EAI Endorsed Transactions
on Collaborative Computing

Research Article EAI.EU

# VTWM: An Incremental Data Extraction Model Based on Variable Time-Windows

Weixing Jia[1], Yang Xu[2], Jie Liu[3] and Guiling Wang[1,*]

[1]Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, School of Information Science and Technology, North China University of Technology, No. 5 Jinyuanzhuang Road, Shijingshan District, Beijing 100144, China
[2]Tianjin E-Hualu Information Technology Co., Ltd, No.1 Tianhua Road, Balitai Industrial Park, Jinnan District, Tianjin 300350, China
[3]Beijing Yidian Wangju Technology Co., Ltd, No. 30, Shixing Street, Shijingshan District, Beijing 100103, China

## Abstract

Continuously extracting and integrating changing data from various heterogeneous systems based on an appropriate data extraction model is the key to data sharing and integration and also the key to building an incremental data warehouse for data analysis. The traditional data capture method based on timestamp changes is plagued with anomalies in the data extraction process, which leads to data extraction failure and affects the efficiency of data extraction. To address the above problems, this paper improves the traditional data capture model based on timestamp increments and proposes VTWM, an incremental data extraction model based on variable time-windows, based on the idea of extracting a small number of duplicate records before removing duplicate values. The model reduces the influence of abnormalities on data extraction, improves the reliability of the traditional data extraction ETL processes, and improves the data extraction efficiency.

**Keywords:** change data capture, incremental data extraction, timestamp, ETL

Received on 27 August 2020, accepted on 06 September 2020, published on 09 September 2020



doi: 10.4108/eai.12-6-2020.166291

*Corresponding author. Email: wangguiling@ncut.edu.cn

## 1. Introduction

In enterprises or government departments, due to the different development times and different development agencies, there are often multiple heterogeneous information systems running on different hardware and software platforms at the same time. These systems have some features such as independence from each other, the fact that it is difficult to share data and so on. These features raise the challenge to build data warehouse for data analysis. ETL (Extract-Transform-Load) is one of the main technical means to solve this problem [1]. Data integration through ETL solves the problem of difficult integration of heterogeneous data, and realizes the integration and sharing of the data between different departments and different system. At the same time, ETL technology also reduces the difficulty and cost of constructing data warehouse.

At present, the data extraction methods can be roughly divided into two categories: full data extraction and incremental data extraction. The full data extraction is simple and direct. It is a one-time extraction of the relevant data of the source data system, equivalent to data migration or data backup. This method is suitable for the first extraction [2], which is not the focus of this paper. Incremental data extraction is the extraction of data that has changed in the source system. The key to incremental data extraction is how to capture the data that has changed. Time Stamping Mode, Triggering Mode, Snapshot Mode, and Log Mode are common Changed Data Capture (CDC) methods [3-8]. For the Time-Stamping Mode or Incremental Timestamp-based Data Capture Mode, it requires the existence of a time attribute columns in the source database table and uses these attributes columns to determine which data is incremental data. It applies to occasions less demanding real-time [3]. For the Triggers Mode, it requires

the establishment of triggers in the source system, and this method is invasive. For the Snapshot Mode, it requires a lot of storage space to save the snapshot information, but there are serious performance problems when the tables are relatively large. The implementation of the Log Mode with a specific database, for example, SQL Server database need to create a log table in the database, and log table maintenance also need to write a specific code to complete [9-13].

Compared with the other three methods, the Time Stamping Mode is relatively both good in performance and has low requirements on the source system. However, the traditional method of Incremental Timestamp-based Data Capture is plagued with anomalies in the process of data extraction, leading to data extraction failure and target database rollback, and affecting the extraction efficiency in the end. To solve this problem, in this study two improvements are made to the way of extracting data based on incremental timestamps. An incremental data extraction model based on variable time-windows (abbreviated as VTW Model or VTWM) is proposed to reduce the impact of abnormalities on data extraction efficiency during data extraction process.

