lotofile(col1, '/tmp/blob.out','client') from BlobTable; -- ex.
/tmp/blob.out.00000000663c9cde
```

```
    ls /tmp/clob.out.* # ex. /tmp/clob.out.00000000663c9cdd
    ls /tmp/blob.out.* # ex. /tmp/blob.out.00000000663c9cde
```

2-限定类型

限定类型在基本数据类型上追加限定词，限定词确定了存储大小，取值范围和类型精度。如DECIMAL(p,s) 中的p表示精度，s表示小数点后面数字。 这样，DECIMAL(6,3) 有6位精度，小数点后面3位，典型数值为123.456.

3-无名透明类型

3.1-COLLECTION类型：LIST，SET，MULTISET

- 示范代码

```
    create Table ListTable(col1 list(int not null));
    insert into ListTable values(list{1,2,3});
    select * from ListTable;
    -- select col1[1] from ListTable; -- 305: Subscripted column (col1) is not of
type CHAR, VARCHAR, TEXT nor BYTES.
    -- LIST,SET,MULTISET不支持SQL下标[]访问其中的元素。
    -- 只有CHAR, VARCHAR, TEXT, BYTES 会支持SQL下标。

    create Table SetTable(col1 set(int not null));
    insert into SetTable values(set{1,2,3});
    insert into SetTable values(set{1,2,3,4,4});
    select * from SetTable;

    create Table MultisetTable(col1 multiset(int not null));
    insert into MultisetTable values(multiset{1,2,3});
    insert into MultisetTable values(multiset{1,2,3,4,4});
    select * from MultisetTable;
```

3.2-UNAMED ROW类型

- 示范代码

```
    create Table UnamedrowTable(col1 row(a int,b char(2)));
    insert into UnamedrowTable values(row(1,'12'));
    select * from UnamedrowTable;
```

4-具名透明类型

用户自定义数据类型，其内部结构对SQL层可见并且可以访问。

4.1-DISTINCT 类型

distinct类型与源类型是不同的类型，虽然继承源类型上的操作，但双向类型转换均需要显式操作。

4.2-NAMED ROW 类型

以下代码对2种类型进行简单验证：

```
    onstat -g cfg|grep SBSPACENAME #确认SBSPACENAME 有值
    echo -n -e '\x01\x02\x03\x04\x05\x06' > /tmp/fakeblob.mp3
```

```
    create distinct type audiofile as blob;
    create row type audiotype (format char(3) not null, filesizebytes int not null,
filecontent audiofile);
    create Table NamedTable(col1 serial, audio audiotype);
    insert into NamedTable values(0, row('mp3',6,
filetoblob('/tmp/fakeblob.mp3','client')::audiofile)::audiotype);
    select * from NamedTable;
    select col1, audio.format,audio.filesizebytes,audio.filecontent from NamedTable;
    select lotofile(audio.filecontent::blob, '/tmp/audiofile.out','client') from
NamedTable; -- ex. /tmp/audiofile.out.00000000663c9cdf
```

5-不透明数据类型

用户自定义数据类型，其内部结构对SQL层来说不可见且不可以访问，这样提供了最大封装能力，但其SQL输入输出需要通过自定义函数完成定义, 开发难度也相应更高。

BOOLEAN, BSON, JSON, and LVARCHAR 等类型是SinoDB采用不透明数据类型来完成实现的，不透明数据类型不能仅仅通过SQL实现。

五、前置能力训练之模块试用

1-获取BizWrapper模块sample001

2-确认sinodbms的数据库环境

使用onstat - 确认数据库是在线状态，看到On-Line字样，如下所示：

```
$ onstat -
Your evaluation license will expire on 2025-07-01 00:00:00

Sinoregal SinoDB Dynamic Server Version 16.8.FC8U0X3TL -- On-Line -- Up 16:50:24 --
327304 Kbytes
```

3-上传模块到sinodbms用户的预备目录

upload sample001.1.0.tar to ~sinodbms/installmedia

4-解压到SINODBMSDIR的extend目录

```
cd $SINODBMSDIR/extend && tar xvf ~sinodbms/installmedia/sample001.1.0.tar
```

5-创建测试数据库

```
export DB_LOCALE=zh_cn.57372
export CLIENT_LOCALE=${DB_LOCALE}
export SERVER_LOCALE=${DB_LOCALE}
dbaccess - -
```

```
 create database sample001 with log;
```

6-注册模块

```
dbaccess sample001 -
```

```
  execute function sysbldprepare('sample001.1.0','create');
```

7-执行函数

```
execute function helloworld();
```

8-卸载模块

```
execute function sysbldprepare('sample001.1.0','drop');
```

9-删除模块

```
cd $SINODBMSDIR/extend && rm -r sample001.1.0;
```

六、模块开发过程指南

1-设计过程

设计用来确定BizWrapper模块的高层数据模型，该模型代表所要关注的领域的对象和操作。

有如下四点设计原则：

6.1-分离出界面和数据上的抽象操作，关注抽象操作和避免界面元素干扰

6.2-考虑关注的数据类型和作用于它们的操作例程

6.3-简化模块设计，交给客户端去组合

6.4-模块要考虑为多个应用而提供服务而非单一应用

比如空数据类型模块需要考虑的对象如多边形，线段；对象的操作有空间包含（within），空间交错(intersect)等，实现“找到当前可见区域内的多边形，而这个区域用一个多边形表示”。

然而，模型不含“显示逻辑”，多边形的绘制属于界面，由客户端应用（访问数据库的应用称之为客户端）负责。

这里有很重要的一个设计理念，模型内聚关注领域内的某个特定方向的功能，其使用者是“其他开发人员”，即客户端软件开发人员。

2-编码过程

开发过程实现设计时提出的功能与非功能目标，开发环境包括一套SinoDB 16.8数据库，Linux c/c++编程环境，Linux gdb调试环境。

2.1-工程目录结构

1. sample001/
2. ……/scripts/
3. ……/prepare.sql
4. ……/objects.sql
5. ……/src/
6. ……/c/
7. ……/udr.c
8. ……/support.c
9. ……/sample001.h
10. ……/src/linux86_64/
11. ……/stage/
12. ……/extend/
13. ……/sample001.1.0/
14. ……/objects.sql
15. ……/prepare.sql
16. ……/sample001.bld
17. ……/installmedia/
18. ……/sample001.1.0.biz.tar
19. ……/support.o
20. ……/udr.o
21. ……/sample001.bld
22. sample001/src/Makefile

2.2-目录和文件说明

1~9 源代码，10～21 编译中间文件和输出的BizWrapper模块安装文件， 22 构建文件Makefile

1. 工程目录, 顶级目录，存放SQL脚本，C源代码，构建文件，中间文件，安装文件。
2. SQL脚本目录, 存放SQL脚本，多语种支持文件。
3. 模块的注册，升级用管理信息，以及对objects.sql的入口调用。
4. 模块内部的类型，函数的对数据库的接口，在模块被注册，使用，注销时调用。
5. C源代码，构建文件，中间文件，安装文件的目录。
6. C源代码目录
7. C源文件，存放udr函数
8. udr函数实现时需要的支持函数，如trace, debug等
9. C头文件
10. 中间文件顶级目录
11. 安装打包用的文件目录
12. 对应sinodbms的标准模块安装目录
13. 对应模块的名字和版本，全局唯一
14. 对应4
15. 对应3
16. 链接后的动态库文件
17. 存放安装文件
18. BizWrapper模块安装文件,分发用
19. 编译目标文件
20. 编译目标文件
21. 链接后的动态库文件
22. 构建用Makefile

3-调试过程

3.1-全量编译调试

全量编译会执行make clean debug install，会清理下中间文件，在编译过程中加入Trace信息和调试信息“-DTRACE_DEBUG_sample001 -g ”，然后把模块复制到数据库的扩展插件目录，并为使用gdb做好准备。

- 调试构建命令

如下两个命令是等价的，后者是前者的缩写:

