DB2 大型数据库容灾备份实时复制系统①

屈志毅，王 涛，李建旭

(兰州大学 信息科学与工程学院，兰州 730000)

摘 要：设计并实现了基于 DB2 大型数据库的实时复制软件。采用模块化设计，本软件主要包括 5 个模块：客户端，服务器端，全同步模块，增量同步模块，加载模块。这 5 个模块相互协调，由客户端发出请求，服务端统一调度。在实现该软件的过程中，使用了实时分析数据库日志的方法来抓取源端数据库的变化。本软件没有调用任何 DB2 数据库的备份接口，完全从底层自主研发。所以在功能和效率上比一般产品要优良。在操作系统为 LINUX，CPU 为酷睿 2.4GHz，内存为 2GB 的测试环境下，本软件全同步，加载，增量同步运行效果良好。

关键词：DB2 数据库；容灾；备份；实时复制；日志分析；

Real-time Duplicate Software in DB2 Large Database

QU Zhi-Yi, WANG Tao, LI Jian-Xu

(School of Information Science and Engineering, Lanzhou University, Lanzhou 230039, China)

Abstract: This paper presents the process for research and implementation a real-time duplicate software in DB2 database. This software used the modular design, and has 5 major modules: Client, Server, Full synchronizing, Increase synchronizing, Load. While the software running, the 5 modules coherent on with another. The client call the server, and the server schedules all task. In the process of the implementation, we grap the database's log real-time to gain the change. This software doesn't use any DB2 database's backup interface, so it well done in the function and efficiency. The test environment is OS:LINUX, CPU:Core 2.4 GHz ,Memory: 2 GB. Under such a test environ-ment, this software worked well.

Key words: DB2database; tolerant; backup; real-time duplicate; log analysis;

随着企业的业务不断增加，数据量也呈几何级数上升。传统的人工数据库备份与恢复已经满足不了实际需要，因此数据库相关备份复制软件就有了广阔的市场需求。各大银行，证券机构，电信等企业对 DB2 数据库备份软件的需要比较迫切，不少公司也针对这个需求研发出了不少相关产品。但是，能够针对 DB2 数据库进行实时逻辑复制的成型产品只有 IBM 自己研发的 Q 复制。本文提出的系统在实现了 Q 复制基本功能的基础上，还具有使用简便，人机交互界面友好，以及价格等优势[2]。

1 DB2数据库实时复制系统概述

本系统是基于事务交易的逻辑级 DB2 数据同步软件。将源端数据库的交易信息以事务为单位，实时传递到目标端并且加载进去，从而达到源端与目标端同步的目的。

1.1 数据容灾

当今银行，电信，证券等行业数据十分重要。相关数据库如果发生问题，后果将十分严重。所以对相关数据库的容灾处理就变得尤其重要。数据容灾是指：在异地建立一个或多个数据库系统，该数据库是本地数据库的一个可用副本。与本地数据库实时保持一致，或者稍微滞后。当本地数据库一旦出现灾难性问题后，可以马上切换到异地数据库，从而尽可能减少数据丢失，并且不影响业务的继续运营。

① 基金项目:安徽省教育厅自然科学基金(2005KJ004ZD)
收稿时间:2011-07-08;收到修改稿时间:2011-08-17

© 中国科学院软件研究所 http://www.c-s-a.org.cn

1.2 实时复制

本文所说的实时复制是指目标端数据库中的数据信息与源端数据库中数据实时保持一致。通过这种方法可以使数据延迟非常低。最大程度保证了目标端数据库与源端数据库的一致。

2 系统工作流程以及系统技术原理

2.1 系统工作流程

系统工作流程见图 1 所示。

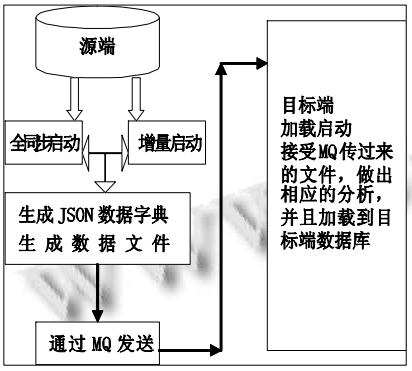

图 1 系统工作流程

2.1 系统技术原理

2.2.1 全同步原理

初始时，需要把源端数据库中已有的数据信息导出，然后发送到目标端进行加载。这个步骤称为全同步。全同步的实现原理分两步。首先，通过查找 DB2 数据库系统表如：SYSIBM.SYSTABLES，SYSIBM.SYSCOLUMNS，SYSIBM.INDEXES，SYSIBM.CHECKS 等[1]。来找出用户表的表结构信息。然后把这些表结构信息写入 JSON 文件，形成数据字典。然后，要把用户表中具体的数据取出来，生成相应的数据文件。根据数据字典在目标端可以组装出相应的 DDL 语句，然后执行。这样就能保证源端与目标端表结构一致。根据数据文件，可以在目标端组装出对应的 DML 语句，把具体的用户数据装载到对应的用户表中。到此为止，源端数据库与目标端数据库内容将完全一致，全同步结束。