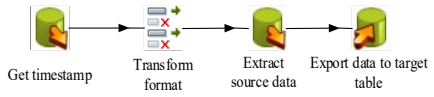


Fig.a Traditional Incremental Timestamp-based Data Extraction Process

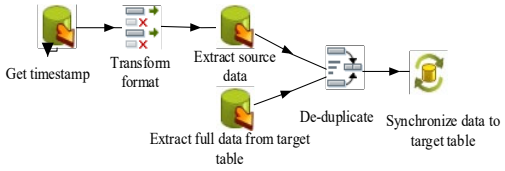


Fig.b Improved Incremental Timestamp-based Data Extraction Process

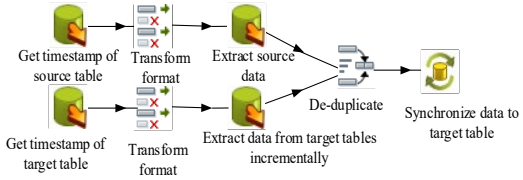


Fig.c Incremental Data Extraction Process Based on Variable Time-Windows

**Figure 1.** Incremental data extraction ETL process for three different models

Figure 1 gives a traditional timestamp incremental data extraction model customized by KETTLE [5] and two improved incremental timestamp-based data extraction workflows (hereinafter referred to as Traditional Model, General Improvement Model and VTW (Incremental Data Extraction Model Based on Variable Time-Windows) Model. The General Improvement Model adds the de-duplication operation to the Traditional Model. The VTW Model proposed in this paper is optimized on top of the General Improvement Model by reducing the amount of data extracted from the target database table.

The main contribution of VTWM proposed in this paper is that it alleviates the problem that the database rollback and the efficiency of the de-duplication operation decrease with the increase of the data volume of the target table due to the exception of the up-extraction operation of the traditional model. It reduces the impact of anomalies on data extraction by taking into account the efficiency of data extraction under the premise of ensuring reliability. The main idea is to avoid database rollback in the case of exception by extracting a small number of duplicate records before removing duplicate values based on variable time-windows. The structure of the rest of the paper is as follows. Section 2 is the related work. In Section 3, we introduce the methodology of this paper, including the basic definitions related with VTWM and the core algorithms. Section 4 introduces the implementation of VTWM. Section 5 provides evaluation results. Section 5 concludes the paper.

## 2. Related Work

Finding a highly efficient and reliable method for capturing change data and leveraging the right ETL tools for extracting changing data from various systems continuously are the key to share data across heterogeneous systems and build incremental Data Warehouse for data analysis [14]. At present, there are mainly three main approaches for the study of change data capture: Log-based approach [9-13], Trigger-based approach [2,15] and Timestamp-based approach[3,5,16] A full-table comparison incremental extraction method based on database transaction log files, called L-C incremental extraction method, is proposed in [10]. A framework of capturing change data based on message queue that enables real-time change data capture is presented in [11]. A CDC mechanism is designed in [12] to capture change data based on reading and analyzing database logs, giving a real-time data update scheme. Research work [13] investigates the reliability conditions and general methods of capturing changed data based on logs, and proposes an algorithm for extracting changed data from logs. An optimization of the traditional trigger-based approach is proposed in [2] using snapshots, which can solve the problem of history data extraction and data inconsistency between data source and target. A method combining trigger-based approach and log-based approach is proposed in [15], solving the problem of data sharing and synchronization in heterogeneous databases. The traditional timestamp-based approach is improved in [3] for solving the problem of deleted data recognition. The general procedure for extracting data incrementally based on timestamp is presented in [5] in details. A detailed data flow for incremental timestamp-based data capture is presented in [16]. Log-based approach and Trigger-based approach are both for a specific database or DBMS, while the Timestamps-based approach is not limited by certain types of database. The related work discussed above focus more on comparing between the advantages and disadvantages of different incremental data extraction approaches, or improving the traditional approaches addressing their

shortcomings. The related work pays more attention on improving the efficiency of data capture than its reliability in practical application. In real world projects, however, reliability is even more important than efficiency. Although the related work [5,16] have studied the incremental timestamp based data capture method and given the detailed capturing process, the timestamp-based approach had drawbacks that data extraction process does not allow an exception occurs. If the event of an exception happened, the extraction will be invalid and the target table needs to be rolled back, thus affecting the efficiency of data extraction.

## 3. Methodology

### 3.1 Traditional incremental timestamp-based data extraction model

Incremental timestamp-based data extraction is achieved by maintaining an additional database table to store the time of the last data extraction [5]. To facilitate the description of the problem, this paper firstly presents the traditional incremental timestamp-based data extraction algorithm [16].

Algorithm 1 Incremental timestamp-based data extraction algorithm

```
1.begin
2.      create time_temp table; //create a timestamps table time_temp, including the last_load and current_load fields
3.      init(time_temp); //set the last_load for a particular value and set the current_load for the system current time
4.      resultSet:=loadData(source_table); //extracting data from the source table source_table
5.      for (i=0; i<resultSet.size(); i++)
6.              record:=resultSet.get(i);
7.              if(isNew(record)) then //determine whether the current record is a new record
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
```

The incremental timestamp-based data extraction method is second only to the trigger incremental data extraction method in terms of performance, but its extraction efficiency is not high in the actual application. We conducted many tests on the traditional incremental timestamp-based data extraction method. We found that the main reason for the low efficiency of data extraction is database rollback caused by abnormal in the process of data extraction (steps 8 and 10 in algorithm 1). The duplicate records in the extracted data originating from data sources or duplicate records that occurs after correlation operations. The presence of duplicate records causing a primary key conflict when updating data to the target database is the main source of anomalies. In addition, memory overflows and disconnected database connections also contribute to the exceptions to some extent.