```
cd sample001/src && make all3
```

或者

```
cd sample001/src && make d
```

- 编译输出

  输出如下：

```
[sinodbms@SINOBOX src]$ pwd
/home/sinoregal/installmedia/sample001/src
[sinodbms@SINOBOX src]$ make all3
rm -f  /home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o
make server -C /home/sinoregal/installmedia/sample001/src/ COPTS=" -
DTRACE_DEBUG_sample001 -g "
make[1]: Entering directory `/home/sinoregal/installmedia/sample001/src'
cc -DMI_SERVBUILD -fPIC -I/opt/sinodb_sinodbms/incl/public -
I/opt/sinodb_sinodbms/incl/esql -I/opt/sinodb_sinodbms/incl -DTRACE_DEBUG_sample001
-g  -o /home/sinoregal/installmedia/sample001/src/linux86_64/support.o -c
/home/sinoregal/installmedia/sample001/src/c/support.c
cc -DMI_SERVBUILD -fPIC -I/opt/sinodb_sinodbms/incl/public -
I/opt/sinodb_sinodbms/incl/esql -I/opt/sinodb_sinodbms/incl -DTRACE_DEBUG_sample001
-g  -o /home/sinoregal/installmedia/sample001/src/linux86_64/udr.o -c
/home/sinoregal/installmedia/sample001/src/c/udr.c
cc -shared -m64 -Bsymbolic -o
/home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld              \
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o             \
2> link.errs
make[1]: Leaving directory `/home/sinoregal/installmedia/sample001/src'
cp -p /home/sinoregal/installmedia/sample001/src/../scripts/objects.sql
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cp -p /home/sinoregal/installmedia/sample001/src/../scripts/prepare.sql
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cp -p /home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cd /home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend && tar cvf
/home/sinoregal/installmedia/sample001/src/linux86_64/installmedia/sample001.1.0.bi
z.tar sample001.1.0
sample001.1.0/
sample001.1.0/objects.sql
sample001.1.0/prepare.sql
sample001.1.0/sample001.bld
cd /opt/sinodb_sinodbms/extend &&  tar xvf
/home/sinoregal/installmedia/sample001/src/linux86_64/installmedia/sample001.1.0.bi
z.tar
sample001.1.0/
sample001.1.0/objects.sql
sample001.1.0/prepare.sql
sample001.1.0/sample001.bld
```

```
make clean debug install finished!
you could debug using following command from current shell:
sudo gdb $SINODBMSDIR/bin/oninit -p 7088;
```

- 使用gdb进入调试

  正常的编译输出的最后一行，会出现如下字样提示，其中的'7088'是sinodb数据库服务的进程号，会随着环境不同而变化：

  ```
  sudo gdb $SINODBMSDIR/bin/oninit -p 7088
  ```

  当sinodb数据库经过重启后，可以前面的构建命令再次获得这个进程号，也可以通过如下命令获得:

  ```
  onstat -g glo 2>/dev/null |grep cpu| tail -1|awk '{print $2}'
  ```

  - 首先开启终端T1, 注册数据库如下：

```
[sinodbms@SINOBOX ~]$ dbaccess sample001 -
Your evaluation license will expire on 2025-07-01 00:00:00

Database selected.

> execute function sysbldprepare('sample001.1.0','create');


(expression)


(U0001) - registerBlade - sample001.1.0 is already registered
Error in line 1
Near character position 56
> execute function sysbldprepare('sample001.1.0','drop');


(expression)

            0

1 row(s) retrieved.

> execute function sysbldprepare('sample001.1.0','create');


(expression)

            0

1 row(s) retrieved.
```

- 其次开启终端T2，设置断点如下：

```
[sinodbms@SINOBOX src]$ sudo gdb $SINODBMSDIR/bin/oninit -p 7088
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-120.el7
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /opt/sinodb_sinodbms/bin/oninit...(no debugging symbols
found)...done.
Attaching to program: /opt/sinodb_sinodbms/bin/oninit, process 7088
Reading symbols from /lib64/libpthread.so.0...Reading symbols from
/usr/lib/debug/usr/lib64/libpthread-2.17.so.debug...done.
done.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Loaded symbols for /lib64/libpthread.so.0
Reading symbols from /lib64/libdl.so.2...Reading symbols from
/usr/lib/debug/usr/lib64/libdl-2.17.so.debug...done.
done.
Loaded symbols for /lib64/libdl.so.2
Reading symbols from /lib64/libcrypt.so.1...Reading symbols from
/usr/lib/debug/usr/lib64/libcrypt-2.17.so.debug...done.
done.
Loaded symbols for /lib64/libcrypt.so.1
Reading symbols from /lib64/libpam.so.0...Reading symbols from
/usr/lib/debug/usr/lib64/libpam.so.0.83.1.debug...done.
done.
Loaded symbols for /lib64/libpam.so.0
Reading symbols from /lib64/libstdc++.so.6...Reading symbols from
/usr/lib/debug/usr/lib64/libstdc++.so.6.0.19.debug...done.
done.
Loaded symbols for /lib64/libstdc++.so.6
Reading symbols from /lib64/libm.so.6...Reading symbols from
/usr/lib/debug/usr/lib64/libm-2.17.so.debug...done.
done.
Loaded symbols for /lib64/libm.so.6
Reading symbols from /lib64/libgcc_s.so.1...Reading symbols from
/usr/lib/debug/usr/lib64/libgcc_s-4.8.5-20150702.so.1.debug...done.
done.
Loaded symbols for /lib64/libgcc_s.so.1
Reading symbols from /lib64/libc.so.6...Reading symbols from
/usr/lib/debug/usr/lib64/libc-2.17.so.debug...done.
done.
```

