基于可变时间窗口的增量数据抽取模型

刘 杰 王桂玲 左小将
(北方工业大学计算机学院 北京 100144)
(大规模流数据集成与分析技术北京市重点实验室 北京 100144)

摘 要 基于合适的数据抽取模型持续不断地将变化的数据从各个数据源系统进行抽取集成,是各个异构系统之间进行数据共享融合的关键,也是构建增量式数据仓库来进行数据分析的关键。传统的时间戳变化数据捕获方式存在因数据抽取过程中发生异常而导致数据抽取失效,进而影响数据抽取效率的问题。鉴于此,文中借鉴时间窗口的思想,采用先抽取少量重复记录再去重的做法,对传统的时间戳增量数据捕获模型进行了改进,提出了基于可变时间窗口的增量数据抽取模型。该模型减少了异常对数据抽取的影响,增强了时间戳增量数据抽取 ETL 流程的可靠性,在一定程度上提高了数据的抽取效率。

关键词 变化数据的捕获,增量抽取,时间戳,ETL
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.11.032

Incremental Data Extraction Model Based on Variable Time-window

LIU Jie WANG Gui-ling ZUO Xiao-jiang
(Department of Computer, North China University of Technology, Beijing 100144, China)
(Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, Beijing 100144, China)

Abstract Continuously extracting and integrating the changed data from different data sources based on appropriate data extraction model is crucial for sharing data between different heterogeneous systems and building the incremental data warehouse to analyze data. There exists a problem of efficiency of data extraction in the traditional timestamp based changed-data-capture method. As long as the exception occurs during the data extraction, the whole data extraction progress will fail. In that case, the database must be rolled back, which reduces the efficiency of extraction. To address the problem above, this paper proposed an incremental data extraction model based on variable time-window. The model extracts a small number of repetitive records and then de-duplicates them based on the idea of time-window. The model reduces the influence of the exception on the data extraction, enhances the reliability for extracting ETL process by the timestamp increment data, and improves the efficiency of data extraction.

Keywords Capture of changed data, Incremental extraction, Timestamp, ETL

1 引言

在企业或政府中,由于软件的开发时间或开发部门的不同,往往在不同的软硬件平台上同时运行着多个异构的信息系统,这些系统彼此独立,难以共享数据,为构建数据仓库进行数据分析提出了挑战。ETL(Extract-Transform-Load)是解决这个问题的一个主要技术手段[1],通过 ETL 对数据进行集成,解决了异构数据集成困难的问题,实现了不同部门、不同系统数据之间的共享,同时也降低了数据仓库构建的难度和成本。

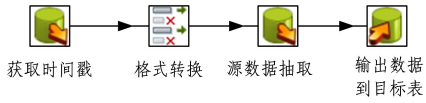
目前,对数据的抽取集成大致可分为全量数据抽取和增量数据抽取两类。全量数据抽取简单直接,相当于数据的迁移或备份,是对源数据系统的相关数据的一次性抽取,适用于首次抽取[2],对此本文不做讨论。增量数据抽取是对源系统中发生变化的数据进行抽取,其关键在于如何捕获发生变化的数据。常见的变化数据捕获(Changed Data Capture,CDC)方式有:时间戳方式、触发器方式、快照表方式、日志方式等[3-8]。时间戳方式要求源数据库表中存在时间属性列,其利用这些属性列来判断哪些数据是增量数据,适用于实时性要求不高的场合[3];触发器方式要求在源系统中建立触发器,具有侵入性;快照表方式需要大量的存储空间来保存快照信息,当表较大时,存在严重的性能问题;日志方式的实施与特定的数据库相关,如 SQL Server 数据库需要在数据库中创建日志表,且日志表的维护也需要编写特定的代码来完成[9-13]。

到稿日期:2017-09-16 返修日期:2018-02-18 本文受北京市自然科学基金(4172018)资助。

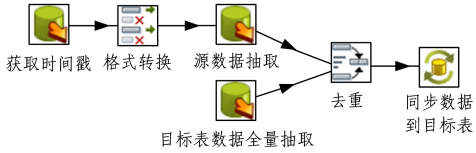
刘 杰(1993-),男,硕士生,主要研究方向为大规模流数据处理、大数据分析;**王桂玲**(1978-),女,副研究员,硕士生导师,CCF 会员,主要研究方向为服务计算、面向服务的数据集成、大规模流数据处理和集成等,E-mail:wangguiling@ict.ac.cn(通信作者);**左小将**(1992-),男,硕士生,主要研究方向为服务计算、大数据处理。

相对于其他 3 种方式,时间戳方式在性能方面相对较好,对源系统的要求相对较低。但是,传统的时间戳增量数据捕获方式存在数据抽取过程中发生异常而导致数据抽取失效、目标数据库回滚,进而影响抽取效率的问题。针对这个问题,本文对时间戳增量数据抽取方式进行了两次改进,最终提出了基于可变时间窗口的增量数据抽取模型,降低了数据抽取过程中发生的异常对数据抽取效率的影响。