To solve this problem, one solution is to de-duplicate the extracted incremental data (*resultSet* in Algorithm 1), that is, adding an additional operation of de-duplicate data after Step 6 in Algorithm 1. After the de-duplication operations are added, the probability of abnormalities in updating the changed data to the target table is reduced, and when the amount of data in the target table is small, the data extraction efficiency is improved to a certain extent. However, as shown in Figure. 2, as the amount of data in the target table increases, the data extraction efficiency gradually decreases.

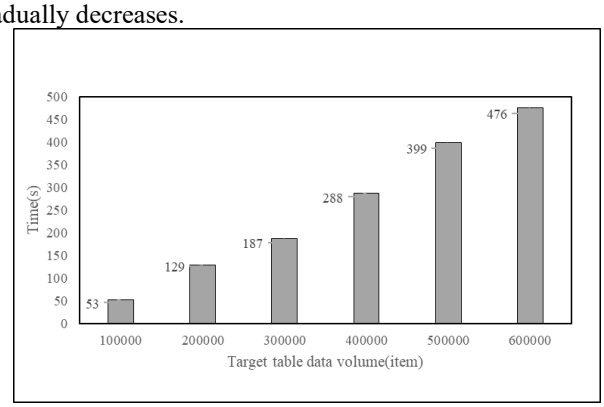


**Figure 2.** Consuming time of extracting 10 thousand changed data under different target data tables

From the above analysis, it can be concluded that the traditional incremental timestamp-based data extraction still has the following problems:

1) The problem of database rollback due to abnormalities in incremental data extraction operations still exists, such as database disconnects and memory overflows, these exceptions can still lead to a reduction in extraction efficiency.

2) The efficiency of the de-duplication operation decreases as the amount of data in the target table increases.

### 3.2 Definitions

For that two problems existing in the traditional incremental timestamp-based data extraction, this paper considers the following two aspects:

1) The target database is not rolled back when an exception occurs in the incremental data extraction process.

2) Improve the efficiency of the de-duplication and try to keep the efficiency of the de-duplication operation stable within a certain range. Instead of changing with the amount of data in the target table, the efficiency of the de-duplication operation should be kept within a certain range.

**Definition 1. Time window.** The time window is used to describe the time information at which the incremental timestamp-based data extraction job is performing data extraction. The time window can be represented by a two-tuple *TW= (TS, TF)*, which *TS* represents the start time node of data extraction and *TF* represents the end time node of the

data extraction, and $TS<TF$. The size of the time window is TF minus TS. For example, a certain time-stamped incremental data extraction process job extracts the changing data in time node $t_1$, $t_2(t_1<t_2)$, then the time window for this data extraction can be expressed as $TW= (t_1, t_2)$ and the time window size is $t_2-t_1$.

**Definition 2. Data maintenance variable time $\Delta t$.** There may be multiple records in the same moment of the source system. If an exception occurs during data extraction, the data at the moment when the exception occurs may have a partial record loss at the next extraction. For this purpose, the start time of data extraction when the data extraction job is started again needs to be set as a certain time point before the end time of the last data extraction, and the difference between the last end time and the start time is defined as data maintain the lead time. For example, let $TW_1= (t_i, t_{i+1})$ and $TW_2= (t_{i+2}, t_{i+3})$ be two consecutive data extraction workflows. There must be a $t_i<t_{i+2}<t_{i+1}<t_{i+3}$ based on $TW$ and $\Delta t$ , and $\Delta t=t_{i+1}-t_{i+2}$.

Normally, the ETL process of incremental timestamp-based data extraction is scheduled [18] with a specific period, such as one day, half an hour, several minutes and so on, and its period is often fixed. Therefore, if no abnormality occurs, the time window size tends to stay the same. However, if an abnormality occurs in the data extraction process and the data extraction process ends prematurely, that is, the $TF$ becomes smaller, the size of the time window in this data extraction will be correspondingly smaller. In such a data extraction workflow, the key to ensure that the target database is not rolled back when an exception occurs in the incremental data extraction process is to expand the time window for data extraction when starting the data extraction process again and extract the data that could not be extracted due to an exception in the previous time window.