```
Loaded symbols for /lib64/libc.so.6
Reading symbols from /lib64/ld-linux-x86-64.so.2...Reading symbols from
/usr/lib/debug/usr/lib64/ld-2.17.so.debug...done.
done.
Loaded symbols for /lib64/ld-linux-x86-64.so.2
Reading symbols from /lib64/libfreebl3.so...Reading symbols from
/lib64/libfreebl3.so...(no debugging symbols found)...done.
(no debugging symbols found)...done.
Loaded symbols for /lib64/libfreebl3.so
Reading symbols from /lib64/libaudit.so.1...Reading symbols from
/usr/lib/debug/usr/lib64/libaudit.so.1.0.0.debug...done.
done.
Loaded symbols for /lib64/libaudit.so.1
Reading symbols from /lib64/libcap-ng.so.0...Reading symbols from
/usr/lib/debug/usr/lib64/libcap-ng.so.0.0.0.debug...done.
done.
Loaded symbols for /lib64/libcap-ng.so.0
Reading symbols from /lib64/libnss_files.so.2...Reading symbols from
/usr/lib/debug/usr/lib64/libnss_files-2.17.so.debug...done.
done.
Loaded symbols for /lib64/libnss_files.so.2
Reading symbols from /lib64/libnss_sss.so.2...Reading symbols from
/usr/lib/debug/usr/lib64/libnss_sss.so.2.debug...done.
done.
Loaded symbols for /lib64/libnss_sss.so.2
Reading symbols from /opt/sinodb_sinodbms/gls/dll/64-libicudata.so.48...(no
debugging symbols found)...done.
Loaded symbols for /opt/sinodb_sinodbms/gls/dll/64-libicudata.so.48
Reading symbols from /opt/sinodb_sinodbms/gls/dll/64-libicuuc.so.48...(no debugging
symbols found)...done.
Loaded symbols for /opt/sinodb_sinodbms/gls/dll/64-libicuuc.so.48
Reading symbols from /opt/sinodb_sinodbms/gls/dll/64-libicui18n.so.48...(no
debugging symbols found)...done.
Loaded symbols for /opt/sinodb_sinodbms/gls/dll/64-libicui18n.so.48
Reading symbols from /opt/sinodb_sinodbms/gls/dll/64-libglu.so.48...(no debugging
symbols found)...done.
Loaded symbols for /opt/sinodb_sinodbms/gls/dll/64-libglu.so.48
Reading symbols from /lib64/libaio.so.1...Reading symbols from
/usr/lib/debug/usr/lib64/libaio.so.1.0.1.debug...done.
done.
Loaded symbols for /lib64/libaio.so.1
Reading symbols from /opt/sinodb_sinodbms/extend/ifxmngr/ifxmngr.bld...done.
Loaded symbols for /opt/sinodb_sinodbms/extend/ifxmngr/ifxmngr.bld
Reading symbols from
/opt/sinodb_sinodbms/extend/sample001.1.0/sample001.bld...done.
Loaded symbols for /opt/sinodb_sinodbms/extend/sample001.1.0/sample001.bld
0x00007ff77660e307 in semop () at ../sysdeps/unix/syscall-template.S:81
81      T_PSEUDO (SYSCALL_SYMBOL, SYSCALL_NAME, SYSCALL_NARGS)
```

```
Missing separate debuginfos, use: debuginfo-install nss-softokn-freebl-3.90.0-
6.el7_9.x86_64
(gdb) b helloworld
Breakpoint 1 at 0x7ff76dd779be: file
/home/sinoregal/installmedia/sample001/src/c/udr.c, line 73.
```

- 切换到终端T1，执行命令如下，保持阻塞状态：

```
> execute function helloworld();
```

- 切换到终端T2，执行命令如下：

```
(gdb) c
Continuing.

Breakpoint 1, helloworld (Gen_fparam=0x47ba1440) at
/home/sinoregal/installmedia/sample001/src/c/udr.c:73
73              DBDK_TRACE_ENTER("helloworld")
(gdb) l
68              mi_lvarchar *        Gen_RetVal;      /* The return value.
             */
69              MI_CONNECTION *      Gen_Con;         /* The connection handle.
            */
70              mi_integer           Gen_ReturnSize;  /* Return value is this size.
            */
71
72              gl_mchar_t *       Gen_RetData;   /* Store the return data here.
     */
73              DBDK_TRACE_ENTER("helloworld")
74              /* Use the NULL connection. */
75              Gen_Con = NULL;
76              char HelloString[] = "Hello World!";
77
(gdb) det
Detaching from program: /opt/sinodb_sinodbms/bin/oninit, process 7088
[Inferior 1 (process 7088) detached]
(gdb) quit
```

请注意在gdb中正常detach，这样有助于保持服务器进程处于正常状态。

- 切换到终端T1，可以观察到阻塞状态解除，输出如下：

```
>  execute function helloworld();



(expression)  Hello World!

