

DATA STORAGE METHOD, AUTOMATIC SYSTEM, AND MECHANICAL EQUIPMENT

TECHNICAL FIELD

5 This application relates to the technical field of automatic mechanical equipment, specifically to a data storage method, an automatic system, and mechanical equipment.

BACKGROUND

10 With the continuous use of automatic wharves, automatic control systems have gradually become an important technology in the port industry. The automatic control systems not only ensure drivers to safely and comfortably operate automatic equipment in remote control rooms, reduce the work intensity, and improve the work environment, but also improve the production operation efficiency of ports, and reduce the operation costs.

SUMMARY

15 In view of this, this application provides a data storage method, an automatic system, and mechanical equipment, which can provide a reliable data storage technology for an automatic wharf.

20 In a first aspect, a data storage method provided by this application includes: obtaining work data; establishing a message queue, and sequentially adding the work data into the message queue; determining whether a storage engine uses a sequence strategy or a thread strategy based on a data volume of the work data; when the storage engine uses the sequence strategy, sequentially extracting the work data from the message queue in a single-thread mode based on an adding sequence of the work data added to the message queue; when the
25 storage engine uses the thread strategy, extracting the work data from the message queue in a multi-thread mode; and performing data cleaning on the work data extracted from the message queue, and storing normal data obtained by cleaning in a database.

 In this aspect, during use, the appropriate number of threads can be selected based on

the data volume to extract the work data from the message queue, so as to implement effective control on computing resources. A single thread is used in case of a low data volume to avoid a waste of computing power; and multiple threads is used at data peak with a large data volume to ensure the data processing speed. The work data is extracted and then
5 cleaned, so as to ensure that the normal data is stored in the database.

In conjunction with the first aspect, in a possible embodiment, the performing data cleaning on the work data extracted from the message queue includes: screening the work data to obtain abnormal data; and storing the abnormal data in the database or an abnormal log; where the abnormal data includes at least one type of wrong data, unknown data, expired
10 data, duplicate data, unordered data, and redundant data.

In conjunction with the first aspect, in a possible embodiment, when the abnormal data includes the wrong data, the screening the work data to obtain abnormal data includes: performing message parsing on the work data; and when the parsing fails, determining that the work data is the wrong data.

15 In conjunction with the first aspect, in a possible embodiment, when the abnormal data includes the wrong data and the unknown data, the screening the work data to obtain abnormal data includes: performing message parsing on the work data; when the parsing fails, determining that the work data is the wrong data; when the parsing succeeds, determining whether the work data meets a preset acquisition requirement; and when the work data does
20 not meet the acquisition requirement, determining that the work data is the unknown data.

In conjunction with the first aspect, in a possible embodiment, when the abnormal data includes the wrong data, the unknown data, and the expired data, the screening the work data to obtain abnormal data includes: performing message parsing on the work data; when the parsing fails, determining that the work data is the wrong data; when the parsing succeeds,
25 determining whether the work data meets a preset acquisition requirement; when the work data does not meet the acquisition requirement, determining that the work data is the unknown data; when the work data meets the acquisition requirement, determining whether an acquisition time point for acquiring the work data is within a preset time range; and when the acquisition time point is not within the preset time range, determining that the work data

is the expired data.

In conjunction with the first aspect, in a possible embodiment, when the abnormal data includes the wrong data, the unknown data, the expired data, and the unordered data, the screening the work data to obtain abnormal data includes: performing message parsing on the work data; when the parsing fails, determining that the work data is the wrong data; when the parsing succeeds, determining whether the work data meets a preset acquisition requirement; when the work data does not meet the acquisition requirement, determining that the work data is the unknown data; when the work data meets the acquisition requirement, determining whether an acquisition time point for acquiring the work data is within a preset time range; when the acquisition time point is not within the preset time range, determining that the work data is the expired data; when the acquisition time point is within the preset time range and a current strategy is the sequence strategy, determining whether the current work data and the previous work data acquired before the current work data conform to the adding sequence; and when the current work data and the previous work data do not conform to the adding sequence, determining that the work data is the unordered data.

In conjunction with the first aspect, in a possible embodiment, when the abnormal data includes the wrong data, the unknown data, the expired data, the unordered data, and the redundant data, the screening the work data to obtain abnormal data includes: performing message parsing on the work data; when the parsing fails, determining that the work data is the wrong data; when the parsing succeeds, determining whether the work data meets a preset acquisition requirement; when the work data does not meet the acquisition requirement, determining that the work data is the unknown data; when the work data meets the acquisition requirement, determining whether an acquisition time point for acquiring the work data is within a preset time range; when the acquisition time point is not within the preset time range, determining that the work data is the expired data; when the acquisition time point is within the preset time range and a current strategy is the sequence strategy, determining whether the current work data and the previous work data acquired before the current work data conform to the adding sequence; when the current work data and the previous work data do not conform to the adding sequence, determining that the work data is

the unordered data; when the current work data and the previous work data conform to the adding sequence, determining whether the current work data is the same as the previous work data; and when the current work data is the same as the previous work data, determining that the work data is the redundant data.

5 In conjunction with the first aspect, in a possible embodiment, the method further includes: when the current work data is not the same as the previous work data, determining that the work data is the normal data.

10 In conjunction with the first aspect, in a possible embodiment, the method further includes: when the storage engine uses the thread strategy, counting the number of threads; obtaining the number of messages of the work data extracted from the message queue; obtaining a data volume of the normal data in the number of messages of the work data; and obtaining a data volume of the abnormal data in the number of messages of the work data.

15 In conjunction with the first aspect, in a possible embodiment, when the abnormal data includes the duplicate data, the screening the work data to obtain abnormal data includes: determining whether the work data violates a uniqueness constraint; and when the work data violates the uniqueness constraint, determining that the work data is the duplicate data.

20 In conjunction with the first aspect, in a possible embodiment, the method further includes: determining whether the work data has a preset error; and when the work data has the preset error, determining that the work data is the wrong data.