2.2.2 增量同步原理

在进行全同步或全同步完成之后源端数据库发生新的变化，如果能够获取源端数据库这些操作，然后把这些操作生成相应的 DDL 数据字典和 DML 数据文件发送到目标端，由目标端加载程序进行加载，那么就能够保证源端数据库与目标端数据库的准实时同步。它的延迟仅仅是网络传输与执行 SQL 语句的延迟。

实现上述功能的原理为：DB2 数据库日志会记录对数据库进行的所有操作，包括对表进行的 DML，DDL 操作以及对数据库结构的更改。通过对 DB2 日志文件的分析，我们就能够获取数据库发生的变化。同时可以得到相应的数据[5]。根据日志分析的结果，生成相应的 JSON 数据字典以及数据文件。首先，找到 DB2 正在写的日志当前块。然后每隔一秒，来检查一下该日志文件，看其内容有无增加。若有，则分析其增加内容。

日志文件的结构[6]如图 2 所示：

```
1 0000000: 0400 0a00 0180 0000 4942 4d4c 4f47 0000
2 0000010: 0000 0000 0100 0000 e803 0000 0100 0000
3 0000020: b93b e74d 82a5 e54d ba3b e74d 0000 0000
  ......
513 0002000: b902 0000 1000 0000 198b 2803 0000 0000
514 0002010: 4e00 0000 4e00 0000 0000 0000 0000 0000
515 0002020: 0000 0000 14c7 0000 01a2 0200 0400 0000
516 0002030: 2200 6c0a 0900 0000 0000 800a 0000 2200
517 0002040: 0100 1a00 0b00 0000 0061 6263 2020 2020
518 0002050: 2020 2020 2020 2020 2020 2020 2000 2000
  ......
```

图 2 DB2 数据库日志文件结构

DB2 数据库日志文件由日志头和若干块(block)组成。日志开始的 8K 字节对我们来说是无用信息。从 8192 字节处开始，每 4K 字节为一个块。每个块由 16 字节的块头，若干个 log_row，以及最后 4 字节块尾组成。而 DB2 数据库的个各种操作就会被记录到 log_row 中。通过详细分析分析每一条 log_row 就可以得到操作的类型以及具体数据。

2.2.3 加载原理

目标端加载程序接收源端发送来的文件，判断文件类型。如果是数据字典，则读取该 JSON 文件，组装出相应的 DDL SQL 语句，并且在目标端数据库执行该 DDL 语句。如果是数据文件则根据预先定义好的数据文件结构，首先分析文件头部得出 DML 的类型。包括插入，删除，更新(INSERT, DELETE, UPDATE)。然后分析文件体，得到具体的数据。组装成对应的

© 中国科学院软件研究所 http://www.c-s-a.org.cn

DML 语句执行该 SQL 语句。

3 系统的整体设计与主要模块实现

本系统由 5 个模块组成：客户端，服务端，全同步模块，增量同步模块，加载模块。这 5 个模块相互协调进行工作。而真正实现数据库实时复制的模块为全同步，加载，增量。下面就这三个核心模块的实现加以论述。

3.1 全同步的实现

全同步模块的功能是生成源端数据库中已有的表结构数据字典以及具体的数据文件。从 DB2 数据库系统表中可以获取表结构信息，从用户表中获取具体数据[7]。为了更好地组织数据流，同时进行模块化，定义了下面几个非常重要的数据结构。

```
typedef struct{
char *schema;//表的 schema 名称
char *name;//表名
int colcount;//列数
int tbid;// schema id
int tbsid; //表 id 号
int compress;//是否压缩
char **colnames;//列名
int keycol;//是主键的列总数
int*keycols;//是主键的所有列
DB2Col*cols;//列的详细信息
GSList*lobs;//lob 大字段列表
char*space;//表空间
}DB2Table;
typedef struct{
char*name;//列名
int type;//列类型
char*type_name;//列类型名称
int nullable;//是否为空
int seq;//列的序号
int scale;//列的增量
int bufflen;//导出数据时 buff 的长度
//default
char *default_text;
}DB2Col;
```

为了提高全同步的速度，我们把所需系统表中内容一次性导出到 HASHTABLE（哈希表）中，减少对数据库的访问次数，从而缩短时间。

全同步的实现分如下步骤：

(1) 得到表信息：

通过编 db2_get_tables_hash()函数从系统表 SYSIBM.SYSTABLES 中读取 TABLENAME,SCHAMENAME 等表信息。

(2) 得到表中列信息：

编写 db2_get_cols_hash()函数从系统表 SYSIBM.SYSCOLUMNS 中读取表中列的信息，例如 column name，tpey，length 等。

(3) 若该表有外键，check，唯一等约束：

通过 db2_get_checks_hash()，db2_get_rels_hash(), db2_get_unique_hash()函数从系统表 SYSIBM.SYSCHECKS,SYSIBM.SYSRELS,SYSIBM.SYSTABCONS 中获取相应信息。