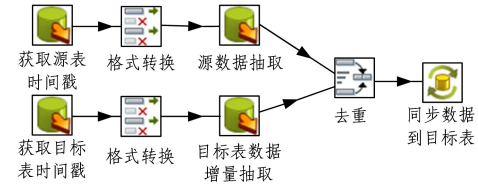
图 1 给出了基于 KETTLE 工具[5]定制的传统时间戳增量数据抽取模型和两次改进的模型时间戳增量数据抽取的流程(以下简称传统模型、一般改进模型和本文模型)。一般改进模型在传统模型的基础上新增了去重操作,本文模型在一般改进模型的基础上减少了从目标数据库表抽取的数据量。



(a)传统时间量数据抽取流程



(b)一般改进时间戳增量数据抽取流程



(c)可变时间窗口增量数据抽取模型数据抽取流程

图 1　3 种模型下的增量数据抽取 ETL 流程

Fig. 1　Incremental data extraction ETL processes of three different models

2　相关工作

寻找一种高效可靠的变化数据捕获方法对变化数据进行捕获并利用合适的 ETL 工具持续不断地将变化的数据从各个数据源系统进行抽取集成,是各个异构系统之间进行数据共享和构建增量式数据仓库进行数据分析的关键[14]。目前,对于变化数据捕获的研究主要有日志方式、触发器方式和时间戳方式。对于采用日志方式的相关工作[9-13],文献[10]提出了基于数据库事务日志文件的全表比对增量抽取方法——L-C 增量抽取方法;文献[11]给出了一个基于消息队列的变化数据捕获框架,能够支持实时地从生产系统捕获变化数据;文献[12]基于读取和分析数据库日志的 CDC 机制来捕获变更数据,给出了一种实时数据更新方案;文献[13]针对基于日志的变化数据捕获的可靠性条件和一般方法进行了研究,提出了一种从日志中抽取变化数据的算法。采用触发器方式的相关工作[2,15],文献[2]结合快照对触发器变化数据捕获方式进行了优化,解决了历史数据无法抽取以及数据源与目标数据端的数据不一致的问题;文献[15]提出了一种将触发器和日志表相结合的变更数据捕获方法,解决在异构数据库中数据的共享及同步问题。采用时间戳方式的相关工作有文献[3,5,16]。文献[3]针对时间戳变化数据捕获方式不能识别删除数据的问题进行了改进;文献[5]对时间戳增量数据抽取的一般步骤进行了详细的描述;文献[16]给出了时间戳增量捕获数据的详细数据流程图。日志方式和触发器方式都是针对某一特定的数据库或 DBMS,时间戳方式不受以上条件的限制。以上研究大多是对不同增量数据抽取方式之间的优缺点进行比较,或针对某种抽取方式的缺点进行改进,更多地关注于提高数据捕获的效率,极少关注其在实际应用中的可靠性。在实际项目中,可靠性甚至比效率更重要。文献[5,16]虽然对时间戳增量抽取进行了研究,给出了时间戳方式捕获增量数据的过程或详细流程图,但是该方法仍存在缺陷,即在数据抽取过程中不允许发生异常,一旦发生异常,本次抽取工作无效,目标表需要回滚,从而影响数据的抽取效率。

3　基于可变时间的窗口增量数据抽取模型

3.1　传统时间戳增量数据抽取模型

时间戳增量数据抽取是通过维护一个额外的数据库表来存储上一次抽取数据的时间[5]。为便于问题的描述,本文给出传统时间戳增量数据捕获算法[16],如算法 1 所示。

算法 1　时间戳增量数据捕获算法

```
1. begin
2.   create time_temp table;//创建时间戳表 time_temp,包含 last_load 和 current_load 字段
3.   init(time_temp);//设置 last_load 为某一特定的值,设置 current_load 为系统当前时间
4.   resultSet:=loadData(source_table);//从源表 source_table 中抽取数据
5.   for(i=0;i<resultSet.size();i++)
6.     record:=resultSet.get(i);
7.     if(isNew(record)) then //判断当前记录是否为新增记录
8.       insert record to target_table;
9.     else
10.      update record to target_table;
11.    end if
12.  end for
13.  last_load:=current_load;
14.  update time_temp;
15. end
```

时间戳增量数据抽取方式在性能上仅次于触发器增量数据抽取方式,但是在实际应用过程中其抽取效率不高。经过多次测试分析,数据抽取过程(算法 1 中的步骤 8 和步骤 10)发生异常致使数据库回滚是造成数据抽取效率低的主要原因。由源数据中存在的重复数据或源数据与源数据间进行关联操作得到的数据中存在的重复数据造成的数据入库时的主键冲突是异常的主要来源;另外,内存溢出、数据库连接断开等也在一定程度上导致了异常的发生。

为了解决这个问题,一种解决方案是对抽取的增量数据(即算法 1 中的 *resultSet*)进行去重,即在算法 1 的步骤 6 后,再增加数据去重操作。加入去重操作后,变化数据到目标表的更新操作发生异常的概率降低,当目标表数据量较少时,数据抽取效率得到一定的改善。但是,如图 2 所示,随着目标表

数据量的增大，数据抽取效率逐渐降低。

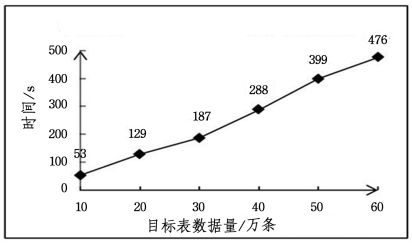


图 2 不同目标表数据量下抽取 1 万条新增数据的耗时对比

Fig. 2 Consuming time of extracting 10 thousand changed data under different target data tables