25 In conjunction with the first aspect, in a possible embodiment, the method further includes: determining that the storage engine uses a Highest security strategy, a Normal security strategy, or a Lowest security strategy based on input information.

30 In conjunction with the first aspect, in a possible embodiment, in the process of sequentially extracting the work data from the message queue, the method further includes: when the work data in the message queue is non-multicast data, setting a data volume upper limit of the work data extracted from the message queue; extracting the work data from the message queue based on the data volume upper limit; and when the work data in the message queue is multicast data, splitting the multicast data into the non-multicast data, and extracting the work data based on the data volume upper limit.

In conjunction with the first aspect, in a possible embodiment, in the process of sequentially extracting the work data from the message queue, the method further includes: setting a timeout limit for extracting the work data from the message queue; and when there is no data in the message queue and time for extracting the work data does not exceed the
5 timeout limit, waiting for the work data to be added into the message queue.

In conjunction with the first aspect, in a possible embodiment, in the process of sequentially extracting the work data from the message queue, the method further includes: setting a timeout limit for extracting the work data from the message queue; and when there is no data in the message queue and the time for extracting the work data exceeds the timeout
10 limit, stopping extracting the work data from the message queue.

In conjunction with the first aspect, in a possible embodiment, in the process of sequentially extracting the work data from the message queue, the method further includes: determining whether the message queue and the database are online and down; when the message queue and the database are down for the first time, performing data cleaning on the
15 extracted work data, saving breakpoint data of the work data extracted from the message queue, and switching the storage engine to use the thread strategy; and when the message queue and the database are online for the first time after being down, resuming performing the operation of extracting the work data from the message queue based on the breakpoint data, and switching the storage engine to use the thread strategy.

20 In conjunction with the first aspect, in a possible embodiment, in the process of sequentially extracting the work data from the message queue, the method further includes: controlling the storage engine and the message queue to be regularly disconnected from and reconnected to each other, and controlling the storage engine and the database to be regularly disconnected from and reconnected to each other; when the storage engine and the message
25 queue are disconnected from each other, controlling the storage engine and the message queue to be immediately disconnected from and reconnected to each other; when the storage engine and the database are disconnected from each other, controlling the storage engine and the database to be immediately disconnected from and reconnected to each other; and when the storage engine and the message queue are successfully disconnected from and

reconnected to each other, and the storage engine and the database are successfully disconnected from and reconnected to each other, switching the storage engine to use the thread strategy.

5 In conjunction with the first aspect, in a possible embodiment, the method further includes: determining that the storage engine uses an immediate stop strategy based on the input information; and terminating and destroying a thread when stopping the extraction of the work data.

10 In conjunction with the first aspect, in a possible embodiment, the method further includes: determining that the storage engine uses a delayed stop strategy based on the input information; performing data cleaning on the extracted work data when stopping the extraction of the work data; and terminating and destroying the thread after the data cleaning is completed.

15 In a second aspect, this application provides an automatic system, including: a data obtaining module configured to obtain work data; a message queue establishment module communicatively connected to the data obtaining module and configured to establish a message queue and sequentially add the work data into the message queue; a strategy determination module communicatively connected to the data obtaining module and configured to determine whether a storage engine uses a sequence strategy or a thread strategy based on a data volume of the work data; a data access module communicatively
20 connected to the message queue establishment module and the strategy determination module separately and configured to: when the storage engine uses the sequence strategy, sequentially extract the work data from the message queue in a single-thread mode based on an adding sequence of the work data added to the message queue; and when the storage engine uses the thread strategy, extract the work data from the message queue in a
25 multi-thread mode; and a data cleaning module communicatively connected to the data access module and configured to perform data cleaning on the work data extracted from the message queue and store normal data obtained by cleaning in a database.

The second aspect refers to an apparatus corresponding to the first aspect, so the technical effects of the second aspect will not be repeated herein.

In a third aspect, this application provides mechanical equipment, including a memory and a processor, where the memory has an executable code stored therein, and the processor is configured to execute the executable code, so as to implement the data storage method as described in the first aspect.

5 The third aspect contains the data storage method described in the first aspect, so the technical effects of the third aspect will not be repeated herein.

BRIEF DESCRIPTION OF DRAWINGS

10 FIG. 1 shows a schematic flowchart of a data storage method provided by an embodiment of this application;

FIG. 2 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

FIG. 3 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

15 FIG. 4 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

FIG. 5 shows a schematic diagram of steps of a data storage method provided by another embodiment of this application;

20 FIG. 6 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

FIG. 7 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

FIG. 8 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

25 FIG. 9 shows a schematic diagram of steps of a data storage method provided by another embodiment of this application;

FIG. 10 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

FIG. 11 shows a schematic flowchart of a data storage method provided by another

embodiment of this application;

FIG. 12 shows a schematic flowchart of a data storage method provided by another embodiment of this application;

FIG. 13 shows a schematic diagram of a whole processing flow for work data provided
5 by another embodiment of this application;

FIG. 14 shows a schematic diagram of task processing in a program form provided by an embodiment of this application;

FIG. 15 shows a schematic structural diagram of an automatic system provided by an embodiment of this application; and

10 FIG. 16 shows a schematic structural diagram of mechanical equipment provided by an embodiment of this application.

DETAILED DESCRIPTION OF EMBODIMENTS

The technical solutions in the embodiments of this application are clearly and
15 completely described below with reference to the accompanying drawings in the embodiments of this application. Apparently, the described embodiments are merely some rather than all of the embodiments of this application. All other embodiments obtained by those of ordinary skill in the art based on the embodiments of this application without creative efforts shall fall within the scope of protection of this application.

20 The foundation of an automatic wharf is that more and more intelligent mechanical equipment is applied to the wharves, and a large amount of data is generated in operation of the mechanical equipment, which is also accompanied with data loss, data peak, high concurrency, slow storage, and other issues. Therefore, how to provide a highly reliable data storage technology for the automatic wharf, ensure normal operation of the automatic wharf,
25 and protect value of data assets has become a necessary and important basic technology.

