FISCO BCOS CRUD使用指南

原创 廖飞强 FISCO BCOS开源社区 2019-11-29

廖飞强

FISCO BCOS核心开发者

极速体验是一种追求

AUTHOR | 作者

本文将介绍 FISCO BCOS的CRUD功能，帮助开发者更高效便捷地开发区块链应用。

FISCO BCOS

为什么设计CRUD功能?

在FISCO BCOS 1.0中，节点采用MPT数据结构，通过LevelDB将数据存储于本地，这种模式受限于本地磁盘大小，当业务量增大时数据会急剧膨胀，要进行数据迁移也非常复杂，给数据存储带来较大成本和维护难度。

为了突破容量和性能瓶颈，FISCO BCOS 2.0针对底层存储进行了重新设计，实现了分布式存储，带来了容量和性能上的提升。得益于分布式存储采用了库表结构，FISCO BCOS 2.0设计一套CRUD(Create增加、Read读取、Update更新和Delete删除)，让接口更加顺其自然。CRUD面向库表的开发方式符合业务开发习惯，同时也为业务开发提供了另外一种选择(以往只能用Solidity合约)，从而让区块链应用开发更加便利。

CRUD有哪些优势?

CRUD的核心设计思想是提供面向SQL编程的区块链应用开发规范。其好处显而易见，主要体现在两升两降。

提升开发区块链应用的效率

CRUD类似传统业务SQL编程开发模式，大大降低了合约开发难度。开发者将合约当做数据库的存储过程，将区块链数据的读写操作转换为面向表的读写操作，简单易用，极大提升了开发区块链应用的效率。

提升区块链应用的性能

CRUD底层逻辑基于预编译合约实现，其数据存储采用分布式存储，读写数据产生的交易不再进入缓慢的EVM虚拟机执行，而是由高速的预编译合约引擎执行，因此提高了合约的数据读写效率，使得基于CRUD开发的区块链应用具有更高的性能。

降低合约维护和升级的复杂度

CRUD合约的逻辑与存储分离，这样开发者就只需要关心核心业务逻辑，数据不再与特定合约绑定，更方便合约升级。当合约逻辑需要变更时，可以部署新合约，新合约可以读写旧合约创建的表和数据。

降低面向SQL业务的迁移成本

传统商业应用很大一部分通过数据库表的方式管理数据，通过CRUD合约可以将面向SQL设计的商业应用非常顺滑的迁移到区块链上，降低业务的迁移成本。

CRUD如何使用?

介绍了CRUD的两升两降优势，想必大家非常关注CRUD具体如何使用？其实，CRUD简单易用，目前可以用两种方式使用CRUD功能，分别是CRUD合约和SDK CRUD Service接口。

CRUD合约

开发者与区块链交互的媒介主要是智能合约，CRUD功能自然会集成在智能合约。集成方式非常轻，Solidity合约只需要引入FISCO BCOS官方提供的Table.sol抽象接口合约文件即可。Table.sol包含分布式存储专用的智能合约接口，其接口实现在区块链节点，可以创建表，并对表进行增删改查操作。引入了该抽象接口的合约称为CRUD合约，以区别于没有引用该接口的Solidity合约。

Table.sol抽象接口合约文件包括以下抽象合约接口，下面分别进行介绍。

TableFactory合约

用于创建和打开表，其固定合约地址为0x1001，接口如下：

接口	功能	参数
createTable(string ,string, string)	创建表	表名，主键名，表的其他字段名（字段之间以英文逗号分隔）
opentTable(string)	打开表	表名

Entry合约

Entry代表记录对象，一个Entry对象代表一行记录，其接口如下：

接口	功能	参数
set(string, int)	设置字段	字段名，字段值
set(string, string)	设置字段	字段名，字段值
set(string, address)	设置字段	字段名，字段值
getInt(string)	获取字段值	字段名
getString(string)	获取字段值	字段名
getBytes64(string)	获取字段值	字段名
getBytes32(string)	获取字段值	字段名
getAddress(string)	获取字段值	字段名