从上面的分析可以得出，传统的时间戳增量数据抽取存在以下问题：

1)增量数据抽取操作发生异常导致数据库回滚的问题仍然存在，数据库连接断开、内存溢出等异常仍会导致抽取效率下降；

2)去重操作的效率随着目标表的更新操作数据量的增大而降低。

3.2 基于可变时间窗口的增量数据抽取模型的提出

针对传统的时间戳增量数据抽取存在的两个问题，本文从以下两个方面进行考虑：

1)增量数据抽取操作流程发生异常时，目标数据库不回滚；

2)提高去重效率，使去重操作的效率稳定在一定范围内，而不是随着目标表数据量的变化而变化。

定义 1(时间窗口) 时间窗口用于描述时间戳增量数据抽取作业进行数据抽取的时间信息，时间窗口可用一个二元组 $TW=(TS,TF)$ 表示，其中 TS 表示数据抽取的开始时间节点，TF 表示数据抽取的结束时间节点，且 $TS<TF$，时间窗口的大小为 TF 与 TS 之差。例如，某个时间戳增量数据抽取作业对时间节点 t_1，t_2 ($t_1<t_2$) 内的变化数据进行抽取，则此次数据抽取的时间窗口即可表示为 $TW=(t_1,t_2)$，时间窗口大小为 t_2-t_1。

定义 2(数据维护前置时间 Δt) 源系统中同一个时间节点可能存在多条记录，如果在数据抽取的过程中出现异常，那么出现异常的时间节点处的数据在下次抽取时可能会存在部分记录丢失的情况。为此，需要将再次启动数据抽取作业时的数据抽取的开始时间设置为上一次数据抽取的结束时间之前的某个时间点，我们将上次结束时间与本次开始时间之差定义为数据维护前置时间。例如：令连续两次数据抽取作业流程抽取数据时的时间窗口分别为 $TW_1=(t_i,t_{i+1})$ 和 $TW_2=(t_{i+2},t_{i+3})$，那么根据时间窗口 TW 和数据维护前置时间的 Δt 定义，必有 $t_i<t_{i+2}<t_{i+1}<t_{i+3}$ 成立，则数据维护前置时间为：$\Delta t=t_{i+1}-t_{i+2}$。

通常情况下，时间戳增量数据抽取 ETL 流程是以特定的周期(如 1 天、半小时、几分钟等)进行调度执行的[18]，其周期往往固定，因此在不发生异常的情况下，时间窗口的大小是保持不变的。但是如果在数据抽取过程中发生异常，数据抽取流程提前结束(即 TF 变小)，则本次数据抽取的时间窗口相应变小。在增量数据抽取流程发生异常时，目标数据库不回滚的关键在于再次启动数据抽取流程时，扩大数据抽取的时间窗口，抽取上个时间窗口中由于异常而未能抽取到的数据。

图 3 展示了本文希望达到的效果。每次数据抽取作业都抽取一定时间窗口大小的增量数据，当数据抽取过程中出现异常时，模型自动缩小时间窗口大小为数据抽取开始时间至异常发生时的时间所间隔的时间；再次启动抽取流程时又自动扩大时间窗口为异常发生时的时间到数据抽取结束时的时间所间隔的时间，从而能抽取由于上次抽取过程发生异常而未能抽取的数据。如在图 3 中，第 $k+1$ 次数据抽取，应该抽取时间窗口 TW_2' 内的全部增量数据，但是在 t_4 时间节点处发生异常，导致本次抽取数据的时间窗口缩小为 TW_2；在进行第 $k+2$ 次数据抽取时，自动地将原来的时间窗口 TW_3' 调整为 TW_3，抽取 t_4 时间节点之后的增量数据。

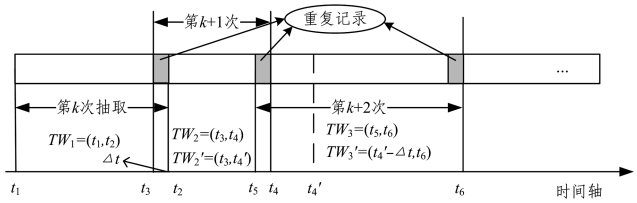


图 3 基于可变时间窗口的增量数据抽取逻辑示意图

Fig. 3 Logical diagram of incremental data extraction based on variable time-window

定义 3(时间窗口内数据记录集) 已知时间窗口 $TW=(t_1,t_2)$，则定义时间窗口大小为 t_2-t_1 的数据库表 T 所包含的所有数据记录为时间窗口 TW 内的数据记录集，记为 $R\_T_{(t_1,t_2)}$。

