模块图:

changeReader

changeFilter

dmlChangeListParser

dmlChangeListFilter

dmlChangeList

dmlRow

transaction

transactionFilter

transactionConsume

二、模块解释

1、changeReader

功能: 从日志文件中读取出一个一个的change。

对应的代码文件:

lgc\_readRedoFile.cpp

lgc\_RedoFile.cpp

lgc\_RedoRecord.cpp

lgc\_RedoFileInput.cpp

lgc\_RedoRecordInput.cpp

lgc\_ChangeInput.cpp

lgc\_RecordHandler.cpp

详细解释:

>>> 根据threadId,startSCN创建LGC\_RedoFileInput对象

>>> 迭代地从LGC\_RedoFileInput对象中读出LGC\_RedoFile

>>> 根据LGC\_RedoFile对象创建LGC\_RedoRecordInput对象

>>> 迭代地从LGC\_RedoRecordInput中读出LGC\_RedoRecord

>>> 根据LGC\_RedoRecord创建LGC\_ChangeInput对象

>>> 迭代地从LGC\_ChangeInput中读出Change

(>>>表时上一步做完了之后，接着做下一步)

2、changeFilter

功能: 过滤掉没必要解析的change

对应的代码文件:

Lgc\_Change.cpp

详细解释:

和dml操作不相关的Change没必要解析，因为我们目前只解析dml操作。

抽象类LGC\_Change的isNeedAddToTrsct虚函数直接决定了该Change是否是需要被过滤的，

如果该方法返回false则该change会被过滤掉。

3、dmlChangeList:

功能: 存放对个change。

对应的代码文件:

Lgc\_Change.cpp

Lgc\_dmlChangeList.cpp

详细解释:

一个dml操作有一个或者多个change组成，同一个dml操作相关的change放在同一个dmlChangeList中。

4．dmlChangeListFilter

功能:过滤掉没必要解析的dmlChangeList

对应的代码文件:

Lgc\_DmlChangeList.cpp

Lgc\_metaTable.cpp

详细解释:

一个dml操作肯定是针对某一张表，但是我们不是对每一张表都感兴趣，针对不感兴趣的表的dml操作应该被过滤掉。感兴趣的表的结构信息都被缓存到一个全局的队列中。

>>> 从dmlChangeList对象中获取属性object\_id(每张表都有唯一的object\_id)

>>> 根据object\_id从感兴趣表队列中查找表的结构信息

>>> 如果没有找到则过滤掉该dmlChangeList

5．dmlChangeListParser

功能: 将dmlChangeList解析成一个或者多个dmlRow

对应的代码文件:

Lgc\_DmlChangeList.cpp

Lgc\_Change.cpp

Lgc\_OpcodeParser.cpp

Lgc\_DmlRow.cpp

Lgc\_DmlRowsOutput.cpp

详细解释:

一个像这样insert into <table\_name> (<col\_name\_list>) values(<col\_val\_list>)的插入语句，只会插入向数据库插入一行数据。也就只会解析成一个dmlRow。

一个像这样insert into <table\_name> as select … 这样的直接插入语句，oracle在写日志时会被拆分成多个单行插入和多个多行插入。一个多行插入对应一个dmlChangelist,该dmlChangeList会被解析成多个dmlRow。一个单行插入对已一个dmlChangeList,该该dmlChangeList会被解析成一个dmlRow。

一个像这样update <table\_name> set … where …这样的update语句，可能会更新多行。在写日志时会被拆分成多个单行update操作。每个单行update操作对应一个dmlChangeList, 该dmlChangeList被解析成单个dmlRow。

一个像这样delete from <table\_name> where … 这样的delete语句，可能会删除多行。在写日志时会被拆分成多个只删除一行的delete操作。一个只删除一行的delete对应一个dmlChangeList, 该dmlChagneList被解析成一个dmlRow。

一个dml操作更新了一列或者多列，被跟新的列的值被分为前镜像和后镜像。前镜像是该列被跟新以前的旧值，后镜像是被跟新后的新值。

任何一个dml操作都可以被解析成以下格式:

<object\_id> <afterimage\_colValue\_list> <beforeImage\_colValue\_list>

其中

object\_id对应表的object\_id。

afterimage\_colValue\_list表示多个列的后镜像的值。

BeforeImage\_colValue\_list表示多个列的前镜像的值。

其中

insert对应的beforeImage\_colValue\_list为空。因为是插入新行，该行的列对应的前镜像不存在。

Delete对应的afterImage\_colValue\_list为空，因为删除行后行就不存在了。所以该行的列对应的后镜像不存在。

Update对应的afterImage\_colValue\_list和beforeImage\_colValue\_list一定都不为空。其中afterImage\_colValue\_list对应update语句中set 和 where之间的列。BeforeImage\_colValue\_list对应update语句where之后的列。

dmlChangeList被分成了两部分: redoChangeList 和 undoChangeList。

其中

redoChangeList被解析成afterImage\_colValue\_list

undoChangeList被解析成beforeImage\_colValue\_list

::解析dmlChangeList

>>> 获取dmlChangeList的object\_id属性

>>> 解析redoChangeList,得出afterImage\_colValue\_list

>>> 解析undoChangeList得出beforeImage\_colValue\_list

>>> 根据object\_id, afterimage\_colValue\_list, beforeImage\_colValue\_list组织成一个dmlRow

6、dmlRow

功能：存放一行数据

对应的文件:

Lgc\_DmlRow.cpp

详细描述:

一个dmlChangeList被解析成一个或者多个dmlRow, 一个dmlRow可以被解析成只针对一行的dml操作。

一个dmlRow的主要组成: <object\_id> <afterimage\_colValue\_list> <beforeImage\_colValue\_list>

7、Transaction

功能: 存放一个事务下的所有dmlRow

对应的代码文件:

Lgc\_Transaction.cpp

详细描述:

Oracle为了保证数据的一致性以事务为单位操作数据库。一个事务里可能包含多个dml操作， 这些dml操作被解析成多个dmlRow, 然后存放在transaction对象中。

8、transactionFilter

功能: 过滤掉没用的事务

对应的文件:

Lgc\_Transaction.cpp

详细描述:

有些事务被回滚了或者事务中没有解析出的dmlRow, 我们把它过滤掉。

9、transactionConsumer

功能: 消费掉数据已经被解析了的事务

代码文件:

Lgc\_Transaction.cpp

Lgc\_TConsumer.cpp

lgc\_DmlRow.cpp

lgc\_DmlRowParser.cpp

lgc\_MediaFileOutput.cpp

详细描述:

::消费transaction

>>> 解析事务开始数据

>>> 解析事务的dmlRow列表

>>> 解析事务结束的数据

(解析出的数据都会用lgc\_MediaFileOutput对象输出到中间文件中)