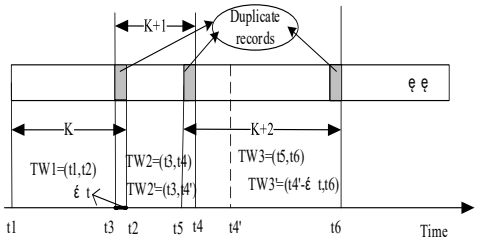


**Figure 3.** Logical diagram of incremental data extraction based on variable time-windows

Figure 3 shows what this paper hopes to achieve. Each time the data extraction workflow starts, incremental data of a certain time window size is extracted, and the size of time window is automatically reduced. And the size of the time window is from the start time of data extraction to the time that the exception occurs. When the extraction process is started again, the size of time window will enlarge automatically. And the size is from the time that the exception occurs to the end time of data extraction. The data that could not be extracted due to the exception during the previous extraction process will be extracted. For example, in Figure3, due to an exception occurs at the time node $t4$, the time window of the extracted data that should have been extracted in the time window $TW_2'$ is reduced to $TW_2$ when the $k$ + 1th data extraction. When the $k$+2th data extraction, it is automatically for the original time window $TW_3'$ adjusted to $TW_3$, extracted after $t4$ time node incremental data.

**Definition 3. Data record set within time window.** Given that the time window $TW= (t_1,t_2)$ then all the data contained in the database table $T$ which the size of time window is $t_2-t_1$ is defined as data record set in time window, recorded as $R\_T_{(t1,\ t2)}$. Let $TW_1=(t_i,t_{i+1})$ and $TW_2=(t_{i+2},t_{i+3})$ are the time window of two consecutive data extraction workflows to extracting data, and $t_i<t_{i+2}<t_{i+1}<t_{i+3}$. then the formula of the data record set of the target database table $T$ in the time window $TW_3$= ($t_i$, $t_{i+3}$) is as follows:

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

**Definition 4. VTWM (Incremental data extraction model based on variable time-windows)**. VTWM relies on incremental data extraction from a given time window. This paper assumes that it is normal for an exception to occur during the data extraction process, and that anomalies often result in inconsistencies between the target table data and the source system data [19]. For that reason, the solution given in this paper is to give two time windows before incremental data extraction. And the one is source time window for extracting data from source system and the other one is target time window for extracting data from target database. Then extract the corresponding data using the given time window from the source system and the target database, respectively. Finally, these data are deduplicated to obtain the final set. Taking into account the consistency of the data be affected by the anomalies of data extraction process, it is needful to make a small adjustment for a given time window use $\Delta t$ before the data extraction. Therefore, we give a formal definition of incremental timestamp-based data extraction model based on variable windows:

$$M = (SRC, dst, gmt, \Delta t, TWS, OP, clean, dup)$$

$SRC$ is an n-tuple with $S= (s_1, s_2, \ldots\ldots, s_n)$, represents n data sources.

$dst$ represents the target database table, which requires an attribute column that marks the storage time.

$gmt$ represents the maximum time of the data put into the database from the $dst$ data obtained.

$\Delta t$ represents data maintenance variable time, used to adjust the time window size.

$TWS$ represents the time window information required for data extraction. $TWS$ can be represented by a two-tuple $TWS= (TW\_S, TW\_T)$, and $TW\_S$ represents the source time window and $TW\_T$ represents the target time window. In M, the determination of $TW\_S$ and $TW\_T$ in $TWS$ (especially the determination of $TS$ in these two time windows) is crucial to ensure data consistency between the source system and the target database, especially when the data extraction workflow is started again for data extraction after an exception occurs during data extraction, the determination of $TS$ in $TWS\_S$ is directly related to whether the data that cannot be extracted due to the exception can be Extracted.

**Theorem 1**: Let the target database records $R\ (F_1, F_2, ..., minTime)$ be $N$ records respectively from $N$ data sources, and $R_1\ (F_1, F_2, ..., time_1)$, $R_2\ (F_1, F_2, ..., time_2)$ ..., $R_{(N)}\ (F_1, F_2, ..., time_N)$ is obtained through some operations, where $minTime = min\ \{time_1, time_2, ..., time_N\}$. Therefore, each time the above model is used for data extraction, the difference between the maximum value of the target data storage time $maxVal$ and data maintenance variable time $\Delta t$ as the start time of extracting incremental data from the source table and the target table, the integrity of the target table data can be guaranteed. It is proved as follows.

**Prove:** Set up the last record $R_t=(F_1,...,F_n,minTime)$ that was updated to the target database table before the abnormal happened of data extraction is the $k$ records from the $k$ source from $SRC_1,...,SRC_k$:$R_{s1}=(F_{11},F_{12},...,Time_1),...,R_{sk}=(F_{k1},F_{k2},...,Time_k)$ obtained after OP operation, where $maxTime=minTime=min\{Time_1,Time_2,...,Time_k\}$. When the data extraction process is started again, the following formula is satisfied:

$$R\_SRC_{1(\mathrm{Time1},\,TF)} \subset R\_SRC_{1(Time1-\Delta t,\,TF)} \subset R\_SRC_{1(maxTime-\Delta t,\,TF)}$$
$$R\_SRC_{2(Time,\,TF)} \subset R\_SRC_{2(Time2-\Delta t,\,TF)} \subset R\_SRC_{2(maxTime-\Delta tTF)}$$
$$R\_SRC_{\mathrm{k}\,(Timek,\,TF)} \subset R\_SRC_{\mathrm{k}(Timek-\Delta t,\,TF)} \subset R\_SRC_{\mathrm{k}(maxTime-\Delta t,TF)}$$