1 row(s) retrieved.
```

3.2-增量编译调试

增量编译会执行make debug install，在编译过程中加入Trace信息和调试信息“-DTRACE_DEBUG_sample001 -g ”，然后把模块复制到数据库的扩展插件目录，并为使用gdb做好准备。

- 调试构建命令

如下两个命令是等价的，后者是前者的缩写:

```
cd sample001/src && make all4
```

或者

```
cd sample001/src && make d2
```

- 输出

  与3.1-全量编译调试中的编译输出相比较，除了编译命令差异外，还少了如下删除中间文件的命令。

```
rm -f  /home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o
```

4-打包发布

4.1-正常发布

正常发布会执行make clean release install，会清理下中间文件，在编译过程中加入优化参数“-O3 ”，生成安装媒体文件，然后把模块复制到数据库的扩展插件目录。

- 正常发布构建命令

如下两个命令是等价的，后者是前者的缩写:

```
cd sample001/src && make all
```

或者

```
cd sample001/src && make r
```

- 观察编译输出

  输出如下：

```
[sinodbms@SINOBOX src]$ pwd
/home/sinoregal/installmedia/sample001/src
[sinodbms@SINOBOX src]$ make all
rm -f  /home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o
make server -C /home/sinoregal/installmedia/sample001/src/ COPTS=" -O3 "
make[1]: Entering directory `/home/sinoregal/installmedia/sample001/src'
cc -DMI_SERVBUILD -fPIC -I/opt/sinodb_sinodbms/incl/public -
I/opt/sinodb_sinodbms/incl/esql -I/opt/sinodb_sinodbms/incl -O3  -o
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o -c
/home/sinoregal/installmedia/sample001/src/c/support.c
cc -DMI_SERVBUILD -fPIC -I/opt/sinodb_sinodbms/incl/public -
I/opt/sinodb_sinodbms/incl/esql -I/opt/sinodb_sinodbms/incl -O3  -o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o -c
/home/sinoregal/installmedia/sample001/src/c/udr.c
cc -shared -m64 -Bsymbolic -o
/home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld               \
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o             \
2> link.errs
make[1]: Leaving directory `/home/sinoregal/installmedia/sample001/src'
cp -p /home/sinoregal/installmedia/sample001/src/../scripts/objects.sql
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cp -p /home/sinoregal/installmedia/sample001/src/../scripts/prepare.sql
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cp -p /home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cd /home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend && tar cvf
/home/sinoregal/installmedia/sample001/src/linux86_64/installmedia/sample001.1.0.biz.ta
r sample001.1.0
sample001.1.0/
sample001.1.0/objects.sql
sample001.1.0/prepare.sql
sample001.1.0/sample001.bld
cd /opt/sinodb_sinodbms/extend &&  tar xvf
/home/sinoregal/installmedia/sample001/src/linux86_64/installmedia/sample001.1.0.biz.ta
r
sample001.1.0/
sample001.1.0/objects.sql
sample001.1.0/prepare.sql
sample001.1.0/sample001.bld
make clean release install finished!
```

```
[sinodbms@SINOBOX src]$
```

其中sample001/src/linux86_64/installmedia/sample001.1.0.biz.tar 就是release版本的安装文件。

4.2-带Trace发布

带Trace发布会执行make clean release_trace install，会清理下中间文件，在编译过程中加入优化参数“-O3 ”，生成安装媒体文件，然后把模块复制到数据库的扩展插件目录。

- 正常发布构建命令

如下两个命令是等价的，后者是前者的缩写:

```
cd sample001/src && make all2
```

或者

```
cd sample001/src && make r2
```

- 观察编译输出

```
[sinodbms@SINOBOX src]$ pwd
/home/sinoregal/installmedia/sample001/src
[sinodbms@SINOBOX src]$ make all2
rm -f  /home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o
make server -C /home/sinoregal/installmedia/sample001/src/ COPTS=" -
DTRACE_DEBUG_sample001 -O3 "
make[1]: Entering directory `/home/sinoregal/installmedia/sample001/src'
cc -DMI_SERVBUILD -fPIC -I/opt/sinodb_sinodbms/incl/public -
I/opt/sinodb_sinodbms/incl/esql -I/opt/sinodb_sinodbms/incl -DTRACE_DEBUG_sample001 -O3
 -o /home/sinoregal/installmedia/sample001/src/linux86_64/support.o -c
/home/sinoregal/installmedia/sample001/src/c/support.c
cc -DMI_SERVBUILD -fPIC -I/opt/sinodb_sinodbms/incl/public -
I/opt/sinodb_sinodbms/incl/esql -I/opt/sinodb_sinodbms/incl -DTRACE_DEBUG_sample001 -O3
 -o /home/sinoregal/installmedia/sample001/src/linux86_64/udr.o -c
/home/sinoregal/installmedia/sample001/src/c/udr.c
cc -shared -m64 -Bsymbolic -o
/home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld               \
/home/sinoregal/installmedia/sample001/src/linux86_64/support.o
/home/sinoregal/installmedia/sample001/src/linux86_64/udr.o             \
2> link.errs
make[1]: Leaving directory `/home/sinoregal/installmedia/sample001/src'
cp -p /home/sinoregal/installmedia/sample001/src/../scripts/objects.sql
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cp -p /home/sinoregal/installmedia/sample001/src/../scripts/prepare.sql
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
```

```
cp -p /home/sinoregal/installmedia/sample001/src/linux86_64/sample001.bld
/home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend/sample001.1.0
cd /home/sinoregal/installmedia/sample001/src/linux86_64/stage/extend && tar cvf
/home/sinoregal/installmedia/sample001/src/linux86_64/installmedia/sample001.1.0.biz.ta
r sample001.1.0
sample001.1.0/
sample001.1.0/objects.sql
sample001.1.0/prepare.sql
sample001.1.0/sample001.bld
cd /opt/sinodb_sinodbms/extend &&  tar xvf
/home/sinoregal/installmedia/sample001/src/linux86_64/installmedia/sample001.1.0.biz.ta
r
sample001.1.0/
sample001.1.0/objects.sql
sample001.1.0/prepare.sql
sample001.1.0/sample001.bld
make clean release_trace install finished!
[sinodbms@SINOBOX src]$
```

其中sample001/src/linux86_64/installmedia/sample001.1.0.biz.tar 就是release版本的安装文件。

5-部署/注册

部署过程把安装包解压到SinoDB的插件目录($SINODBMSDIR/extend)，注册则负责把模块加载到特定database，完成注册后，就可以使用模块提供的扩展功能。

5.1-部署

- 自动部署

  调试过程和打包发布过程都会自动完成部署过程，部署目标为开发时的SinoDB服务器。

- 手工部署

  当部署目标不是开发时的数据库服务器时，则需要采用手工部署方式：

  - 获取4.1或者4.2中的安装文件sample001.1.0.biz.tar
  - 传输安装文件到目标服务器
  - 使用sinodbms操作系统用户，解压sample001.1.0.biz.tar到目标服务器的$INFORMIXDIR/extend目录

5.2-注册/删除

参考五、前置能力训练之模块试用，完成模块的注册，使用和删除。

6-查看TRACE(可选)

make all2/r2/all3/d/all4/d2 的输出都带有Trace log功能，可以查看log，观察函数的进入和退出。

log内容由两个宏(DBDK_TRACE_ENTER DBDK_TRACE_EXIT)的输出产生，参见udr.c文件可看到如下代码：
DBDK_TRACE_ENTER("helloworld")
DBDK_TRACE_EXIT("helloworld")

6.1-打开Trace的步骤

Trace默认关闭，需要执行dbaccess sample001 - 后，执行如下步骤：