令连续两次数据抽取作业流程抽取数据时的时间窗口分别为 $TW_1=(t_i,t_{i+1})$ 和 $TW_2=(t_{i+2},t_{i+3})$，则有 $t_i<t_{i+2}<t_{i+1}<t_{i+3}$，那么时间窗口 $TW_3=(t_i,t_{i+3})$ 内目标数据库表 T 的数据记录集为：

$$R\_T_{(t_i,t_{i+3})}=R\_T_{(t_i,t_{i+1})}\cup R\_T_{(t_{i+2},t_{i+3})}$$

定义 4(基于可变窗口的时间戳增量数据抽取模型) 可变窗口的时间戳增量数据抽取模型是依赖于给定的时间窗口进行增量数据抽取的。本文假设在数据抽取过程中发生异常是正常的，而异常的发生往往导致目标表数据与源系统数据不一致。为此，本文给出的解决方案是在进行增量数据抽取前首先给定两个时间窗口(源时间窗口和目标时间窗口，分别用于从源系统和目标数据库中抽取数据)，然后利用给定的时间窗口分别从源系统和目标数据库抽取相应的数据，最后再对这些数据进行去重操作，产生最终的结果集。考虑到数据抽取过程中异常发生对数据一致性的影响，在进行数据抽取前需要利用数据维护前置时间对给定的时间窗口做微小的调整。在此，给出可变窗口的时间戳增量数据抽取模型的形式化定义：$M=(SRC,dst,gmt,\Delta t,TWS,OP,clean,dup)$。

SRC 是一个 n 元组，$S=(s_1,s_2,\cdots,s_n)$ 表示 n 个数据源。

dst 表示目标数据库表，该表要求具有标志记录入库时间的属性列。

gmt 表示从表 dst 中得到数据的入库时间的最大值。

Δt 表示数据维护前置时间，用于调整时间窗口的大小。

TWS 表示数据抽取所需的时间窗口信息，TWS 可以用

一个二元组 $TWS=(TW\_S,TW\_T)$ 表示，其中 TW_S 表示源时间窗口，TW_T 表示目标时间窗口。在模型 M 中，TWS 中 TW_S 和 TW_T 的确定（特别是这两个时间窗口中 TS 的确定）对保证源系统和目标数据库的数据一致性至关重要，特别是在数据抽取过程中发生异常后，再次启动数据抽取作业流程进行数据抽取时，TWS_S 中 TS 的确定直接关系到由于异常发生未能抽取的数据能否被抽取。

定理 1　设目标数据库记录 $R(F_1,F_2,\cdots,minTime)$ 是分别来自 N 个数据源的 N 条记录 $R_1(F_1,F_2,\cdots,time_1)$，$R_2(F_1,F_2,\cdots,time_2)$，$\cdots$，$R_N(F_1,F_2\cdots,time_N)$ 经过某种操作得到的，其中 $minTime=min\{time_1,tim_2,\cdots,time_N\}$。则每次利用上述模型进行数据抽取时，选择目标数据入库时间最大值 $maxVal$ 与数据维护前置时间 Δt 之差作为从源表和目标表抽取增量数据的开始时间，必能保证目标表数据的完整性，证明如下。

证明：令在数据抽取发生异常前更新到目标数据库表的最后一条记录 $R_t=(F_1,\cdots,F_n,minTime)$ 是由来自 $SRC_1,\cdots,SRC_k$ 的 k 个源的 k 条记录 $Rs_1=(F_{11},F_{12},\cdots,Time_1),\cdots,Rs_k=(F_{k1},F_{k2},\cdots,Time_k)$ 经过 OP 操作后得到的，其中 $maxTime=minTime=\min\{Time_1,Time_2,\cdots,Time_k\}$。再次启动数据抽取流程时，有：

$$R\_SRC_{1(Time_1,TF)}\subset R\_SRC_{1(Time_1-\Delta t,TF)}\subset R\_SRC_{1(maxTime-\Delta t,TF)}$$

$$R\_SRC_{2(Time_2,TF)}\subset R\_SRC_{2(Time_2-\Delta t,TF)}\subset R\_SRC_{2(maxTime-\Delta t,TF)}$$

$$\cdots$$

$$R\_SRC_{k(Time_k,TF)}\subset R\_SRC_{k(Time_k-\Delta t,TF)}\subset R\_SRC_{k(maxTime-\Delta t,TF)}$$

对上式中“⊂”左右两侧各个数据源的时间窗口内的数据记录集分别做 OP 操作，则有：

$$R_L=R\_SRC_{1(Time_1,TF)}\ OP\cdots OP\ R\_SRC_{k(Time_k,TF)}$$

$$R_M=R\_SRC_{1(Time_1-\Delta t,TF)}\ OP\cdots OP\ R\_SRC_{k(Time_k-\Delta t,TF)}$$

$$R_R=R\_SRC_{1(maxTime-\Delta t,TF)}\ OP\cdots OP\ R\_SRC_{k(maxTime-\Delta t,TF)}$$

其中，R_L 是理论上为了保证目标数据库表中的数据不丢失所需要抽取的数据记录，R_R 是实际抽取的数据记录。根据 R_L，R_M 和 R_R 的产生式，必有 $R_L\subset R_M\subset R_R$。因此，按照上述方法进行数据抽取取一定能保证目标数据库表中的数据与源数据的一致性。