Perform the OP operation on the data record set from the data sources of the left and right sides of the symbol $\subset$ in the above formula to respectively. And there are:

$$R_L = R\_SRC_{1(Time1,\,TF)} OP \ldots OP\ R\_SRC_{k\,(Timek,\,TF)}$$
$$R_M = R\_SRC_{1(Time1-\Delta t,\,TF)} OP \ldots OP\ R\_SRC_{k\,(Timek-\Delta t,\,TF)}$$
$$R_R = R\_SRC_{1(\mathrm{max}Time1-\Delta t,\,TF)} OP \ldots OP\ R\_SRC_{k\,(\mathrm{max}Timek-\Delta t,\,TF)}$$

$R_L$ represents the data records that is need to be extracted in order to ensure that the target database table data is not lost. $R_R$ is the actually extracted data record. According to the production of $R_L$, $R_M$ and $R_R$, there must be $R_L \subset R_M \subset R_R$. Therefore, data extraction according to the above methods will be able to ensure that the target database table data and source data consistency.

*OP* represents the operation defined on the data source. *OP* is also a tuple whose dimensions are related to the operations defined on the data source, such as $OP=(INNERJOIN_{s1\text{-}s2}, \ldots\ldots)$, $INNERJOIN\ s_1\text{-}s_2$ represents the data extracted from data source s1 and data source s2 are interconnected.

*clean* represents a simple cleaning operation for data extracted from *SRC*, removes the extra time field value. The extra time field value is each record that not the minimum data storage time in the *SRC*.

*dup* represents de-duplication operation of data extracted from *SRC* and data extracted from *dst*.

As shown in Figure 4, the working process of the model is described as follows:

1) perform *gmt* operation on the target database table *dst* to get the maximum value of *maxTime* for the data storage time in *dst.*

2) set the *TS* value as *maxTime* in the time window of *TW_S* and *TW_T*, and then the time window *TW_S* and *TW_T* are adjusted to obtain *TW_S'* and *TW_T'* according to the given $\Delta t$.

3) take *TW_S'* as a new time window to extract data from the data source $SRC_i$ and perform *OP* operation on the extracted data, and then perform a clean operation, Remove the extra time field value in the data to get the result set *tempRS_SRC*;

4) take *TW_T'* as the target time window to extract the data from *dst* to get the result set *tempRS_dst*.

5) perform *dup* operation on *tempRS_SRC* and *tempRS_dst* to obtain the final result set *RS*;

6) load the *RS* into the target database *dst*.

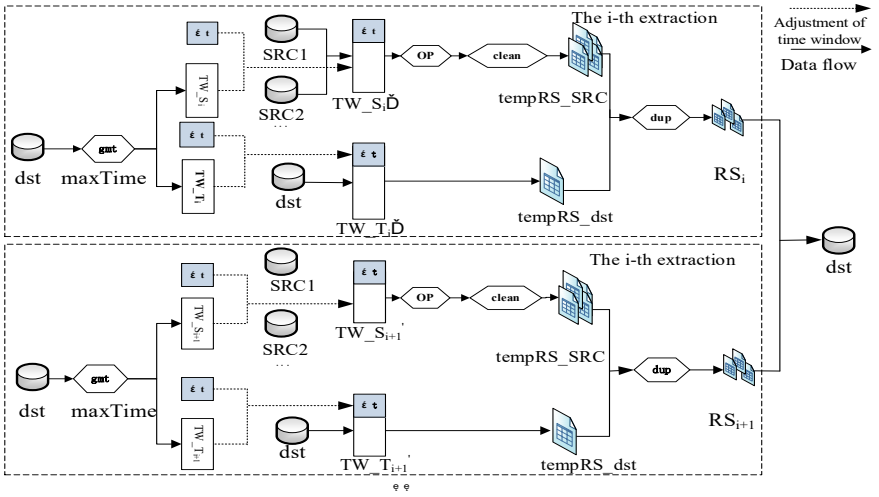


**Figure 4.** Incremental data extraction model diagram based on variable time windows

## 3.3 Data deduplication

Data deduplication is achieved through comparison operations on data, and the efficiency of data de-duplication is highly dependent on the number of comparisons between data, and trying to reduce the number of useless comparisons between data is an effective way to improve the efficiency of deduplication. Data deduplication needs to solve two key problems: reducing the search space and record matching [17].