```
delete from systraceclasses where name='sample001';
insert into systraceclasses(name) values('sample001');
execute function sysbldprepare('TimeSeries.6.00.FC7','create');
execute function TSSetTraceLevel('sample001 20');
execute function TSSetTraceFile('/tmp/sample001.trc');
execute function sysbldprepare('TimeSeries.6.00.FC7','drop');
execute function helloworld();
```

- 说明

TSSetTraceLevel和TSSetTraceFile函数是打开Trace需要的，而这两个函数由TimeSeries.6.00.FC7 模块提供，所以注册了时间序列模块。

打开Trace仅仅在当前会话有效期内生效，不是永久的。

6.2-命令输出

```
dbaccess sample001 -
Your evaluation license will expire on 2025-07-01 00:00:00

Database selected.
```

```
> delete from systraceclasses where name='sample001';

1 row(s) deleted.

> insert into systraceclasses(name) values('sample001');

1 row(s) inserted.

> execute function sysbldprepare('TimeSeries.6.00.FC7','create');


(expression)

           0

1 row(s) retrieved.

> execute function TSSetTraceLevel('sample001 20');


(expression)

           0
```

```
1 row(s) retrieved.

> execute function TSSetTraceFile('/tmp/sample001.trc');


(expression)

            0

1 row(s) retrieved.

> execute function sysbldprepare('TimeSeries.6.00.FC7','drop');


(expression)

            0

1 row(s) retrieved.

> execute function helloworld();



(expression)  Hello World!

1 row(s) retrieved.
```

6.3-查看结果

```
[sinodbms@SINOBOX c]$ cat /tmp/sample001.trc

============================================================

Tracing session: 37 on 07/02/2024

21:27:18  Entering function helloworld
(/home/sinoregal/installmedia/sample001/src/c/udr.c).
21:27:18  Successfully exiting helloworld
(/home/sinoregal/installmedia/sample001/src/c/udr.c).

============================================================

Tracing session: 39 on 07/02/2024
```

```
21:59:05  Entering function helloworld
(/home/sinoregal/installmedia/sample001/src/c/udr.c).
21:59:05  Successfully exiting helloworld
(/home/sinoregal/installmedia/sample001/src/c/udr.c).
[sinodbms@SINOBOX c]$
```

七、模块开发示例

1-基本类型限定类型与UDR结合

- sample101: 输入和返回4字节（含）以内的整数
- sample102: 输入和返回BIGINT
- sample103: 输入和返回DOUBLE PRECISION
- sample104: 合并sample101~103
- sample105: 参数含BLOB类型
- sample106: 参数含CLOB类型

2-复杂类型与UDR结合

复杂类型主要包括COLLECTION类型：LIST，SET，MULTISET和UNAMED ROW类型。

- sample201: 参数含LIST类型
- sample202: 参数含SET类型
- sample203: 参数含MULTISET类型
- sample204: 参数含UNAMED ROW类型

3-透明类型与UDR结合

- sample301: 参数含DISTINCT类型
- sample302: 参数含NAMED ROW类型
- sample303: 参数含NAMED ROW类型，NAMED ROW嵌套BLOB类型

4-不透明类型与扩展索引

BizWrapper的BTREE算法和RTREE算法都预留了扩展接口，自定义不透明数据类型可以扩展BTREE索引，以及RTREE索引。

- 通过重定义不透明类型上的比较函数，可以建立BTREE索引
- 通过重定义不透明类型上的支持（Support）函数和策略（Strategy）函数，可以建立BTREE索引

5-自定义表与索引

BizWrapper提供虚表VTI接口（Virtual Table Interface）和VII接口(Virtual Index Interface), VTI用来使用SQL访问特定格式的存储，VII用来建立索引。

- VTI对数据提供主访问方式（Primary Access Method），实现原始数据的增删改查操作
- VII则提供二级访问方式（Secondary Access Method），通过匹配和索引间接对数据进行访问

八、附录-sample001源代码

1-Makefile

```
# This Makefile builds the sample001 BizWrapper.
space:=
space+=

MAKEFILEPATH := $(subst $(lastword $(notdir $(MAKEFILE_LIST))),,$(subst
$(space),\$(space),$(shell realpath '$(strip $(MAKEFILE_LIST))')))
# check env
ifeq ("x$(SINODBMSDIR)","x")
$(error Please make sure SINODBMSDIR env varialbe is set!)
endif
# You can consult sinoregal for the SINOPLATFORM setting.
SINOPLATFORM = linux86_64
include  $(SINODBMSDIR)/incl/dbdk/makeinc.$(SINOPLATFORM)
SHLIBLOD = $(CC)
# ====================================================================

# This is the project title.
PROJECT_TITLE = sample001
VERSION = 1.0

# The linked BizWrapper module is placed here.
BINDIR = $(MAKEFILEPATH)$(SINOPLATFORM)

# The BizWrapper module component to be packed is placed here.
STAGEDIR = $(BINDIR)/stage
EXTENDDIR = $(STAGEDIR)/extend
COMPONETDIR = $(EXTENDDIR)/$(PROJECT_TITLE).$(VERSION)

# The tar pakage of module is placed here, which is packed file of stage dir .
MEDIADIR = $(BINDIR)/installmedia

# Platform independent code goes here.

MI_INCL    = $(SINODBMSDIR)/incl
CFLAGS     = -DMI_SERVBUILD $(CC_PIC) -I$(MI_INCL)/public -I$(MI_INCL)/esql -I$(MI_INCL)
$(COPTS)
```

```
LINKFLAGS = $(SHLIBLFLAG) $(SYMFLAG)