Entries合约

Entries是记录集合对象，Entries用于存放Entry对象，其接口如下：

接口	功能	参数
get(int)	获取指定索引的Entry	Entries的索引
size()	获取Entries的大小	无

Condition合约

查询、更新和删除记录时指定的过滤条件对象，其接口如下：

接口	功能	参数
EQ(string, int)	相等条件	字段名，字段值
EQ(string, string)	相等条件	字段名，字段值
NE(string, int)	不等条件	字段名，字段值
NE(string, string)	不等条件	字段名，字段值
GT(string, int)	大于条件	字段名，字段值
GE(string, int)	大于或等于条件	字段名，字段值
LT(string, int)	小于条件	字段名，字段值
LE(string, int)	小于或等于条件	字段名，字段值
limit(int)	记录选取条件	返回多少条记录
limit(int, int)	记录选取条件	记录启始行位置， 返回多少条记录

Table合约

用于对表进行增删改查操作的对象，其接口如下：

接口	功能	参数
select(string, Condition)	查询数据	主键值，过滤条件对象
insert(string, Entry)	插入数据	主键值，记录对象
update(string, Entry, Condition)	更新数据	主键值，记录对象，过滤条件对象
remove(string, Condition)	删除数据	主键值，过滤条件对象

newEntry()	创建Entry对象	无
newCondition()	创建Condition对象	无

有了以上对Table.sol抽象接口合约的了解，现在可以正式进行CRUD合约的开发。以官方提供的CRUD合约TableTest.sol为示例，介绍其开发的三步曲：

第一步：引入Table.sol

将Table.sol合约放入TableTest.sol同级目录，并在TableTest.sol合约文件中引入Table.sol，其代码如下：

```
import "./Table.sol";
```

第二步：创建表

在TableTest.sol合约文件中，创建表的核心代码如下：

```
// 创建TableFactory对象，其在区块链上的固定地址是0x1001
TableFactory tf =TableFactory(0x1001);
// 创建t_test表，表的主键名为name，其他字段名为item_id和item_name
int count =tf.createTable("t_test", "name","item_id,item_name");
```

注：

- createTable执行原理：createTable执行成功之后，将会在区块链系统表_sys_tables_(区块链启动会自动创建该表，专门记录区块链中所有表的信息)中插入t_test的表信息，即表名，主键名和其他字段名，但并没有正式创建该表。当对t_test表进行增删改查操作时，会首先判断t_test表是否存在，若不存在，则会查询_sys_tables_表获取t_test表的信息，如果查询有t_test表信息，则创建该表，否则执行失败。t_test表存在，则继续执行增删改查操作。

- 这一步是可选操作：比如新合约只是读写旧合约创建的表，则不需创建表这步操作。

第三步：针对表进行CRUD操作

在TableTest.sol合约文件中，插入记录核心代码如下：

```
TableFactory tf =TableFactory(0x1001);
// 打开创建的t_test表
Table table =tf.openTable("t_test");
// 创建Entry对象，即创建一条空记录
Entry entry = table.newEntry();
// 设置Entry对象，即设置记录的字段值
entry.set("name", name);
entry.set("item_id",item_id);
entry.set("item_name",item_name);
// 调用Table的insert方法插入记录
// 返回插入影响的记录数，值为1则插入成功，否则插入失败
int count = table.insert(name,entry);
```