The general improvement model de-duplication the data by comparing the new change data and the entire target table data one by one. This approach is inefficient and there are two main reasons:

Firstly, there is a large proportion of ineffective comparison operations. Secondly, the efficiency of data matching is too low. In general, the earlier the storage of data, the less likely to change in the future. Based on the above assumptions, we filter data of the target table according to the time of entry of data into the database. We only select the data from the target database for a period of time before the occurrence of abnormal data rather than the entire target table data to compare with the new data to narrow the search space. It can significantly reduce the number of comparisons between data.

## 4. Implementation

From the definition of VTWM, we can see that the table structure of time table described in 3.1 cannot meet the requirements of the new model. Therefore, this paper redesigns the middle table and the name is CDC_TIME. The table structure is shown in Table 1. *TS_S* and *TF* constitute a source time window *TW_S*. *TS_T* and *TF* constitute a target time window *TW_T*. And *pre_time* represents the data maintenance lead time $\Delta t$ and will not change once determined.

Table 1. Structure of CDC_TIME table

| Field | Type | Explain |
|---|---|---|
| table_name | varchar(30) | Primary key, the target table name for the current record maintenance |
| TS_S | datetime | The source system extracts the start time of the record |
| TS_T | datetime | The start time of the target database extraction of records |
| TF | datetime | The current time of the system when the extraction process is started |
| pre_time | long | Data maintenance pre - time, unit: Second |

The key for VTWM implementation is the maintenance of the time window. The maintenance algorithm for CDC_TIME table is shown in Algorithm 2.

Algorithm 2 Update algorithm for CDC_TIME table

Input: Record<TF, TS_S, TS_T, TABLE>, Timestamps information used to extract data from the source to the target table

Output: Record'<TF, TS_S, TS_T, TABLE>, Updated Record

```
1. begin
2.      if (Record equals null) then
3.              TS_S:=1970-01-01 12:00:00;
4.              TS_T:=sysDate;//system current time
5.      end if
6.      TF:=sysDate;
7.      start Job until finished ;//job: extraction of incremental data
8.      maxVal:=getMaxTime(TABLE);//the maximum value of the attribute column of the data update time from the target database table
9.      TS_S:=maxVal-pre_time;
10.     TS_T:=maxVal-pre_time;
11.     update Record;
12.     return Record;
13.end
```

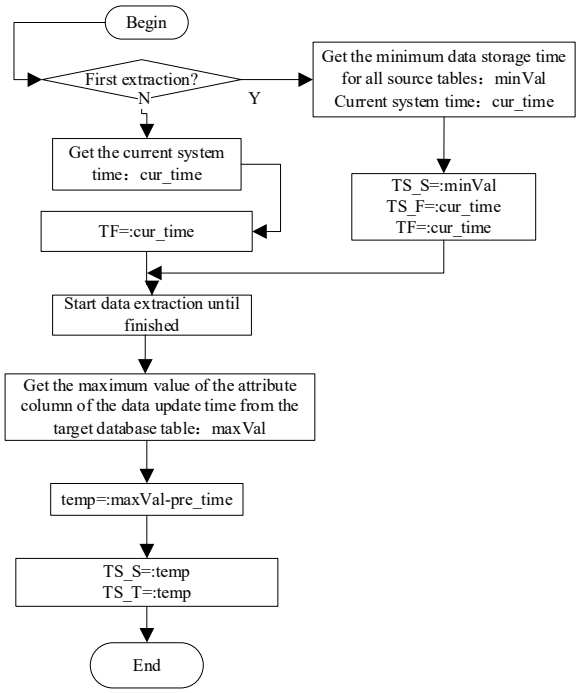


**Figure 5.** Processing flow of the update algorithm for CDC_Time table

For the first time extraction, all the data of all data sources needed to be extracted. Therefore, *TS_S* needed to be set as an earlier time for the first extraction. In this study, the initial value of *TS_T* was taken from 1970-01-01 12:00:00. In addition, the target database table was empty for first time extraction, so set the value of *TS_T* was the current system time. Except the first extraction, the value of *TS_S* and *TS_T* of each extracted data were the difference between the maximum value, *maxVal*, of the time of putting data into current target database table and the data maintenance lead time.

## 5. Experiment and Analysis

The traditional time-stamped incremental data capture model is deficient in reliability, which is improved accordingly in this paper. In order to verify the effectiveness of the design, this paper analyzes and compares the process shown in Figure 1 in terms of reliability and data extraction efficiency.