# Output lib compiled from this project
PROJECT_LIBS = $(BINDIR)/$(PROJECT_TITLE).$(BLDLIB_SUFF)

# Additonal LIBS to be linked into PROJECT_LIBS
LIBS      =

# This is a list of the C object files.
PROJECTC_OBJS = \
  $(BINDIR)/support.$(OBJSUFF) \
  $(BINDIR)/udr.$(OBJSUFF)


.phoney : server debug release package media install clean
.phoney : all all2 all3 all4 r r2 d d2

# release and install from clean
# r is the same as all
all : clean release install
  $(info make clean release install finished!)
r: all

# release with trace  and install from clean
# r2 is the same as all2
all2: clean release_trace install
  $(info make clean release_trace install finished!)
r2: all2

# debug and install from clean
# d is the same as all3

all3: clean all4
d: all3

# debug and install from debug
# d2 is the same as all4
ISONLINE := $(shell onstat - 2>/dev/null |grep On-Line >/dev/null 2>/dev/null; echo
$$?)
ifeq ($(ISONLINE),0)
ISNUMMAX1 := $(shell onstat -g cfg 2>/dev/null |grep "^VPCLASS.*num=1.*max=1"
>/dev/null 2>/dev/null; echo $$?)
ifeq ($(ISNUMMAX1),0)
PID := $(shell onstat -g glo 2>/dev/null |grep cpu| tail -1|awk '{print $$2}')
endif
endif

all4: debug install
ifeq ($(ISONLINE),0)
```

```
ifeq ($(ISNUMMAX1),0)
  $(info make clean debug install finished!)
  $(info you could debug using following command from current shell:)
  $(info sudo gdb $$SINODBMSDIR/bin/oninit -p $(PID); )
else
  $(info -- onstat -g cfg can be used to check if VPCLASS include num=1 and max=1)
  $(error Please make sure VPCLASS                    cpu,num=1,max=1,noage is
configured in $$ONCONFIG file.)
endif
else
  $(error Please make sure sinodbms is On-Line to be able to debug.)
endif

d2: all4
# Construct each object file.

$(BINDIR)/support.$(OBJSUFF) : $(MAKEFILEPATH)c/support.c
$(MAKEFILEPATH)c/$(PROJECT_TITLE).h
  $(CC) $(CFLAGS) -o $@ -c $<

$(BINDIR)/udr.$(OBJSUFF) : $(MAKEFILEPATH)c/udr.c $(MAKEFILEPATH)c/$(PROJECT_TITLE).h
  $(CC) $(CFLAGS) -o $@ -c $<

# Construct the shared library.
$(PROJECT_LIBS) : $(PROJECTC_OBJS)
  $(SHLIBLOD) $(LINKFLAGS) -o $(PROJECT_LIBS)   \
  $(PROJECTC_OBJS) $(LIBS)      \
  2> link.errs
  @if test -x $(SINODBMSDIR)/bin/filtersym.sh ;   \
  then $(SINODBMSDIR)/bin/filtersym.sh<link.errs ;  \
  else cat link.errs ;                            \
  fi ;              \
  $(RM) $(RMFLAGS) link.errs

server : $(PROJECT_LIBS)

FLAG_HEAD = -DTRACE_DEBUG_$(PROJECT_TITLE)
DEBUG_FLAG = $(FLAG_HEAD) -g
RELEASE_FLAG = -O3
RELEASE_TRACE_FLAG = $(FLAG_HEAD) -O3
QUOTES = "

debug  : $(BINDIR)
  $(MAKE) server -C $(MAKEFILEPATH) COPTS=$(QUOTES) $(DEBUG_FLAG) $(QUOTES)

release : $(BINDIR)
  $(MAKE) server -C $(MAKEFILEPATH) COPTS=$(QUOTES) $(RELEASE_FLAG) $(QUOTES)

release_trace: $(BINDIR)
```

```
  $(MAKE) server -C $(MAKEFILEPATH) COPTS=$(QUOTES) $(RELEASE_TRACE_FLAG) $(QUOTES)

package : server $(COMPONETDIR) $(MAKEFILEPATH)../scripts/objects.sql
$(MAKEFILEPATH)../scripts/prepare.sql
  $(CP) $(CPPRESERVE) $(MAKEFILEPATH)../scripts/objects.sql $(COMPONETDIR)
  $(CP) $(CPPRESERVE) $(MAKEFILEPATH)../scripts/prepare.sql $(COMPONETDIR)
  $(CP) $(CPPRESERVE) $(PROJECT_LIBS) $(COMPONETDIR)

media: package $(MEDIADIR)
  cd $(EXTENDDIR) && $(TAR) $(TAR_CREATE)
$(MEDIADIR)/$(PROJECT_TITLE).$(VERSION).biz.tar $(PROJECT_TITLE).$(VERSION)

install : media $(SINODBMSDIR)/extend
  cd $(SINODBMSDIR)/extend &&  $(TAR) xvf
$(MEDIADIR)/$(PROJECT_TITLE).$(VERSION).biz.tar

clean :
  $(RM) $(RMFLAGS) $(PROJECT_LIBS) $(PROJECTC_OBJS)


$(BINDIR) :
  -mkdir $(BINDIR)

$(STAGEDIR) :
  -mkdir -p $(STAGEDIR)

$(EXTENDDIR) :
  -mkdir -p $(EXTENDDIR)

$(COMPONETDIR) :
  -mkdir -p $(COMPONETDIR)

$(MEDIADIR) :
  -mkdir -p $(MEDIADIR)
```

2-prepare.sql

```
execute procedure ifx_allow_newline('t');
insert into sysbldmodules (bld_id, bld_server_type, bld_inst_state) values (
"%SYSBLDNAME%", "any", 0);
insert into sysbldscripts (bld_id, blocale_id, bscr_type, bscr_sequence,
bscr_sql_script) values ( "%SYSBLDNAME%", "any", 4, 1, "objects.sql");
insert into sysbldupgrades (bld_id, bld_upgrade_like) values ( "%SYSBLDNAME%",
"sample001.%");
```

3-objects.sql