OP 表示定义在数据源上的操作，OP 也是一个多元组，元组的维数与定义在数据源上的操作有关，如 $OP=(INNERJOIN_{s_1-s_2},\cdots)$，$INNERJOIN_{s_1-s_2}$ 表示将从数据源 s_1 和数据源 s_2 抽取的数据做内连接操作。

clean 表示对从 SRC 中抽取的数据进行简单的清洗操作，即去除多余的时间字段值（来自 SRC 的每一条记录的非最小数据入库时间）。

dup 表示对从 SRC 抽取的数据和从 dst 抽取的数据进行去重操作。

基于可变时间窗口的增量数据抽取模型如图 4 所示，模型工作过程的描述如下：

1）对目标数据库表 dst 进行 gmt 操作，获取 dst 中数据入库时间的最大值 $maxTime$；

2）分别设置 TW_S 和 TW_T 时间窗口中 TS 的值为 $maxTime$，然后根据给定的数据维护前置时间对时间窗口 TW_S 和 TW_T 进行调整，得到 TW_S' 和 TW_T'；

3）以 TW_S' 为新的时间窗口，从数据源 SRC_i 中进行数据抽取并对抽取的数据进行 OP 操作，然后进行 *clean* 操作，去除数据中多余的时间字段值，得到结果集 $tempRS\_SRC$；

4）再以 TW_T' 为目标时间窗口，从 dst 中进行数据抽取，得到结果集 $tempRS\_dst$；

5）对 $tempRS\_SRC$ 和 $tempRS\_dst$ 进行 dup 操作，得到最终结果集 RS；

6）将 RS 装载到目标数据库 dst。

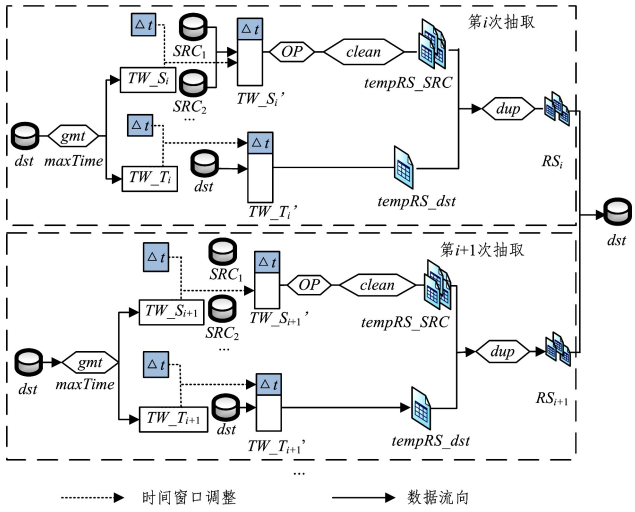


图 4　基于可变时间窗口的增量数据抽取模型

Fig. 4　Incremental data extraction model based on variable time window

3.3　数据去重

数据去重是通过数据之间的比较操作实现的，数据去重的效率与数据之间的比较次数有很大的关系，设法减少数据之间无用的比较是提高去重效率的有效方法。数据去重需要解决两个关键问题：缩小搜索空间和记录匹配[17]。

一般改进模型采用新增的变化数据与整个目标表数据逐一比较的方式对新增数据进行去重。这种去重方式的效率较低的原因主要是：1）无效比较操作占比较大；2）数据逐一匹配的效率太低。一般而言，对于入库时间越早的历史数据，在未来发生修改的可能性越小。基于以上假设，本文利用数据的入库时间对目标表数据进行筛选，仅从目标数据库中选择数据抽取发生异常的时间点之前一段时间的数据而非整个目标表数据与新增数据进行比较，从而达到缩小搜索空间的目的，大大减少了数据之间的对比次数。另外，传统模型的数据匹配效率较低的主要原因是：在整个目标数据库记录集中进行查找需要逐一匹配所有记录，时间复杂度为 $O(n)$。哈希表采用函数映射的思想将记录的存储位置与记录的关键字段关联起来，从而实现快速查找的目的。借鉴哈希表的思想，提出去重算法，如算法 2 所示。

算法 2 基于哈希表的数据去重算法

Input:tempRS_SRC〈key,value〉,即模型中的源数据记录集,其中 key 为关键字段信息,value 为其余字段信息
tempRS_dst〈key,value〉,即模型中的目标数据记录集

Output:ResultSet,去重后的 tempRS_SRC

```
1. begin
2. for record in tempRS_dst
3.    code:=hash(record.key);
4.    HashTable.put(code,record.value)
5. end for
6. for record′ in tempRS_SRC
7.    code′:= hash(record′.key);
8.    v:=HashTable.get(code′);
9.    if(v not equals record′.value ‖ v is null)
10.         ResultSet.put(record′);
11.   end if
12. end for
13. return ResultSet;
14. end
```

3.4 基于可变窗口时间戳增量数据抽取模型的实现

由基于可变窗口的时间戳增量数据抽取模型的定义可知,3.1 节中描述的记录时间戳的中间表的表结构已经无法满足新模型的要求。为此,本文重新设计该中间表(以"CDC_TIME"命名),其结构如表 1 所列。其中,TS_S 和 TF 构成源时间窗口 TW_S,TS_T 和 TF 构成目标时间窗口 TW_T,*pre_time* 表示数据维护前置时间 Δt 且一旦确定就不再改变。