Exemplary data storage method

FIG. 1 shows a schematic flowchart of a data storage method provided by an embodiment of this application. This application provides a data storage method. The method may be applied to mechanical equipment with a data processing capability. As

shown in FIG. 1, the method includes the following steps.

In step 110, work data is obtained.

5 In this embodiment, the work data may be various data generated by the mechanical equipment in operation, and may be acquired by a programmable logic controller of the mechanical equipment and then obtained from the programmable logic controller of the mechanical equipment. A specific source of obtaining the work data is not limited in this embodiment of the present application. The source of obtaining the work data varies depending on different application scenarios.

10 In step 120, a message queue is established, and the work data is sequentially added into the message queue.

In step 130, it is determined whether a storage engine uses a sequence strategy or a thread strategy based on a data volume of the work data. In this embodiment, a correspondence between the data volume and the sequence strategy or the thread strategy is preset. For example, it is set that the sequence strategy is used for the data volume within a range of 0 to x, and the thread strategy is used for the data volume within a range of x to y.

In step 140, when the storage engine uses the sequence strategy, the work data is sequentially extracted from the message queue in a single-thread mode based on an adding sequence of the work data added to the message queue.

20 In this embodiment, when the data volume is small, a main thread and a slave thread may be established. The work data is extracted from the message queue by the slave thread. During this period, the work data is sequentially obtained based on the adding sequence of the work data. Due to the use of such sequence strategy, the data volume processed once is small, and only one slave thread may be established, so as to avoid a waste of computing power.

25 In step 150, when the storage engine uses the thread strategy, the work data is extracted from the message queue in a multi-thread mode.

In this embodiment, when the data volume is too large, the thread strategy may be used, that is, one main thread and multiple slave threads are established, where the multiple slave threads are managed by the main thread. A correspondence between the data volume and the

number of slave threads is preset, such that a corresponding number of slave threads are established to ensure a sufficient number of slave threads. The multiple slave threads work in parallel to extract the work data from the message queue, thereby ensuring the data processing speed and improving the data processing efficiency.

5 After step 140 or step 150, step 160 is performed, in which data cleaning is performed on the work data extracted from the message queue, and normal data obtained by cleaning is stored in a database.

In this embodiment, data cleaning may be performed on the work data extracted by the slave thread, which can avoid storing abnormal data in the database.

10 In this embodiment, the appropriate number of threads can be selected based on the data volume to extract the work data from the message queue, so as to implement effective control on computing resources. A single thread is used in case of a low data volume to avoid a waste of computing power; and multiple threads is used at data peak with a large data volume to ensure the data processing speed. The work data is extracted and then cleaned, so
15 as to ensure that the normal data is stored in the database.

FIG. 2 shows a schematic flowchart of a data storage method provided by another embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, in the process of data cleaning, as shown in FIG. 2, step 160 includes:

20 step 1601: screening the work data to obtain abnormal data; and
step 1602: storing the abnormal data in the database or an abnormal log.

In this embodiment, the abnormal data includes at least one type of wrong data, unknown data, expired data, duplicate data, unordered data, and redundant data.

In this embodiment, multiple slave threads may be established, and data cleaning is
25 performed on the work data by using the multiple threads, so as to extract the abnormal data and store the abnormal data in the database or the abnormal log. Later, the abnormal data may be called from the database or the abnormal log at any time for analysis.

FIG. 3 shows a schematic flowchart of a data storage method provided by another embodiment of this application. The method may be applied to mechanical equipment with a

data processing capability. In an embodiment, in the process of data cleaning, when the abnormal data includes the wrong data, the unknown data, the expired data, the unordered data, and the redundant data, as shown in FIG. 3, step 1601 includes the following operations.

5 In step 16011, message parsing is performed on the work data by a parser.

In this embodiment, the parser is established, where the parser is managed by the slave thread. The parser performs message parsing on the work data.

When the parsing fails, it is determined that the work data is the wrong data. When the parsing succeeds, step 16012 is performed, in which it is determined whether the work data
10 meets a preset acquisition requirement by a filter.

In this embodiment, the filter is established, and it is determined whether the work data meets the preset acquisition requirement by the filter.

In this embodiment, the acquisition requirement is preset for the mechanical equipment requiring acquisition and/or several measuring points on the mechanical
15 equipment requiring acquisition. When it is determined that the current work data is data corresponding to the mechanical equipment requiring acquisition, the work data meets the acquisition requirement; and/or when it is determined that the current work data is the measuring points on the mechanical equipment requiring acquisition, the work data meets the acquisition requirement.

20 When a performing result of step 16012 is that the work data does not meet the preset acquisition requirement, it is determined that the work data is the unknown data. When the performing result of step 16012 is that the work data meets the preset acquisition requirement, step 16013 is performed, in which it is determined whether an acquisition time point for acquiring the work data is within a preset time range.

25 In this embodiment, it is determined whether the acquisition time point for acquiring the work data is within the preset time range by the filter. The acquisition time point for acquiring the work data may be obtained from the mechanical equipment. The acquisition time point refers to when the work data is acquired. The preset time range may be set to, for example, within 1 year or 2 years. When the acquisition time point is not within the preset

time range, it indicates that the work data is relatively old and belongs to the expired data.

When a performing result of step 16013 is that the acquisition time point for acquiring the work data is not within the preset time range, it is determined that the work data is the expired data. When the determination result of step 16013 is that the acquisition time point
5 for acquiring the work data is within the preset time range, and a current strategy is the sequence strategy, step 16014 is performed, in which it is determined whether the current work data and the previous work data acquired before the current work data conform to the adding sequence.

In this embodiment, it is determined whether the current work data and the previous
10 work data conform to the adding sequence by the filter. For example, it is determined whether the currently processed work data is after the previously processed work data based on the adding sequence. When the thread strategy is used currently, step 16014 needs to be skipped.