```
execute procedure ifx_allow_newline('t');
insert into sysbldobjects (bld_id, obj_kind, obj_signature, obj_owner, sequence,
create_sql, create_can_fail, drop_sql, drop_can_fail) values
(
  "%SYSBLDNAME%", 0, "helloworld ()", "%SYSBLDUSER%", 0,
  "
  create function helloworld () returns lvarchar
  with (not variant, parallelizable)
  external name ""$SINODBMSDIR/extend/%SYSBLDDIR%/sample001.bld(helloworld)"" language
c;
  grant execute on function helloworld () to public;
  ",
  "f",
  "drop function helloworld (); ",
  "f"
);
insert into sysbldobjects (bld_id, obj_kind, obj_signature, obj_owner, sequence,
create_sql, create_can_fail, drop_sql, drop_can_fail) values
(
  "%SYSBLDNAME%", 0, "sysbldregistered", "%SYSBLDUSER%", 0,
  "insert into sysbldregistered (bld_id) values ( ""%SYSBLDNAME%""); ",
  "f",
  "delete from sysbldregistered where bld_id = ""%SYSBLDNAME%""; ",
  "f"
);
```

4-sample001.h

```
#ifndef HDR_sample001_H
#define HDR_sample001_H
```

```
/*
**  Configure tracing by setting TRACE_DEBUG_sample001
**  to 0 to completely disable tracing or 1 to  enable
**  tracing.  This define can be set from the compiler
**  command line by using the -DTRACE_DEBUG_sample001=0 flag.
*/
#ifndef TRACE_DEBUG_sample001
#define TRACE_DEBUG_sample001 1
#endif

#ifndef DBDK_LOHSIZE
#define DBDK_LOHSIZE    sizeof(MI_LO_HANDLE)
#define DBDK_LOBINFNSIZE  DBDK_LOHSIZE
#endif

/* Define MI_LO_HANDLES for backwards compatability. */
#define MI_LO_HANDLES MI_LO_LIST

/*
**  Large object file name mask.  '?' is a wild-card that
**  is filled in when a large object is written to disk.
*/
#define LO_FN_MASK  "????????.lo"

/*  Error messages
**
*/
#define ERRORMESG1  "UGEN1"
#define ERRORMESG2  "UGEN2"
#define ERRORMESG3  "UGEN3"
#define ERRORMESG4  "UGEN4"
#define ERRORMESG5  "UGEN5"
#define ERRORMESG6  "UGEN6"
#define ERRORMESG7  "UGEN7"
#define ERRORMESG8  "UGEN8"
#define ERRORMESG9  "UGEN9"
#define ERRORMESG10 "UGENA"
#define ERRORMESG11 "UGENB"
#define ERRORMESG12 "UGENC"
#define ERRORMESG13 "UGEND"
#define ERRORMESG14 "UGENE"
#define ERRORMESG15 "UGENF"
#define ERRORMESG16 "UGENG"
#define ERRORMESG17 "UGENH"
#define ERRORMESG18 "UGENI"
#define ERRORMESG19 "UGENJ"

/* Use DBDK_TRACE to direct trace messages to the trace file. */
#if TRACE_DEBUG_sample001
```

```
#define DBDK_TRACE    (1 << 16)
#else
#define DBDK_TRACE    0
#endif

/*
** Print a message to the trace file and for  the user.
** N.B.: This macro uses Gen_Con.  Your  function  must
**       declare Gen_Con as MI_CONNECTION * and  either
**       open the connection or set it to NULL.
*/
#define DBDK_TRACE_ERROR( Caller, ErrNo, ErrLevel )            \
                          Gen_Trace                            \
                          (                                    \
                              Gen_Con,                         \
                              Caller,                          \
                              __FILE__,                        \
                              __LINE__,                        \
                              ErrNo,                           \
                              "sample001",                     \
                              ErrLevel,                        \
                              MI_SQL | DBDK_TRACE              \
                          );

/* Print a message to the trace file. */
#if TRACE_DEBUG_sample001

/*
** Print a message to the trace file.
** N.B.: This macro uses Gen_Con.  Your  function  must
**       declare Gen_Con as MI_CONNECTION * and  either
**       open the connection or set it to NULL.
*/
#define DBDK_TRACE_MSG( Caller, ErrNo, ErrLevel )              \
                          Gen_Trace                            \
                          (                                    \
                              Gen_Con,                         \
                              Caller,                          \
                              __FILE__,                        \
                              __LINE__,                        \
                              ErrNo,                           \
                              "sample001",                     \
                              ErrLevel,                        \
                              DBDK_TRACE                       \
                          );

#else

#define DBDK_TRACE_MSG( Caller, ErrNo, ErrLevel )
```

```
#endif

/* These macros are used on entry to, and on exit from, a function. */
#define DBDK_TRACE_ENTER( Caller )  DBDK_TRACE_MSG( Caller, ERRORMESG13, 20 )
#define DBDK_TRACE_EXIT( Caller )   DBDK_TRACE_MSG( Caller, ERRORMESG14, 20 )

/*
**  Interval types.
*/
#define YEAR_TO_MONTH   1
#define DAY_TO_SECOND   2

/*
**  UDREXPORT is normally used to export a function from the BizWrapper when
**  linking on Windows.  UNIX source files should maintain this define in source
**  for use when porting back to Windows.
*/
#ifndef UDREXPORT
#define UDREXPORT
#endif

/* Function prototypes. */
void Gen_Trace
(
  MI_CONNECTION *      Gen_Con,
  char *               Gen_Caller,
  char *               Gen_FileName,
  mi_integer           Gen_LineNo,
  char *               Gen_MsgNo,
  char *               Gen_Class,
  mi_integer           Gen_Threshold,
  mi_integer           Gen_MsgType
);

#endif
```

5-support.c

```
/*
**  The following is placed here to insure
**  that name "mangling" does not occur.
*/
#ifdef __cplusplus

extern "C"
{
```

```
#endif

/* Standard library includes. */
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>

/* Used by GLS routines. */
#include <ifxgls.h>

/* Include when accessing the API. */
#include <mi.h>

/* This is the project include file. */
#include "sample001.h"

/*****************************************************************************
**
** Function name:
**
**  Gen_Trace
**
** Description:
**
**  This function writes trace information to the trace file.
**
**  To enable tracing, you must first create a trace class
**  by inserting a record into the systraceclasses system
**  catalog:
**
**    insert into sinodbms.systraceclasses(name) values('sample001');
**
**  The name of the trace file must be set. If the file name
**  is not set, the server uses a default file name:
**  the session id followed by ".trc" in the /tmp directory.
**  Use "onstat -g ses" to get the session id.
**
**  The following code snippet may be used to set the name
**  of the output trace file from within your code.
**
**    mi_tracefile_set( "/yourpath/yourfile.trc" );
**
**  Alternately, the TraceSet_sample001 procedure may be
**  used from SQL to set the trace file name and trace threshold
**  level.  See this procedure for more details.
**
**  To insure that tracing text  actually  appears in the
```