表 1 CDC_TIME 结构

Table 1 Structure of CDC_TIME table

字段	类型	说明
table_name	varchar(30)	主键,当前记录维护的目标表表名
TS_S	datetime	源系统抽取记录的开始时间
TS_T	datetime	目标数据库抽取记录的开始时间
TF	datetime	启动抽取流程时系统的当前时间
pre_time	long	数据维护前置时间(单位:s)

基于可变时间的窗口增量数据抽取模型的实现的最关键问题是时间窗口的维护,CDC_TIME 表的更新算法如算法 3 所示。

算法 3 CDC_TIME 表更新算法

Input:Record〈TF,TS_S,TS_T,TABLE〉,用于从源抽取数据到目标表 TABLE 的时间戳信息

Output:Record′〈TF,TS_S,TS_T,TABLE〉,更新后的 Record

```
1. begin
2.    if (Record is null) then
3.    TS_S:=1970-01-01 12:00:00;
4.    TS_T:=sysDate;//系统当前时间
5.  end if
6.  TF:=sysDate;
7.  start Job until finished;//job:增量数据抽取作业
8.  maxVal:=getMaxTime(TABLE);//从目标数据库表中获取数据更新时间的属性列的最大值
9.  TS_S:=maxVal-pre_time;
10. TS_T:=maxVal-pre_time;
11. update Record;//更新 Record 的值
12. return Record;
13. end
```

首次抽取时,需要抽取所有数据源的所有数据,因此首次抽取时需要设置 TS_S 为一个较早的时间,本文将 1970-01-01 12:00:00 作为 TS_T 的初始值;另外,首次抽取时目标数据库表为空,因此可以将 TS_T 的值设置为系统当前时间。除首次抽取外,每一次抽取数据的源开始时间 TS_S 和目标开始时间 TS_T 的值都是当前目标数据库表中数据入库时间最大值 $maxVal$ 与数据维护前置时间之差。

4 实验与分析

传统的时间戳增量数据捕获模型在可靠性方面存在不足,本文对此做出了相应的改进。为了验证设计的有效性,本文在可靠性和数据抽取效率方面分别对图 1 所示的流程进行了对比和分析。

4.1 实验环境

本文实验是在 64 位 Windows 系统下进行的。系统版本为 Windows 10 专业版;CPU 为 Intel i5-2400,四核;主频为 3.10GHz;内存为 4GB;硬盘为 1TB 机械硬盘。CPU、内存和硬盘空闲时负载分别为 24%,45%和 27%,负载来源主要由系统基础软件运行产生。本次实验采用的数据库环境为Oracle 11g Release2 Standalone。

4.2 可靠性对比分析

本实验在可靠性方面对传统模型、一般改进模型和本文改进模型进行了数据抽取测试对比,分别验证 3 个模型在内存溢出异常、数据库连接异常时的可靠性。表 2 列出了对表中的 19880536 条记录进行抽取的过程中,出现内存溢出异常时 3 个模型的可靠性对比;表 3 列出了数据抽取过程中出现数据库连接异常时 3 个模型的可靠性对比。为了避免内存溢出对实验结果的影响,将待抽取数据的记录数降低为 300 万。由于抽取的数据量相对较小,抽取过程中发生异常的情况很少,因此在实际数据抽取过程中通过拔出网线人为制造了数据库连接异常。

表 2 内存溢出异常情况下的可靠性对比

Table 2 Reliability comparison under exception of out of memory

模型	首次出现异常目标表数据量	异常次数	目标表数据量	抽取结果
传统模型	7517000	∞	7517000	失败
一般模型	7528000	∞	7528000	失败
本文模型	7539000	2	19880536	成功

表 3 数据库连接异常情况下的可靠性对比

Table 3 Reliability comparison under exception of database connection

模型	首次出现异常目标表数据量	异常次数	目标表数据量	抽取结果
传统模型	1506000	∞	1506000	失败
一般模型	1428091	1	3000000	成功
本文模型	1490000	1	3000000	成功

在表 2 中,传统模型和一般改进模型在发生内存溢出异常时,无法继续抽取数据;而本文改进模型在出现内存溢出异常后,再次启动数据抽取作业流程仍能继续抽取数据。在表

3 中，传统模型在出现数据库连接异常后，无法继续抽取数据，而一般改进模型和本文改进模型在出现数据库连接异常后，重新启动数据抽取流程，均能继续抽取数据，但是一般改进模型抽取数据耗时较长。综合以上两点，在异常发生的情况下，本文改进模型的可靠性更好。

4.3　性能对比分析

为了进一步验证基于可变窗口的增量数据抽取模型的性能，本文将所提模型与传统模型、一般改进模型进行对比。由于模型是以可靠性为前提的，因此本次实验主要对比分析数据抽取过程中发生异常的情况下 3 种模型的抽取效率。本次对比实验分两次进行：1）固定目标表数据量（100 万条），依次增加源数据量；2）固定源数据量（100 万条），依次增加目标表数据量。由于在实际数据抽取过程中发生异常的几率相对较小，因此本次实验采用手动干扰的方式制造异常，两次对比实验分别在目标表和源表数据抽取到 45 万条时发生异常，对于发生异常而未能成功抽取数据的模型，统计时间均以最大值计算。比较结果如图 5 和图 6 所示。