When a determination result of step 16014 is that the current work data and the
15 previous work data acquired before the current work data do not conform to the adding sequence, it is determined that the work data is the unordered data.

When the determination result of step 16014 is that the current work data and the previous work data acquired before the current work data conform to the adding sequence, step 16015 is performed, in which it is determined whether the current work data is the same
20 as the previous work data.

In this embodiment, it is determined whether the current work data is the same as the previous work data by the filter. When a determination result of step 16015 is that the current work data is the same as the previous work data, it is determined that the work data is the redundant data. When the determination result of step 16015 is that the current work data is
25 not the same as the previous work data, it is determined that the work data is the normal data.

In addition, when the abnormal data includes the duplicate data, it may also be determined whether the work data is the duplicate data by the following determination process. It is determined whether the work data violates a uniqueness constraint. When the work data violates the uniqueness constraint, it is determined that the work data is the

duplicate data.

In this embodiment, it is determined whether the work data violates the uniqueness constraint by the filter to further screen the work data, so as to screen out the duplicate data. Specifically, a uniqueness constraint determination algorithm may be used to search for
5 whether the work data violating the uniqueness constraint exists, where the work data violating the uniqueness constraint refers to the duplicate work data.

In this embodiment, the abnormal data may be effectively screened out by using the above multiple determination processes.

However, it is to be noted that the number of determination processes varies depending
10 on different data types included in the work data. For example, when the work data includes only the wrong data, only one determination process is included, that is, message parsing is performed on the work data; and when the parsing fails, it is determined that the work data is the wrong data. For another example, when the work data includes the wrong data and the unknown data, two determination processes are included, that is, message parsing is
15 performed on the work data; when the parsing fails, it is determined that the work data is the wrong data; when the parsing succeeds, it is determined whether the work data meets the preset acquisition requirement; and when the work data does not meet the acquisition requirement, it is determined that the work data is the unknown data. For another example, when the work data includes the wrong data, the unknown data, and the expired data, three
20 determination processes are included, that is, message parsing is performed on the work data; when the parsing fails, it is determined that the work data is the wrong data; when the parsing succeeds, it is determined whether the work data meets the preset acquisition requirement; when the work data does not meet the acquisition requirement, it is determined that the work data is the unknown data; when the work data meets the acquisition
25 requirement, it is determined whether the acquisition time point for acquiring the work data is within the preset time range; and when the acquisition time point is not within the preset time range, it is determined that the work data is the expired data. By analogy, the above multiple determination processes are obtained.

FIG. 4 shows a schematic flowchart of a data storage method provided by another

embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, as shown in FIG. 4, the data storage method further includes the following steps.

In step 1603, it is determined whether the work data violates a uniqueness constraint.

5 When a determination result of step 1603 is that the work data violates the uniqueness constraint, it is determined that the work data is the duplicate data. When the determination result of step 1603 is that the work data does not violate the uniqueness constraint, step 1604 is performed, in which it is determined that the work data has a preset error. When a determination result of step 1604 is that the work data has the preset error, it is determined
10 that the work data is the wrong data.

 This embodiment may be performed by the filter to further screen the work data, so as to screen out the duplicate data and the wrong data. Specifically, a uniqueness constraint determination algorithm may be used to search for whether the work data violating the uniqueness constraint exists, where the work data violating the uniqueness constraint refers
15 to the duplicate work data. In step 1604, determination criteria for wrong data are preset, for example, it is screened whether the work data has at least one of multiple types of errors such as a format error, a network error, and a code error. The determination criteria may be specifically set as needed.

 However, a precondition for performing step 1604 is not specifically limited in this
20 embodiment of the present application. Step 1604 may be performed only when the determination result of step 1604 is that the work data has the preset error, or step 1604 may be performed directly, that is, it is directly determined whether the work data has the preset error.

 FIG. 5 shows a schematic flowchart of a data storage method provided by another
25 embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, as shown in FIG. 5, the data storage method further includes the following steps.

 In step 200, it is determined that the storage engine uses a Highest security strategy, a Normal security strategy, or a Lowest security strategy based on input information.

Step S200 may be performed before step 110, which is selected by a person or a program. The three security strategies may implement balance between the data security and the storage speed, and can cope with special scenarios that the process is killed, the program immediately stops, and the database or the message queue is down. In a Highest security mode, the data security is preferentially ensured, but the running speed in step 110 to step 160 is reduced, that is, less computing power is allocated to step 110 to step 160. In a Normal security mode, the data security and the running speed will be balanced, and the abnormal data may be lost when the process is killed and the program immediately stops, but the normal data will not be lost. In a Lowest security mode, the storage speed is preferred to the data security, that is, the storage speed of storing the normal data in the database in step 160 is increased. The Highest security strategy, the Normal security strategy, or the Lowest security strategy is a common data storage technology, so their working principles will not be repeated herein.

FIG. 6 shows a schematic flowchart of a data storage method provided by another embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, as shown in FIG. 6, the data storage method further includes:

step 210: when the storage engine uses the thread strategy, counting the number of threads;

step 220: obtaining the number of messages of the work data extracted from the message queue;

step 230: obtaining a data volume of the normal data in the number of messages of the work data; and

step 240: obtaining a data volume of the abnormal data in the number of messages of the work data.

In this embodiment, the multiple pieces of data above may be counted to obtain current thread resource usage, the number of data messages, and the data volume, and the counted data may be used to regulate and control computing resources.

FIG. 7 shows a schematic flowchart of a data storage method provided by another

embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, when the thread strategy is used, as shown in FIG. 7, the data storage method further includes:

step 250: obtaining the number of active slave threads, the number of idle slave threads,
5 the number of created slave threads, and the number of destroyed slave threads; and

step 260: creating or destroying the slave threads based on the number of the active slave threads, the number of the idle slave threads, the number of the created slave threads, the number of the destroyed slave threads, and the data volume of the work data.