(4) 若表有索引：

db2_get_indexintab_hash() 从 SYSIBM.SYSCHECKS 中获取该表的索引信息。

(5) 生成 JSON 文件：

set_json_arr_name()把这些结构写入 JSON 文件，生成数据字典。

(6) 生成数据文件：

为了提高导出数据的速度，我们采用了多线程技术来进行处理。通过 GLIB 库函数 g_thread_pool_new()来创建一个线程池，用线程池来管理多个线程[4]。在线程函数中调用 db2_dump_table()来进行数据的导出并生成数据文件。数据文件由文件头和文件体组成。文件头由 12 个字节组成，它指明了该文件的内容信息。具体内容如下：

cmd: 1 字节，指明该数据文件的内容是用来插入，删除还是更新的。(insert, delete, update)

sid: 2 字节，指明对应的 SCHAME ID。

tid: 2 字节，指明对应的表的 ID。

coln: 1 字节，指明该表有多少列。

rown: 2 字节，指明这个表有多少条记录。

len: 4 字节，指明该文件的总长度。

3.2 加载的实现

加载模块在目标端启动，通过 recv_file_use_mq()函数接收传输中间件 MQ(message queue)从源端发来的文件。然后调用 load_file()函数，该函数首先读取接收到的文件的第一个字节。JSON 数据字典的第一个字

© 中国科学院软件研究所 http://www.c-s-a.org.cn

符固定为“{”，而数据文件则不是。由此可以判断接收到的文件是数据字典，还是数据文件。如果是数据字典调用 load_ddl_from_json()函数，用来解析 JSON 文件，组装成对应的 DDL SQL 语句，然后调用 input_sql()函数在目标端数据库执行该 DDL 语句。如果是数据文件，则调用 db2_load_data()函数。此函数负责解析数据文件，生成相应的 DML 语句，并且在目标端数据库执行。

3.3 增量的实现

为了有利于我们进行程序设计，使模块化程度更高，实现高内聚低耦合目标，我们定义了如下重要数据结构：

```
typedef struct{
guint16 counter;//计数
guint16 logver;//日志版本
gchar magic[7];
guint32 seq;//日志序列号
guint64 lsn;
}DB2LogHead;//日志头的结构
typedef struct{
guint32 seq;//块序号
guint16 n;
guint16 len;//长度
guint16 off;//偏移量
guint64 lsn;
}DB2LogBlock;//日志块结构
struct DB2LogRow{
guint32 len;//长度
guint32 seq;//序列号
guint32 off;//偏移量
guchar *data;//数据
};//日志 log_row 结构
```

在全同步启动时，增量同步模块也同时启动。read_log_file()函数用来分析 DB2 数据库日志文件。需要两个外部参数，一个是当前 DB2 数据库正在写的日志文件序号，另外一个是正在写的块号。在 read_log_file()函数内部通过调用 deal_row()函数来具体处理一条日志记录(log_row)。分析出操作类型和具体数据，然后生成相应的 JSON 数据字典与数据文件，通过 MQ 中间件发送到目标端去执行[3]。

4 软件性能测试结果

表 1

源端数据库大小	全同步总时间	每秒导出大小	目标端加载时间	同步延迟时间
6GB	4min	25.1MB	2.6min	34s
4GB	2.8min	24.3MB	2min	30s
10GB	6.8min	25.6MB	4.6min	38s
12GB	8.1min	25.2MB	6min	34s

由测试结果可以看出本软件在性能效率方面还是非常优秀的。与 Q 复制相比，本软件安装成功后，就可以通过客户端图形界面操作和监控。减小了软件使用难度。并且 Q 复制只有针对增量的功能，本软件除此之外还提供了全同步功能。

5 结语

本实时复制软件没有借助 DB2 数据库各种已有的备份接口，而是自己从 DB2 数据库底层出发，设计并实现了该软件。因此无论是从功能，效率，还是速度来说都是其他同类软件无法比拟的。本软件具有良好的模块化结构，有利于后期维护与添加新功能。在性能测试时，无论是全同步所用时间，还是加载所用时间都表现良好。从而尽可能缩短了源端数据库与目标端数据库同步延迟的时间。降低了数据损失的风险，提高了数据库容灾的质量。本软件是在 LINUX 平台下使用 C 语言开发，采用多进程和多线程技术。能够比较容易地实现跨平台。

参考文献

1 牛新庄.深入解析 DB2—高级管理、内部体系结构与诊断案例.北京:清华大学出版社,2009.

2 肖正春,张建伟,林光国,王东明.DB2 V9/9.5 高级应用开发.北京:电子工业出版社,2009.

3 Kernighan BW, Ritchie DM. The C Programming Language (2nd Edition).北京:机械工业出版社,2010.

4 Kernighan BW, Pike R. The UNIX Programming Environment.北京:机械工业出版社,2005.

5 IBM 官方开发网站 http://www.ibm.com/developerworks/.

6 IBM DB2 联机文档.

7 Patrick O. Database: Principles, Programming, and Perforance. 2nd Edition. San Fransisco Morgan Kaufmann Publishers, 2000.

© 中国科学院软件研究所 http://www.c-s-a.org.cn