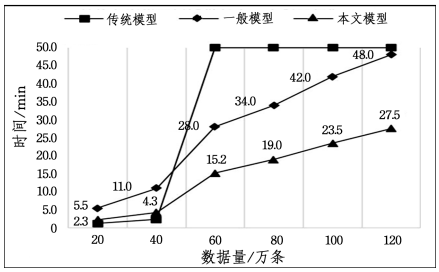


图 5　目标表数据量固定情况下 3 个模型的性能对比

Fig. 5　Performance comparison of three models in case of fixed target table data

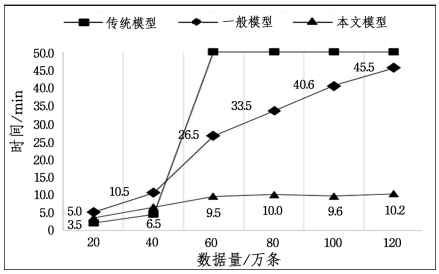


图 6　源表数据量固定情况下 3 个模型的性能对比

Fig. 6　Performance comparison of three models in case of fixed source table data

从图 5 和图 6 中的对比数据可以看出，虽然在抽取过程未发生异常的情况下，本文提出的模型抽取效率较传统模型低，但是，在发生异常时，传统模型无法继续抽取数据，而一般改进模型与本文提出的模型均能继续抽取数据；而且相比于一般改进模型，本文提出的模型的效率提高了近一倍。将传统模型和一般改进模型的抽取效率进行比较发现，去重操作是造成数据抽取效率降低的主要原因。本文改进模型在一般改进模型的基础上进行了优化，减少了数据的对比量，从而在一定程度上提高了数据抽取的效率；同时，从图 6 中也可以看出，本文提出的模型避免了一般改进模型的抽取效率随着目标表数据量的增大而逐渐降低的问题。

结束语　本文对时间戳变化数据捕获方式进行了研究，针对时间戳增量数据抽取存在的问题进行了优化，虽然在不发生异常的前提下，与传统的抽取模型相比，本文提出的改进方案仍然存在差距，但是其降低了数据抽取流程发生异常对数据抽取工作的影响，提高了时间戳方式捕获变化数据的流程的可靠性，同时兼顾了数据抽取的效率。

目前，本文提出的数据维护前置时间需要手动设置，并且该值的大小没有统一的标准，其数值对数据抽取的效率存在一定的影响。我们将在以后的研究中根据源表数据的变化频率进行个性化设置，将该值对数据抽取效率的影响降到最小。

参 考 文 献

[1] MINAKSHI M，SHARMA H C. Near Real-Time Data Warehousing Using State-of-the-Art ETL Tools[J]. International Journal of Research，2014，41(10)：100-117.

[2] SHU Q. The Research on Optimization of ETL Process and Incremental Data Extraction[D]. Changsha：Hunan University，2011.（in Chinese）
舒琦. ETL 过程优化与增量数据抽取的研究[D]. 长沙：湖南大学，2011.

[3] WEN L. Design and Implementation of Incremental Data Extractor based on Sector Inquiry[D]. Shijiazhuang：Hebei University of Science and Technology，2015.（in Chinese）
温璐. 基于区段查询的增量数据抽取器的设计与实现[D]. 石家庄：河北科技大学，2015.

[4] TANK D M，GANATRA A，KOSTA Y P，et al. Speeding ETL Processing in Data Warehouses Using High-Performance Joins for Changed Data Capture（CDC）[C]// Advances in Recent Technologies in Communication & Computing International Conference. 2010：365-368.

[5] CASTERS M，BOUMAN R，DONGEN J V. Pentaho Kettle Solutions：Building Open Source ETL Solutions with Pentaho Data Integration[M]. Indianapolis，Indiana：Wiley Publishing，2010.

[6] JORG T，DESLOCH S. Towards generating ETL processes for incremental loading[C]// International Database Engineering and Applications Symposium. 2008：101-110.

[7] MEKTEROVIC I，BRKIC L. Delta view generation for incremental loading of large dimensions in a data warehouse[C]// International Convention on Information and Communication Technology Electronics and Microelectronics. 2015：1417-1422.

[8] LIN Z Y，YANG D Q，SONG G J，et al. Change Data Capture in Real-Time Active Data Warehouses：A Survey[J]. Journal of Computer Research and Development，2007，44(Z3)：447-451.（in Chinese）
林子雨，杨冬青，宋国杰，等. 实时主动数据仓库中的变化数据捕捉研究综述[J]. 计算机研究与发展，2007，44(Z3)：447-451.

[9] SHI J G，BAO Y B，LENG F L，et al. Study on Log-Based Change Data Capture and Handling Mechanism in Real-Time Data Warehouse[C]// International Conference on Computer Science and Software Engineering. IEEE Computer Society，2008：478-481.

（下转第 230 页）

刘群.基于句法的统计机器翻译模型与方法[J].中文信息学报,2011,25(6):63-71.

[4] KOEHN P,OCH F J,MARCU D. Statistical Phrase-based Translation[C]// Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics,2003:48-54.