In this embodiment, the numbers of various types of slave threads may be obtained in
10 real time to create or destroy the slave threads based on the current actual situation, such that the slave threads are regulated and controlled to match with the current thread situation and the data volume of the work data, thereby implementing the best management and control on the computing power.

FIG. 8 shows a schematic flowchart of a data storage method provided by another
15 embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, under the sequence strategy, in the process of sequentially extracting the work data from the message queue, as shown in FIG. 8, the data storage method further includes:

step 310: screening whether the work data is non-multicast data or multicast data;
20 step 320: when the work data in the message queue is the non-multicast data, setting a data volume upper limit of the work data extracted from the message queue;

step 330: extracting the work data from the message queue based on the data volume upper limit; and

step 340: when the work data in the message queue is the multicast data, splitting the
25 multicast data into the non-multicast data, and extracting the work data based on the data volume upper limit.

In this embodiment, it is determined whether the work data is split based on the type of the work data, and when the work data is the non-multicast data, the data volume upper limit of the extracted work data is set, so as to avoid data jamming or even downtime caused by a

large amount of data extracted once.

FIG. 9 shows a schematic flowchart of a data storage method provided by another embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, under the sequence strategy, in the process of sequentially extracting the work data from the message queue, as shown in FIG. 9, the data storage method further includes:

step 350: setting a timeout limit for extracting the work data from the message queue;

step 360: when there is no data in the message queue and time for extracting the work data does not exceed the timeout limit, waiting for the work data to be added into the message queue; and

step 370: when there is no data in the message queue and the time for extracting the work data exceeds the timeout limit, stopping extracting the work data from the message queue.

In this embodiment, the timeout limit for extracting the work data is set, and when there is no data in the message queue and the extraction time does not exceed the timeout limit, it indicates that the current computing power occupation is within an acceptable range, the work data may be waited to be added into the message queue, and data extraction is performed after the work data is added into the message queue. When there is no data in the message queue and the extraction time exceeds the timeout limit, the computing power is managed and controlled, the extraction of the work data is stopped, and data extraction is not performed even if subsequent work data is added into the message queue. The program corresponding to this method enters a disconnection state. In this case, a disconnection and reconnection strategy is needed to resume performing data extraction.

FIG. 10 shows a schematic flowchart of a data storage method provided by another embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, under the sequence strategy, in the process of sequentially extracting the work data from the message queue, as shown in FIG. 10, the data storage method further includes the following steps.

In step 410, it is determined whether the message queue and the database are online

and down.

In step 420, when the message queue and the database are down for the first time, data cleaning is performed on the extracted work data, breakpoint data of the work data extracted from the message queue is saved, and the storage engine is switched to using the thread
5 strategy.

In this embodiment, after the message queue and the database are down for the first time, the slave thread performs data cleaning on the currently extracted work data and then enters a rest state, so as to avoid resource waste.

In step 430, when the message queue and the database are online for the first time after
10 being down, the operation of extracting the work data from the message queue is resumed to be performed based on the breakpoint data, and the storage engine is switched to using the thread strategy.

In this embodiment, when the message queue and the database are online for the first time after being down, the slave thread resumes performing the extraction of the work data
15 based on the breakpoint data and switches the current strategy to the thread strategy. Because there may be a large amount of data and accumulated data that need to be processed after disconnection and reconnection, switching to the thread strategy using multiple threads can speed up data processing.

FIG. 11 shows a schematic flowchart of a data storage method provided by another
20 embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, as shown in FIG. 11, the data storage method further includes the following steps.

In step 440, the storage engine is controlled to be regularly disconnected from and reconnected to the message queue and the database separately.

25 It is determined whether the storage engine and the message queue are disconnected from each other, and when the storage engine and the message queue are disconnected from each other, step 450 is performed, in which the storage engine and the message queue are controlled to be immediately disconnected from and reconnected to each other.

It is determined whether the storage engine and the database are disconnected from

each other, and when the storage engine and the database are disconnected from each other, step 460 is performed, in which the storage engine and the database are controlled to be immediately disconnected from and reconnected to each other.

5 In step 470, when the storage engine is successfully disconnected from and reconnected to the message queue, and the storage engine is successfully disconnected from and reconnected to the database, the storage engine is switched to using the thread strategy.

10 In this embodiment, the disconnection and reconnection strategy is executed between the storage engine and the message queue and between the storage engine and the database separately. The regular execution of disconnection and reconnection can ensure that the storage engine automatically keeps connection with the message queue and the database separately for a long term. When disconnected from either the message queue or the database, the storage engine can be immediately disconnected therefrom and reconnected thereto, so as to ensure data connection. Because there may be a large amount of data and accumulated data that need to be processed after disconnection and reconnection, switching to the thread strategy using multiple threads can speed up data processing.

FIG. 12 shows a schematic flowchart of a data storage method provided by another embodiment of this application. The method may be applied to mechanical equipment with a data processing capability. In an embodiment, as shown in FIG. 12, the data storage method further includes:

20 step 510: determining whether the storage engine uses an immediate stop strategy or a delayed stop strategy based on the input information;

step 520: when the storage engine uses the immediate stop strategy, terminating and destroying a thread when stopping the extraction of the work data;

25 step 530: when the storage engine uses the delayed stop strategy, performing data cleaning on the extracted work data when stopping the extraction of the work data; and

step 540 performed after step 530, which is: terminating and destroying the thread after the data cleaning is completed.

In this embodiment, the immediate stop strategy or the delayed stop strategy is selected by the person or the program. When the immediate stop strategy is used and the

process of extracting the work data is closed by the person or the program, the thread is immediately stopped and destroyed to quickly end the program corresponding to the method, thereby achieving the effect of quickly closing the program. When the delayed stop strategy is used and the process of extracting the work data is closed by the person or the program,
5 the extracted work data is cleaned, and then the thread is stopped and destroyed, so as to avoid loss of the work data.

