DATA BACKFILLING METHOD AND SYSTEM, AND STORAGE MEDIUM

TECHNICAL FIELD

[0001] This specification relates to the field of data processing technologies, and in particular, to a data backfilling method and system, and a storage medium.

BACKGROUND

[0002] In a scenario involving a data workflow, a data backfilling requirement usually exists. Data backfilling is usually rerunning the data workflow to process data in a past period of time, to supplement missing data, correct incorrect data, etc.

[0003] In a related technology, data backfilling can be implemented in two manners: full backfilling and empirical backfilling. Full backfilling is performing data backfilling on data in all data tables. Empirical backfilling means that a worker performs, based on rich experience of the worker, data backfilling on a data table on which data backfilling needs to be performed.

[0004] However, if full backfilling is performed, there are disadvantages such as a relatively large quantity of consumed resources and low data backfilling efficiency. If empirical backfilling is performed, there is a relatively high experience requirement for the worker, and there is a disadvantage such as low data backfilling accuracy due to subjective factor impact of the worker.

[0005] It should be noted that, content of the related technology is merely information known to the inventor, and neither means that the information has entered a public domain before an application date of this specification, nor means that the information can become a conventional technology of this specification.

SUMMARY

[0006] This specification provides a data backfilling method and system, and a storage medium, to avoid at least one of the above-mentioned technical problems.

[0007] According to a first aspect, this specification provides a data backfilling method,

including: obtaining a data backfilling request, where the data backfilling request is used to indicate to perform data backfilling from at least one backfilling start task data table to at least one backfilling end task data table; determining an end task set and a start task set in a pre-built task graph based on the data backfilling request, where the task graph is used to represent a plurality of tasks and a dependency relationship between the plurality of tasks, the end task set includes at least one backfilling end task in the plurality of tasks that is used to generate the at least one backfilling end task data table, and the start task set includes at least one backfilling start task in the plurality of tasks that is used to generate the at least one backfilling start task data table; determining, in the task graph, a minimum task subgraph including the end task set and the start task set; and executing each task in the minimum task subgraph, to implement data backfilling.

[0008] According to a second aspect, this specification provides a data backfilling system, including: at least one storage medium, storing at least one instruction set to perform task processing; and at least one processor, communicatively connected to the at least one storage medium, where when running, the at least one processor reads the at least one instruction set, and performs the following operations based on an indication of the at least one instruction set: obtaining a data backfilling request, where the data backfilling request is used to indicate to perform data backfilling from at least one backfilling start task data table to at least one backfilling end task data table; determining an end task set and a start task set in a pre-built task graph based on the data backfilling request, where the task graph is used to represent a plurality of tasks and a dependency relationship between the plurality of tasks, the end task set includes at least one backfilling end task in the plurality of tasks that is used to generate the at least one backfilling end task data table, and the start task set includes at least one backfilling start task in the plurality of tasks that is used to generate the at least one backfilling start task data table; determining, in the task graph, a minimum task subgraph including the end task set and the start task set; and executing each task in the minimum task subgraph, to implement data backfilling.

[0009] According to a third aspect, this specification provides a computer-readable non-transitory storage medium. The computer-readable non-transitory storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the method according to the first aspect is implemented.

[0010] According to a fourth aspect, this specification provides a computer program product, including a computer program. When the computer program is executed by a processor, the method

according to the first aspect is implemented.

[0011] It can be learned from the above-mentioned technical solutions that, according to the data backfilling method and system, and the storage medium provided in this specification, the data backfilling request indicates at least one backfilling start task data table to at least one backfilling end task data table on which backfilling is to be performed, to implement pertinent data backfilling, in other words, implement pertinence of data backfilling. Therefore, requirement-based backfilling can be implemented, or data backfilling with a relatively high priority is implemented for an emergency problem caused by a data fault. In addition, according to the above-mentioned solution, the minimum task subgraph can be determined in the task graph based on the at least one backfilling start task data table to the at least one backfilling end task data table on which backfilling is to be performed, to perform data backfilling on the minimum task subgraph, without a need to perform full task data backfilling. Therefore, resources can be saved, and data backfilling efficiency can be improved. Further, according to the above-mentioned solution, impact on another normal task can be further avoided. For example, a task corresponding to another data table on which data backfilling does not need to be performed can continue to be executed. The above-mentioned solution does not need to depend on working experience of a worker. Therefore, subjective impact of a human factor can be avoided, and objectivity and accuracy of data backfilling can be improved.

[0012] Other functions of the data backfilling method and system, and the storage medium provided in this specification are partially listed in the following descriptions. Creative aspects of the data backfilling method and system, and the storage medium provided in this specification can be fully explained by practice or by using the methods, apparatuses, and combinations described in the following detailed examples.

BRIEF DESCRIPTION OF DRAWINGS

[0013] To describe the technical solutions in the embodiments of this specification more clearly, the following briefly describes the accompanying drawings needed for describing the embodiments. Clearly, the accompanying drawings in the following descriptions show merely some embodiments of this specification, and a person of ordinary skill in the art can still derive other drawings from these accompanying drawings without creative efforts.