[5] BAHDANAU D,CHO K,BENGIO Y. Neural Machine Translation by Jointly Learning to Align and Translate[J]. arXiv:1409.0473,2014.

[6] LUONG M T,PHAM H,MANNING C D. Effective Approaches to Attention-based Neural Machine Translation[J]. arXiv:1508.04025,2015.

[7] SUTSKEVER I,VINYALS O,LE Q V. Sequence to Sequence Learning with Neural Networks[C]// Advances in Neural Information Processing Systems. 2014:3104-3112.

[8] LI Y C,XIONG D Y,ZHANG M. A survey of Neural Machine Translation[J/OL]. Chinese Journal of Computers. http://cjc.ict.ac.cn/online/bfpub/lyc-20171229152034.pdf. (in Chinese)
李亚超,熊德意,张民.神经机器翻译综述[J/OL].计算机学报.http://cjc.ict.ac.cn/online/bfpub/lyc-20171229152034.pdf.

[9] GEHRING J,AULI M,GRANGIER D,et al. Convolutional Sequence to Sequence Learning[J]. arXiv preprint arXiv:1705.03122,2017.

[10] XU K,BA J,KIROS R,et al. Show,Attend and Tell:Neural Image Caption Generation with Visual Attention[C]// International Conference on Machine Learning. 2015:2048-2057.

[11] CHENG Y,WU H,WU H,et al. Agreement-based joint training for bidirectional attention-based neural machine translation[J]. arXivpreprint arXiv:1512.04650,2015.

[12] TU Z,LU Z,LIU Y,et al. Modeling Coverage for Neural Machine Translation[J]. arXiv preprint arXiv:1601.04811,2016.

[13] VASWANI A,SHAZEER N,PARMAR N,et al. Attention Is All You Need[C]// Advances in Neural Information Processing Systems. 2017:5998-6008.

[14] CHO K,MERRIENBOER B,GULECHRE C,et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation[J]. arXiv preprint arXiv:1406.1078,2014.

[15] HOCHREITER S,SCHMIDHUBER J. Long short-term memory[J]. Neural Computation,1997,9(8):1735-1780.

[16] HOCHREITER S,SCHMIDHUBER J. Long short-term memory[J]. Neural Computiton,1997,9(8):1735-1780.

[17] NEUBIG G,DYER C,GOLDBERG Y,et al. DyNet:The Dynamic Neural Network Toolkit[J]. arXiv:1701.03980.

[18] NEUBIG G. Lamtram:A toolkit for language and translation modeling using neural networks[OL]. http://www.github.com/neubig/lamtram,2015.

[19] PAPINENI K,ROUKOS S,WARD T,et al. Bleu:a Method for Automatic Evaluation of Machine Translation[C]// Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics,2002:311-318.

(上接第 209 页)

[10] JIA Y K. Research and Design on Data Extraction in Multiple Data Sources[D]. Harbin:Harbin Engineering University,2013.(in Chinese)
贾艳凯.多源异构增量数据抽取方法研究与设计[D].哈尔滨:哈尔滨工程大学,2013.

[11] YANG L. Design and Implementation of Real-time data extraction Mechanism in data warehousing [D]. Beijing:Beijing University of Posts and Telecommunications,2007.(in Chinese)
杨乐.数据仓库中实时抽取机制的研究与实现[D].北京:北京邮电大学,2007.

[12] TAN G W,WU T. Study on Method of Data Warehouse Real-time Data Updating Based on Mechanism of CDC [J]. Computer Science,2015,42(S1):546-548.(in Chinese)
谭光玮,武彤.基于 CDC 机制的数据仓库实时数据更新方法研究[J].计算机科学,2015,42(S1):546-548.

[13] ZOU X X,JIA W J,PAN J H. Research of Log-based Change Data Capture [J]. Journal of Chinese Computer Systems,2012,33(3):531-536.(in Chinese)
邹先霞,贾维嘉,潘久辉.基于数据库日志的变化数据捕获研究[J].小型微型计算机系统,2012,33(3):531-536.

[14] JAIN T,SALUJA S. Refreshing Datawarehouse in Near Real-Time[J]. International Journal of Computer Applications,2012,46(18):24-29.

[15] WANG Y B,RAO X R,HE P. Incremental database synchronization update mechanism under heterogeneous environment [J]. Computer Engineering and Design,2011,32(3):948-951.(in Chinese)
王玉标,饶锡如,何盼.异构环境下数据库增量同步更新机制[J].计算机工程与设计,2011,32(3):948-951.

[16] CUI Y W,ZHOU J H. Research on Data Integration Based on KETTLE [J]. Computer Technology and Development,2015(4):153-157.(in Chinese)
崔有文,周金海.基于 KETTLE 的数据集成研究[J].计算机技术与发展,2015(4):153-157.

[17] LIU X Q,WU G,DENG H P. Data deduplication in Web information integration[J]. Journal of Computer Applications,2013,33(9):2493-2496.(in Chinese)
刘雪琼,武刚,邓厚平.Web 信息整合中的数据去重方法[J].计算机应用,2013,33(9):2493-2496.

[18] TANK D M. Reducing ETL Load Times by a New Data Integration Approach for Real-time Business Intelligence[J]. International Journal of Engineering Innovations & Research,2012,1(2):56-60.