FIG. 13 shows a schematic diagram of a whole processing flow for work data provided by another embodiment of this application. As shown in FIG. 13, the programmable logic controller of the mechanical equipment acquires the work data from the mechanical
10 equipment; then, an acquisition engine obtains the work data and adds the work data into the message queue; next, the storage engine extracts the work data from the message queue and stores the work data in the database; and the database may store the work data in a storage medium.

FIG. 14 shows a schematic diagram of task processing in a program form provided by
15 an embodiment of this application. As shown in FIG. 13, a main thread and a slave thread pool managed by the main thread are established, where the slave thread pool includes one or more slave threads. After the program in the main thread is started and run, a configuration is loaded, such that a controller of the storage engine obtains the mechanical equipment requiring acquisition this time, measuring point information, and other configuration
20 information. The controller executes a stop strategy to terminate the operation of this program. The stop strategy may be the method of step 510 to step 540 described above.

A disconnection and reconnection processor is established in the main thread. The controller may regularly execute the disconnection and reconnection strategy to control the disconnection and reconnection processor to regularly perform disconnection and
25 reconnection. When the storage engine is disconnected from the message queue and the database, the disconnection and reconnection processor regularly performs disconnection and reconnection, such that the storage engine is reconnected to the message queue and the database. Meanwhile, the disconnection and reconnection processor will also regularly execute the thread strategy, such that automatic switching to the thread strategy is

implemented during disconnection and reconnection.

A counter is further established in the main thread to: count the number of active slave threads, the number of idle slave threads, the number of created slave threads, and the number of destroyed slave threads; count the number of processed messages, the volume of the normal data stored in the database, the volume of the unknown data, the volume of the expired data, the volume of the wrong data, and the volume of the duplicate data; update the number of messages to be processed and the number of messages being processed; and count other various data that needs to be counted.

A queue data iterator is established in the slave thread pool. The queue data iterator is configured to extract the work data from the message queue. When the message queue is down, the disconnection and reconnection processor immediately executes the disconnection and reconnection strategy to reestablish connection between the queue data iterator and the message queue. When the database is down, the disconnection and reconnection processor regularly executes the disconnection and reconnection strategy to reestablish connection between a memory and the database. When the queue data iterator works, a batch strategy will also be executed. The batch strategy may be the method of step 310 to step 370 described above. The queue data iterator extracts the work data and then sends the extracted work data to the parser for parsing, and the filter performs data cleaning after parsing. The memory stores the normal data obtained by cleaning in the database and executes an abnormal data strategy to store the abnormal data in an abnormal log processor. In addition, the filter will also perform cleaning to obtain the duplicate data and other wrong data.

Exemplary automatic system

This application further provides an automatic system. As shown in FIG. 15, the system includes a data obtaining module 1510, a message queue establishment module 1511, a strategy determination module 1512, a data access module 1513, and a data cleaning module 1514. The data obtaining module 1510 is configured to obtain work data. The message queue establishment module 1511 is communicatively connected to the data obtaining module 1510 and configured to establish a message queue and sequentially add the work data into the message queue. The strategy determination module 1512 is

communicatively connected to the data obtaining module 1510 and configured to determine whether a storage engine uses a sequence strategy or a thread strategy based on a data volume of the work data. The data access module 1513 is communicatively connected to the message queue establishment module 1511 and the strategy determination module 1512
5 separately and configured to: when the storage engine uses the sequence strategy, sequentially extract the work data from the message queue in a single-thread mode based on an adding sequence of the work data added to the message queue; and when the storage engine uses the thread strategy, extract the work data from the message queue in a multi-thread mode. The data cleaning module 1514 is communicatively connected to the data
10 access module 1513 and configured to perform data cleaning on the work data extracted from the message queue and store normal data obtained by cleaning in a database.

Exemplary mechanical equipment and computer-readable storage medium

Mechanical equipment according to an embodiment of this application is described with reference to FIG. 16 below. FIG. 16 shows a schematic structural diagram of
15 mechanical equipment provided by an embodiment of this application.

As shown in FIG. 16, the mechanical equipment 150 includes one or more processors 1501 and a memory 1502.

The processor 1501 may be a central processing unit (CPU) or any other form of processing unit with a data processing capability and/or an instruction execution capability,
20 and may control other components in the mechanical equipment 150 to perform desired functions.

The memory 1502 may include one or more computer program products which may include various forms of computer-readable storage media, such as a volatile memory and/or a non-volatile memory. The volatile memory may include, for example, a random access
25 memory (RAM) or a cache. The non-volatile memory may include, for example, a read-only memory (ROM), a hard disk, a flash memory, etc. One or more computer program instructions may be stored on the computer-readable storage medium, and the program instructions may be run by the processor 1501 to implement the data storage method according to the various embodiments of this application described above and/or other

desired functions. Various contents such as a big data error parameter of the mechanical equipment may also be stored in the computer-readable storage medium.

In an example, the mechanical equipment 150 may further include an input device 1503 and an output device 1504. These components are connected to each other by a bus system and/or any other form of connecting mechanism (not shown).

The input device 1503 may include, for example, a keyboard, a mouse, a joystick, a touch screen, etc.

The output device 1504 may output various information, including work data and the like, to the outside. The output device 1504 may include, for example, a display, and a communication network and a remote output device connected thereto, etc.

Certainly, for simplicity, only some of the components in the mechanical equipment 150 that relate to this application are shown in FIG. 16, and the components such as a bus and an input/output interface are omitted. In addition, the mechanical equipment 150 may further include any other suitable components according to the specific application.

In addition to the above method and equipment, the embodiments of this application may further provide a computer program product, including a computer program instruction that, when run by a processor, causes the processor to perform the steps of the data storage method according to the various embodiments of this application described in this specification.