## 5.1 Experimental environment

This experiment was carried out in a 64-bit windows system. System version was windows 10 Professional Edition, and CPU was Intel i5-2400, quad-core, clocked at 3.10GHz. The memory was 4GB and the mechanical hard drive was 1TB.CPU, memory and hard disk free load were about 24%, 45% and 27% respectively. The source of the load was mainly generated by the operation of the system basic software. The database environment used in this experiment was Oracle11gRelease2 Standalone.

## 5.2 Comparison and analysis of reliability

In this experiment, we compared the data extraction of the Traditional Model, the General Improvement Model and VTWM. This experiment compared the three models from the exception of memory overflow and abnormal database connection. Table 2 is a comparison of the reliability of three models in the case of a memory overflow exception in the extraction process of 19880536 records in the A view. Table 3 is the reliability comparison of three models in the case of database connection exception in the data extraction process. The number of records to be extracted reduced to 3 million for avoiding the impact of memory overflow on the experimental results. Due to the amount of data to be extracted is small and there are very few anomalies in the extraction process. Therefore, the database connection abnormality is created by pulling out the network cable in the actual data extraction process.

Table 2. Reliability comparison under exception of out of memory

| Model | First Time Abnormal | Number of Exceptions | Target table data volume | Extraction results |
|---|---|---|---|---|
| traditional model | 7517000 | ∞ | 7517000 | fail |
| General model | 7528000 | ∞ | 7528000 | fail |
| Model of this paper | 7539000 | 2 | 19880536 | success |

Table 3. Reliability comparison under the exception of database connection

| Model | First Time Abnormal | Number of Exceptions | Target table data | Extraction results |
|---|---|---|---|---|
| traditional model | 1506000 | ∞ | 1506000 | fail |
| General model | 1428091 | 1 | 3000000 | success |
| Model of this paper | 1490000 | 1 | 3000000 | success |

In Table 2, the traditional model and the general improvement model cannot continue to extract data when a memory overflow exception occurs in the data extraction process. However, after the improved memory overflow exception in the data extraction process of the VTWM, the data extraction workflow can be started again to extract the data. In Table 3, the traditional model cannot continue to extract data after database connection anomaly occurs in the data extraction process. In the general improvement model and the VTWM, the data extraction process can be restarted after the database connection exception occurs. But the general improvement model takes a long time. Combining the above two points, the VTWM in this study is better than other two models in the case of anomaly.

## 5.3 Comparison and analysis of time performance

In order to further verify the performance of incremental timestamp-based data extraction model of variable window, this study compared the model in the data extraction time performance with the Traditional Model and the general improvement model. This model was based on reliability, so this experiment mainly compared the extraction efficiency of the three models in the process of data extraction in the case of abnormal circumstances. The experiment was carried out in two times. The first fixed target table data was 1 million and followed by an increase of the amount of source data. The second fixed source data was 1 million and followed by increasing the target table data volume. Due to the probability of abnormalities in the data extraction process was relatively small in reality, so this experiment used manual interference to create abnormalities. Two comparative experiments were performed abnormalities when the data amount of the target table and the source table were extracted to 450 thousand. The model that extracted data unsuccessfully due to abnormalities, the statistical time were calculated with the maximum. The comparison results were shown in Figure 6 and Figure 7.

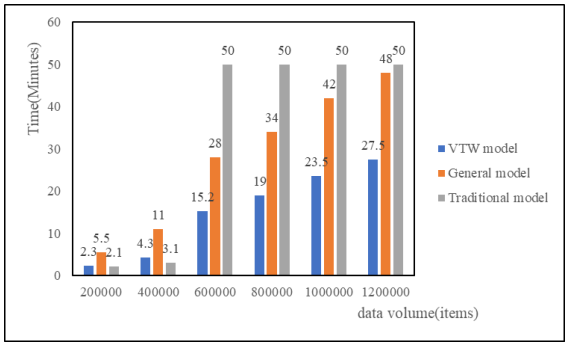


**Figure 6.** Performance comparison of three models in the case of fixed target table data

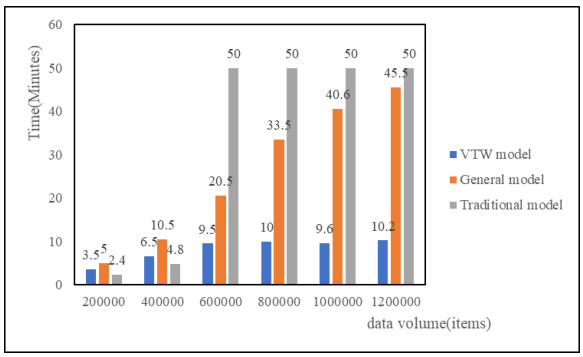


**Figure 7.** Performance comparison of three models in the case of fixed source table data

It can be seen from the comparative data in Figure 6 and Figure 7 that, although the extraction efficiency of the VTWM is lower than the traditional model if no abnormality occurs in the extraction process, the traditional model cannot continue to extract data when an abnormality occurs, while the general improvement model and the VTWM can continue to extract data and the efficiency of the VTWM is nearly doubled compared with the general improvement model. Comparing the extraction efficiency of the traditional model and the general improvement model, it is found that the main reason for the reduction of data extraction efficiency is the de-duplication operation. The VTWM is optimized on the basis of the general improvement model, and the amount of data comparison is reduced, thus improving the data extraction efficiency to a certain extent; meanwhile, as can be seen from the data in Figure 7, the VTWM avoids the problem that the extraction efficiency of the general improvement model gradually decreases as the amount of data in the target table increases.