```
**  output trace file, SERVER_LOCALE, CLIENT_LOCALE,  and
**  DB_LOCALE must be set in the environment to the
**  appropriate locale (e.g., "en_us.1252").
**
** Parameters:
**
**  MI_CONNECTION *    Gen_Con        The database connection.
**  char *             Gen_Caller     Call originated from this routine.
**  char *             Gen_FileName   Call originated in this file.
**  mi_integer         Gen_LineNo     Call originated on this line.
**  char *             Gen_MsgNo      ERRORMESG number.
**  char *             Gen_Class      Tracing class.
**  mi_integer         Gen_Threshold  Tracing threshold.
**  mi_integer         Gen_MsgType    MI_SQL | MI_MESSAGE | DBDK_TRACE.
**
** Return value:
**
**  None.
**
** History:
**
**
** Identification:
**
**
*****************************************************************************
*/
void
Gen_Trace
(
MI_CONNECTION *            Gen_Con,                    /* The database connection.
          */
char *                     Gen_Caller,                 /* Call originated from this
routine.     */
char *                     Gen_FileName,               /* Call originated in this
file.          */
mi_integer                 Gen_LineNo,                 /* Call originated on this
line.          */
char *                     Gen_MsgNo,                  /* ERRORMESG number.
           */
char *                     Gen_Class,                  /* Tracing class.
          */
mi_integer                 Gen_Threshold,              /* Tracing threshold.
          */
mi_integer                 Gen_MsgType                 /* MI_SQL | MI_MESSAGE |
DBDK_TRACE.      */
)
{
  MI_CONNECTION *    Gen_TraceCon;  /* The database connection.           */
```

```
/* Route the message to the trace file? */
if( Gen_MsgType & DBDK_TRACE )
{
  /* Write the message to the trace file. */
  GL_DPRINTF( Gen_Class,
        Gen_Threshold,
        (
            Gen_MsgNo,
            "FUNCTION%s", Gen_Caller,      /* Substitute the caller here.    */
            "FILENAME%s", Gen_FileName,    /* Substitute the file name here. */
            "LINENO%d",   Gen_LineNo,      /* Substitute the line info here. */
            MI_LIST_END                    /* Terminate the list!!           */
        )
      );
}

/* Mask off the mi_db_error_raise flags. */
Gen_MsgType &= 0xffff;

/* Route the message back to the user? */
if( Gen_MsgType )
{
  /*
  ** Obtain a connection for mi_db_error_raise.
  ** If available, use the connection that  has
  ** already been established.
  */
  if( Gen_Con == NULL )
  {
    /* No connection is available so open a new one. */
    Gen_TraceCon = mi_open( NULL, NULL, NULL );
  }
  else
  {
    Gen_TraceCon = Gen_Con;
  }

  /* If requested, also write the message to the user. */
  mi_db_error_raise( Gen_TraceCon,              /* This is the connection handle. */
              Gen_MsgType,                /* Route to the user.             */
              Gen_MsgNo,                  /* Print this message.            */
              "FUNCTION%s", Gen_Caller,   /* Substitute the caller here.    */
              (char *)NULL );             /* Terminator.                    */

  /*
  ** mi_db_error_raise may not return
  ** and this line may not be reached.
  */
```

```
  }
}


#ifdef __cplusplus

}

#endif
```

6-udr.c

```
/*
**  The following is placed here to insure
**  that name "mangling" does not occur.
*/
#ifdef __cplusplus

extern "C"
{

#endif

/* Standard library includes. */
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stddef.h>

#include <ifxgls.h>

/* Include when accessing the API. */
#include <mi.h>

/* This is the project include file. */
#include "sample001.h"



/*****************************************************************************
**
** Function name:
**
**  helloworld
**
** Description:
```

```
**
** Special Comments:
**
**  Entrypoint for the SQL routine helloworld () returns lvarchar.
**
**  A stack size of 32,767 bytes  has  been  requested  for
**  the routine.  Normally, this is sufficient memory for most
**  invocations of your UDR.  If you intend, however, to  call
**  this routine recursively or other routines that use  large
**  or unknown stack sizes, you should use mi_call().  mi_call
**  checks to insure that sufficient stack space is available.
**
** Parameters:
**
**
** Return value:
**
**  mi_lvarchar
**
** History:
**
**
** Identification:
**
**
*****************************************************************************
*/


UDREXPORT mi_lvarchar *helloworld(MI_FPARAM *Gen_fparam)

{
  mi_lvarchar *        Gen_RetVal;      /* The return value.                    */
  MI_CONNECTION *      Gen_Con;         /* The connection handle.               */
  mi_integer           Gen_ReturnSize;  /* Return value is this size.           */

  gl_mchar_t *       Gen_RetData;   /* Store the return data here.        */
  DBDK_TRACE_ENTER("helloworld")
  /* Use the NULL connection. */
  Gen_Con = NULL;
  char HelloString[] = "Hello World!";

  /*
  ** TO DO: Remove this comment and call to
  **        mi_db_error_raise after implementing
  **        this function.
  ** mi_db_error_raise( Gen_Con, MI_EXCEPTION,
  **         "Function helloworld has not been implemented." );
  */
```

```
  /*
  ** TO DO: Specify  the appropriate amount of memory that
  ** you require for the return value.  Because the return
  ** value is being allocated using mi_new_var(), there is
  ** no need to allocate a trailing NULL character.
  */
  Gen_ReturnSize = sizeof(HelloString) - 1;

  /*
  ** Allocate memory for the return value. This
  ** memory will be freed by the server.
  */
  Gen_RetVal = (mi_lvarchar *)mi_new_var( Gen_ReturnSize );
  Gen_RetData = (gl_mchar_t *)mi_get_vardata( (mi_lvarchar *)Gen_RetVal );

  /*
  ** Store the UDR's return value in the allocated
  ** memory for Gen_RetVal.
  */
  bycopy(HelloString, Gen_RetData, Gen_ReturnSize);

  /* Return the function's return value. */
  DBDK_TRACE_EXIT("helloworld")
  return Gen_RetVal;
}


#ifdef __cplusplus

}

#endif
```