The computer program product may write a program code for performing the operation of the embodiments of this application in any combination of one or more programming languages, where the programming languages include object-oriented programming languages such as Java and C++, and further include conventional procedural programming languages such as "C" language or similar programming languages. The program code may be completely executed on a computing device of a user, partially executed on a device of a user, executed as an independent software package, partially executed on a computing device of a user and partially executed on a remote computing device, or completely executed on a remote computing device or a server.

In addition, the embodiments of this application may further provide a

computer-readable storage medium, having a computer program instruction stored thereon, where the computer program instruction, when run by a processor, causes the processor to perform the steps of the data storage method according to the various embodiments of this application in this specification.

5 The computer-readable storage medium may be any combination of one or more readable media. The readable medium may be a readable signal medium or a readable storage medium. The readable storage medium may include, for example, but not limited to, electric, magnetic, optical, electromagnetic, infrared, or semiconductor systems, apparatuses or devices, or any combination thereof. More specific examples (a non-exhaustive list) of the
10 readable storage medium include an electrical connection having one or more wires, a portable disk, a hard disk, an RAM, an ROM, an erasable programmable read-only memory (EPROM or flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination thereof.

15 The basic principles of this application have been described above with reference to the specific embodiments. However, it is to be pointed out that the advantages, effects, etc., mentioned in this application are merely examples rather than limitations, and cannot be regarded as essential to the various embodiments of this application. In addition, the specific details disclosed above are only for the purposes of providing examples and facilitating
20 understanding rather than limitations. The above details do not limit the case where this application must be implemented with the above specific details.

 The block diagrams of the devices, apparatuses, equipment and systems referred to in this application are merely illustrative examples and are not intended to require or imply that they must be connected, arranged or configured in the manner shown in the block diagrams.
25 As will be recognized by those skilled in the art, the devices, apparatuses, equipment and systems may be connected, arranged or configured in any manner. The terms such as "include", "contain", and "have/has" are open-ended terms, refer to "include but not limited to", and may be used interchangeably therewith. The terms "or" and "and" used herein refer to a term "and/or", and may be used interchangeably therewith, unless the context clearly

indicates otherwise. The term "such as" used herein refers to a term "such as but not limited to" and may be used interchangeably therewith.

It is also to be pointed out that the components or steps in the apparatus, equipment and method of this application may be decomposed and/or recombined. These
5 decompositions and/or recombinations are to be regarded as equivalent solutions of this application.

The above description of the disclosed aspects is provided to enable any person skilled in the art to make or use this application. The various modifications to these aspects are very obvious to those skilled in the art, and the general principles defined herein may be applied
10 to other aspects without departing from the scope of this application. Therefore, this application is not intended to be limited to the aspects shown herein but is to be in accordance with the widest scope consistent with the principles and novel features of the present invention.

The above are only preferred embodiments of this application and are not intended to
15 limit this application. Any modifications, equivalent substitutions, etc. made within the spirit and principle of this application shall be included within the scope of protection of this application.

CLAIMS

1. A data storage method, comprising:

obtaining work data;

establishing a message queue, and sequentially adding the work data into the message
5 queue;

determining whether a storage engine uses a sequence strategy or a thread strategy
based on a data volume of the work data;

when the storage engine uses the sequence strategy, sequentially extracting the work
data from the message queue in a single-thread mode based on an adding sequence of the
10 work data added to the message queue;

when the storage engine uses the thread strategy, extracting the work data from the
message queue in a multi-thread mode; and

performing data cleaning on the work data extracted from the message queue, and
storing normal data obtained by cleaning in a database.

2. The data storage method according to claim 1, wherein the performing data cleaning
on the work data extracted from the message queue comprises:

screening the work data to obtain abnormal data; and

storing the abnormal data in the database or an abnormal log;

wherein the abnormal data comprises at least one type of wrong data, unknown data,
20 expired data, duplicate data, unordered data, and redundant data.

3. The data storage method according to claim 2, wherein when the abnormal data
comprises the wrong data, the screening the work data to obtain abnormal data comprises:

performing message parsing on the work data; and

when the parsing fails, determining that the work data is the wrong data.

4. The data storage method according to claim 2, wherein when the abnormal data
comprises the wrong data and the unknown data, the screening the work data to obtain
abnormal data comprises:

performing message parsing on the work data;

when the parsing fails, determining that the work data is the wrong data;

when the parsing succeeds, determining whether the work data meets a preset acquisition requirement; and

when the work data does not meet the acquisition requirement, determining that the work data is the unknown data.

5 5. The data storage method according to claim 2, wherein when the abnormal data comprises the wrong data, the unknown data, and the expired data, the screening the work data to obtain abnormal data comprises:

performing message parsing on the work data;

when the parsing fails, determining that the work data is the wrong data;

10 when the parsing succeeds, determining whether the work data meets a preset acquisition requirement;

when the work data does not meet the acquisition requirement, determining that the work data is the unknown data;

15 when the work data meets the acquisition requirement, determining whether an acquisition time point for acquiring the work data is within a preset time range; and

when the acquisition time point is not within the preset time range, determining that the work data is the expired data.

20 6. The data storage method according to claim 2, wherein when the abnormal data comprises the wrong data, the unknown data, the expired data, and the unordered data, the screening the work data to obtain abnormal data comprises:

performing message parsing on the work data;

when the parsing fails, determining that the work data is the wrong data;

when the parsing succeeds, determining whether the work data meets a preset acquisition requirement;

25 when the work data does not meet the acquisition requirement, determining that the work data is the unknown data;

when the work data meets the acquisition requirement, determining whether an acquisition time point for acquiring the work data is within a preset time range;

when the acquisition time point is not within the preset time range, determining that the

work data is the expired data;

when the acquisition time point is within the preset time range and a current strategy is the sequence strategy, determining whether the current work data and the previous work data acquired before the current work data conform to the adding sequence; and

5 when the current work data and the previous work data do not conform to the adding sequence, determining that the work data is the unordered data.