## 6. Conclusion

In this paper, we propose VTWM, an incremental data extraction model based on variable time-windows. We study the problems occurred in the incremental data capture method based on timestamp change, and optimize the incremental timestamp-based data extraction method. Although, without anomalies, there are still gaps in the improvements of VTWM proposed in this paper when compared to the traditional extraction model, VTWM reduces the impact of anomalies on data extraction and improves the reliability of the incremental timestamp-based data capture method, and also takes into account the efficiency of data extraction under the premise of ensuring reliability.

The data maintenance variable time in VTWM proposed in this paper needs to be manually set currently, and there is no uniform standard for setting the value size. The size of the value has a certain impact on the efficiency of data extraction. This study hopes to make a personalized setting based on the frequency of change of the source table data in the future studies, and minimize the effect of this value on data extraction efficiency.

## References

[1] Mis Minakshi.HC Sharma. Near Real-Time Data Warehousing Using State-of-the-Art ETL Tools [J]. International Journal of Research (IJR) Vol-1, Issue-10 November 2014.

[2] SHU Qi. The Research on Optimization of ETL Process and Incremental Data Extraction [D]. Hunan University,2011.

[3] Jorg T, Desloch S. Towards generating ETL processes for incremental loading[C]. International Database Engineering and Applications Symposium, 2008: 101-110.

[4] 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.

[5] Jia Yankai. Research and Design on Data Extraction in Multiple Data Sources [D]. Harbin Engineering University,2013.

[6] Wen Lu. Design and Implementation of Incremental Data Extractor based on Sector Inquiry[D]. Hebei University of Science and Technology,2015.

[7] XU Fuliang , ZHOU Zude.Research on Change-Data-Capture Technology[J]. Journal of WUT(Information & Management Engineering),2009, 05:740-743.

[8] DAI Hao,YANG Bo.Researches on mechanics of incremental data extraction in ETL[J].Computer Engineering and Design, 2009,(23):5552-5555.

[9] WANG Yu-biao, RAO Xi-ru, HE Pan.Incremental database synchronization update mechanism under heterogeneous environment [J].Computer Engineering and Design,2011,03: 948-951.

[10] YANG Le. Design and Implementation of Real-time data extraction Mechanism in data warehousing [D]. Beijing University of Posts and Telecommuni- cations,2007.

[11] [TAN Guang-wei,WU Tong. Study on Method of Data Warehouse Real-time Data Updating Based on Mechanism of CDC [J].Computer Science, 2015, 42(s1).

[12] ZOU Xian-xia , JIA Wei-jia,PAN Jiu-hui. Research of Log-based Change Data Capture [J]. Journal of Chinese Computer Systems, 2012, 33(3):531-536.

[13] MattCasters, RolandBouman, JosvanDongen,etl. Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration [M]. Publishing House of Electronics Industry, 2014.

[14] CUI You-wen,ZHOU jin-hai. Research on Data Integration Based on KETTLE [J]. Computer Technology and Development,2015,(04):153-157.

[15] LIU Xueqiong,WU Gang,DENG Houping.Data deduplication in Web information integration[J]. Journal of Computer Applications,2013, (09):2493-2496.

[16] Ari Wibisono,Wisnu Jatmiko,Hanief Arief Wisesa,Benny Hardjono,Petrus Mursanto. Traffic big data prediction and visualization using Fast Incremental Model Trees-Drift Detection (FIMT-DD)[J]. Knowledge-Based Systems,2016,93.

[17] Annie Anak Joseph,Takaomi Tokumoto,Seiichi Ozawa. Online feature extraction based on accelerated kernel principal component analysis for data stream[J]. Evolving Systems,2016,7(1).

[18] Haixiang Li,Zhanhao Zhao,Yijian Cheng,Wei Lu,Xiaoyong Du,Anqun Pan. Efficient time-interval data extraction in MVCC-based RDBMS[J]. World Wide Web,2019,22(6).
[19] Chao Tan,Genlin Ji. Semi-supervised incremental feature extraction algorithm for large-scale data stream[J]. Concurrency and Computation: Practice and Experience,2017,29(6).