查询记录核心代码如下：

```
TableFactory tf =TableFactory(0x1001);
Table table =tf.openTable("t_test");
// 创建Condition对象，条件为空表示不过滤，也可以根据需要使用条件过滤
Condition condition =table.newCondition();
// 调用Table的select方法查询记录，获得记录集合
Entries entries =table.select(name, condition);
// 获取记录集合的大小
int size = entries.size();
bytes32[] memoryuser_name_bytes_list = new bytes32[](uint256(size));
int[] memory item_id_list = newint[](uint256(size));
bytes32[] memoryitem_name_bytes_list = new bytes32[](uint256(size));
// 遍历记录集合
// 将记录的三个字段值分别存放到三个数组中，方便方法返回查询的数据
for(int i = 0; i< size; ++i) {
// 根据索引获取记录集合中的记录
Entry entry = entries.get(i);
// 根据记录的字段名查询字段值
// 注意字段值的类型不同需要选择相应的get方法
user_name_bytes_list[uint256(i)] =entry.getBytes32("name");
item_id_list[uint256(i)] =entry.getInt("item_id");
item_name_bytes_list[uint256(i)] =entry.getBytes32("item_name");
}
```

更新记录核心代码如下：

```
TableFactory tf =TableFactory(0x1001);
Table table =tf.openTable("t_test");
Entry entry = table.newEntry();
entry.set("item_name",item_name);
Condition condition =table.newCondition();
// 设置过滤条件
condition.EQ("name",name);
condition.EQ("item_id",item_id);
// 调用Table的update方法更新记录
// 返回更新影响的记录数，值大于0则更新成功，否则更新失败
int count = table.update(name, entry,condition);
```

删除记录核心代码如下：

```
TableFactory tf =TableFactory(0x1001);
Table table =tf.openTable("t_test");
Condition condition =table.newCondition();
condition.EQ("name",name);
condition.EQ("item_id",item_id);
// 调用Table的remove方法删除记录
// 返回删除影响的记录数，值大于0则删除成功，否则删除失败
int count = table.remove(name,condition);
```

SDKCRUD Sevice接口

通过CRUD合约，我们可以看到只要合约方法涉及到数据读写操作，则首先打开对应的表，然后调用读写相关接口就可以进行区块链数据的读写。

同时，我们注意到，CRUD合约的开发方式还是离不开编写合约、编译合约、部署合约，最后才能调用合约实现相关功能。

那么有没有更方便，更简洁的方式呢？比如开发者合约都不用写，不用部署合约，就可以读写区块链上的数据。答案显而易见，我们已经实现了这样的极致易用需求。

FISCO BCOS SDK提供CRUD Service数据上链接口，这些接口实现的原理是调用区块链内置的一个预编译的CRUD合约，专门负责对用户表进行增删改查操作。

Java SDKCRUD Service 实现在（Python SDK 和 Nodejs SDK 类似）org.fisco.bcos.web3j.precompile.crud.CRUDService类，其接口如下：

接口	功能	参数
createTable(Table table)	创建表	表对象(需要设置其表名，主键字段名和其他字段名。其中，其他字段名是以英文逗号分隔拼接的字符串)
select(Table table, Condition condition)	查询数据	表对象，Condition对象
insert(Table table, Entry entry)	插入数据	表对象，Entry对象
update(Table table, Entry entry，Condition condition)	更新数据	表对象，Entry对象，Condition对象
remove(Table table, Condition condition)	删除数据	表对象，Condition对象
desc(String tableName)	查询表的信息	表名

以上接口覆盖了表的创建、查看和增删改查操作。用户只需要调用SDK的接口就能完成相关操作，具体示例以下地址查看。
https://github.com/FISCO-BCOS/web3sdk/blob/master/src/integration-test/java/org/fisco/bcos/precompile/CRUDServiceTest.java

其中调用写接口会产生与调用CRUD合约接口等效的交易，需要共识节点共识一致后才会落盘存储。值得关注的是，利用CRUD Service接口，FISCO BCOS控制台针对每个接口实现了易用的sql语句命令，欢迎访问以下地址体验。
https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/manual/console.html

两种CRUD使用方式的比较

可能用户有这样的疑问，CRUDService接口如此简洁易用，那么是不是我们就只依靠这组接口完

成区块链业务呢？其实也不尽然，相比于CRUD合约，其局限主要在两点：

- 数据访问范围受限：CRUDService接口做的事情非常专一，其本质是一组针对区块链用户表数据的读写接口。因此对于不是用户表数据，这组接口无能为力。例如Solidity合约的状态变量，并没有存储在一个用户创建的表中，通过这组接口就查询不到状态变量的数据。然而CRUD合约本质是Solidity合约，只是多引入了一个Table.sol合约文件，使其有了CRUD功能。通过CRUD合约就可以操作Solidity状态数据和用户表数据。