7. The data storage method according to claim 2, wherein when the abnormal data comprises the wrong data, the unknown data, the expired data, the unordered data, and the redundant data, the screening the work data to obtain abnormal data comprises:

10 performing message parsing on the work data;

when the parsing fails, determining that the work data is the wrong data;

when the parsing succeeds, determining whether the work data meets a preset acquisition requirement;

15 when the work data does not meet the acquisition requirement, determining that the work data is the unknown data;

when the work data meets the acquisition requirement, determining whether an acquisition time point for acquiring the work data is within a preset time range;

when the acquisition time point is not within the preset time range, determining that the work data is the expired data;

20 when the acquisition time point is within the preset time range and a current strategy is the sequence strategy, determining whether the current work data and the previous work data acquired before the current work data conform to the adding sequence;

when the current work data and the previous work data do not conform to the adding sequence, determining that the work data is the unordered data;

25 when the current work data and the previous work data conform to the adding sequence, determining whether the current work data is the same as the previous work data; and

when the current work data is the same as the previous work data, determining that the work data is the redundant data.

8. The data storage method according to claim 7, further comprising:

when the current work data is not the same as the previous work data, determining that the work data is the normal data.

9. The data storage method according to claim 8, further comprising:

when the storage engine uses the thread strategy, counting a number of threads;

5 obtaining a number of messages of the work data extracted from the message queue;

obtaining a data volume of the normal data in the number of messages of the work data;

and

obtaining a data volume of the abnormal data in the number of messages of the work data.

10 10. The data storage method according to any one of claims 2 to 9, wherein when the abnormal data comprises the duplicate data, the screening the work data to obtain abnormal data comprises:

determining whether the work data violates a uniqueness constraint; and

15 when the work data violates the uniqueness constraint, determining that the work data is the duplicate data.

11. The data storage method according to any one of claims 1 to 10, wherein the data storage method further comprising: determining whether the work data has a preset error; and

20 when the work data has the preset error, determining that the work data is the wrong data.

12. The data storage method according to any one of claims 1 to 11, wherein the data storage method further comprising:

determining that the storage engine uses a Highest security strategy, a Normal security strategy, or a Lowest security strategy based on input information.

25 13. The data storage method according to any one of claims 1 to 12, wherein in the process of sequentially extracting the work data from the message queue, the method further comprises:

when the work data in the message queue is non-multicast data, setting a data volume upper limit of the work data extracted from the message queue;

extracting the work data from the message queue based on the data volume upper limit;
and

when the work data in the message queue is multicast data, splitting the multicast data
into the non-multicast data, and extracting the work data based on the data volume upper
5 limit.

14. The data storage method according to any one of claims 1 to 13, wherein in the
process of sequentially extracting the work data from the message queue, the method further
comprises:

setting a timeout limit for extracting the work data from the message queue; and

10 when there is no data in the message queue and time for extracting the work data does
not exceed the timeout limit, waiting for the work data to be added into the message queue.

15. The data storage method according to any one of claims 1 to 14, wherein in the
process of sequentially extracting the work data from the message queue, the method further
comprises:

15 setting a timeout limit for extracting the work data from the message queue; and

when there is no data in the message queue and the time for extracting the work data
exceeds the timeout limit, stopping extracting the work data from the message queue.

16. The data storage method according to any one of claims 1 to 15, wherein in the
process of sequentially extracting the work data from the message queue, the method further
20 comprises:

determining whether the message queue and the database are online and down;

when the message queue and the database are down for the first time, performing data
cleaning on the extracted work data, saving breakpoint data of the work data extracted from
the message queue, and switching the storage engine to use the thread strategy; and

25 when the message queue and the database are online for the first time after being down,
resuming performing a operation of extracting the work data from the message queue based
on the breakpoint data, and switching the storage engine to use the thread strategy.

17. The data storage method according to any one of claims 1 to 16, wherein the data
storage method further comprising:

controlling the storage engine and the message queue to be regularly disconnected from and reconnected to each other, and controlling the storage engine and the database to be regularly disconnected from and reconnected to each other;

5 when the storage engine and the message queue are disconnected from each other, controlling the storage engine and the message queue to be immediately disconnected from and reconnected to each other;

when the storage engine and the database are disconnected from each other, controlling the storage engine and the database to be immediately disconnected from and reconnected to each other; and

10 when the storage engine and the message queue are successfully disconnected from and reconnected to each other, and the storage engine and the database are successfully disconnected from and reconnected to each other, switching the storage engine to use the thread strategy.

15 18. The data storage method according to any one of claims 1 to 17, wherein the data storage method further comprising:

determining that the storage engine uses an immediate stop strategy based on the input information; and

terminating and destroying a thread when stopping the extraction of the work data.

20 19. The data storage method according to any one of claims 1 to 18, wherein the data storage method further comprising:

determining that the storage engine uses a delayed stop strategy based on the input information;

performing data cleaning on the extracted work data when stopping the extraction of the work data; and

25 terminating and destroying the thread after the data cleaning is completed.

20. An automatic system, comprising:

a data obtaining module configured to obtain work data;

a message queue establishment module communicatively connected to the data obtaining module and configured to establish a message queue and sequentially add the work

data into the message queue;

a strategy determination module communicatively connected to the data obtaining module and configured to determine whether a storage engine uses a sequence strategy or a thread strategy based on a data volume of the work data;

5 a data access module communicatively connected to the message queue establishment module and the strategy determination module separately and configured to: when the storage engine uses the sequence strategy, sequentially extract the work data from the message queue in a single-thread mode based on an adding sequence of the work data added to the message queue; and when the storage engine uses the thread strategy, extract the work
10 data from the message queue in a multi-thread mode; and

a data cleaning module communicatively connected to the data access module and configured to perform data cleaning on the work data extracted from the message queue and store normal data obtained by cleaning in a database.

21. Mechanical equipment, comprising a memory and a processor, wherein the memory
15 has an executable code stored therein, and the processor is configured to execute the executable code, so as to implement the data storage method according to any one of claims 1 to 19.