- 业务处理能力受限：很多区块链业务不仅仅只是利用区块链上链数据，而是需要设计相关合约逻辑，满足相关条件和验证的前提下才可以向区块链提交数据，仅利用CRUD Service接口就难以胜任这样的区块链业务，因此CRUD合约必不可少。

有哪些Effective CRUD最佳实践？

CRUD功能虽然简单易用，但也需要清晰的知道如何高效的使用CRUD。

条款1：绝大多数情况下请选择CRUD合约开发区块链应用

区块链应用开发中，如果确定仅仅只是上链数据且数据只采用用户表存储，不依赖合约执行相关业务逻辑，那么考虑使用CRUD Service接口。否则，推荐使用CRUD合约。另外，为了方便开发和调试，可以随时利用CRUD Service的读接口（select和desc接口）或者利用控制台对应的相关命令进行数据查询。

条款2：Table.sol文件内容不可修改

Table.sol文件定义的是CRUD功能相关的抽象接口，每个接口区块链均有对应的具体实现。如果用户修改该文件，会导致交易执行失败或功能异常等问题。

条款3：CRUD用户表的主键不唯一

用户习惯于关系型数据库中的主键字段是唯一的，因此默认CRUD用户表中的主键也是唯一的，其实并不是。插入多条记录时，可以指定相同的主键。主键不唯一是由于CRUD表中存储的是主键到对应Entries的映射，进而基于主键进行增删改查。因此，调用CRUD接口(插入、更新、查询和删除记录)时，均需要传入主键值(不是主键名)。另外，主键值过长，会影响读写效率，建议主键值不

要设置太长。

条款4：使用CRUD合约时，表字段过多，使用数组或struct封装字段参数

由于Solidity合约语言的限制，一个合约方法中的局部变量个数不得超过16个，否则会出现"Stacktoo deep, try removing local variables"编译错误。因此对于表字段比较多的情况，可以考虑将多个字段参数使用数组或struct封装，以减少局部变量个数，使其满足编译条件。

条款5：使用CRUD合约时，方法中如果涉及表操作，均需要先打开表

TableFactory抽象合约有openTable(string)方法，该方法返回打开的表对象。我们可能会将该表对象设置为合约的全局变量，方便下次操作表时直接使用。注意这是错误操作，因为每次打开表时，其获取的表对象注册在一个临时地址上，在下一个区块之后，该地址不再有效，因此不能将表对象设置为全局变量使用，而是当方法需要操作表之前，均需要打开表。另外，由于TableFactory在区块链中注册的是固定地址0x1001，因此可以将TableFactory对象设置为全局变量，不必每次均创建该对象。

条款6：使用CRUD合约，可以采用表和状态变量混合存储数据

CRUD合约是具备CRUD功能的Solidity合约，因此Solidity之前的状态变量存储方式依然可以使用。当存储一个变量值时，可能直接使用一个变量存储比较方便，因此可以既定义几个状态变量，同时也创建几个用户表进行数据存储。对于比较结构化的数据，推荐使用表的存储方式。

条款7：采用权限控制管理用户表

CRUD合约的逻辑与用户表数据分离，因此通过合约接口的限制已控制不住用户表的写操作（包括插入、更新和删除操作）。为了避免任何账户均可以写用户表，推荐使用权限控制来管理用户表（点击参考权限控制具体使用：
https://fisco-bcos-documentation.readthedocs.io/zh_CN/latest/docs/design/security_control/permission_control.html）。
指定特定账户才有权限写用户表。

FISCO BCOS的代码完全开源且免费

下载地址↓↓↓

https://github.com/FISCO-BCOS/FISCO-BCOS