[0014] FIG. 1 is a schematic diagram of a framework of a data workflow management system according to an embodiment of this specification;

[0015] FIG. 2 is a schematic diagram of a framework of a minimum task subgraph according to an embodiment of this specification;

[0016] FIG. 3 is a schematic diagram of a scenario of a data backfilling method according to an embodiment of this specification;

[0017] FIG. 4 is a schematic structural diagram of a data backfilling system according to an embodiment of this specification;

[0018] FIG. 5 is a schematic flowchart of a data backfilling method according to an embodiment of this specification;

[0019] FIG. 6 is a schematic diagram of a directed acyclic graph according to an embodiment of this specification;

[0020] FIG. 7 is a schematic diagram of an interface output by a data backfilling system according to an embodiment of this specification;

[0021] FIG. 8 shows a task graph including a backfilling end virtual task and a backfilling start virtual task according to an embodiment of this specification; and

[0022] FIG. 9 is a schematic diagram of a framework of a minimum task subgraph according to another embodiment of this specification.

DESCRIPTION OF EMBODIMENTS

[0023] Example embodiments are described in detail here, and examples of the example embodiments are presented in the accompanying drawings. When the following description relates to the accompanying drawings, unless specified otherwise, same numbers in different accompanying drawings represent same or similar elements. Implementations described in the following example embodiments do not represent all implementations consistent with the present specification. On the contrary, the embodiments are merely examples of apparatuses and methods that are described in the appended claims in details and consistent with some aspects of the present specification.

[0024] It should be understood that the terms "include" and "have" in the embodiments of this specification and any variants thereof are intended to cover the non-exclusive inclusion, for

example, a product or device that includes a series of components are not necessarily limited to those components that are clearly listed, but may include other components that are not clearly listed or inherent to such product or device.

[0025] In the embodiments of this specification, the term "and/or" describes an association relationship between associated objects and indicates that three relationships may exist. For example, "A and/or B" may indicate the following three cases: Only A exists, both A and B exist, and only B exists. The character "/" generally indicates an "or" relationship between the associated objects.

[0026] In the embodiments of this specification, the term "a plurality of" refers to two or more, and other quantifiers are similar to this term.

[0027] The terms "first", "second", "third", etc. in this specification are used to distinguish between similar or same objects or entities, and do not necessarily mean limiting a specific order or sequence, unless otherwise indicated. It should be understood that the terms used in such a way is interchangeable in appropriate cases, for example, can be implemented in an order other than those illustrated or described in the embodiments of this specification.

[0028] The term "unit/module" used in this specification means any known or later developed hardware, software, firmware, artificial intelligence, fuzzy logic, or a combination of hardware code or/and a combination of software code, and can perform functions related to the element.

[0029] To facilitate readers' understanding of this specification, at least some terms in this specification are explained below:

[0030] A data workflow is a set of a series of orderly and executable tasks or steps in a data processing and analysis process. These tasks include data collection, cleaning, conversion, analysis, final visualization or storage, etc., and jointly constitute an automated procedure for completing a specific data processing target. The data workflow defines how data is moved, processed, and transformed at different stages to ensure efficient, consistent, and repeatable data processing. The data workflow helps team members understand overall logic of data processing, and facilitates management and automation of a complex analysis procedure. One data workflow can include a plurality of tasks, and there can be a dependency relationship between different tasks in the plurality of tasks.

[0031] A data workflow management system is a software system, and is specially designed to define, execute, and monitor a data workflow. The data workflow management system provides

a graphical interface or a programming interface, and allows a user to design a workflow and set rules, dependency relationships and scheduling policies of data processing.

[0032] Data backfilling can also be referred to as backfilling, data reprocessing, and historical data reprocessing. In a data workflow management system, data backfilling is usually rerunning a data workflow to process data in a past period of time.

[0033] A directed acyclic graph is a directed graph without a cycle, and data storage of the directed acyclic graph can be implemented by using an adjacency matrix or an adjacency list. The directed acyclic graph includes a plurality of nodes (Vertices) and a plurality of directed edges (Edges). In a data backfilling scenario, one node corresponds to one task, and one directed edge represents a dependency relationship between tasks corresponding to two nodes connected by the directed edge.

[0034] The dependency relationship represented by the directed edge can be referred to as a scheduling dependency, and is a dependency relationship of executing different tasks in a specific logical sequence or condition in a data processing procedure. The dependency relationship usually represents a data processing sequence, and ensures data correctness, integrity, and timeliness.

[0035] A graph database is a database designed to store and manage graph structure data. In the graph database, data is mainly stored in a form of a graph structure. The graph structure includes a node and an edge, the node represents an entity, and the edge represents a relationship between entities. Similarly, in a data workflow scenario, one node corresponds to one task, and one directed edge represents a dependency relationship between tasks corresponding to two nodes connected by the directed edge.

[0036] To facilitate readers' understanding of this specification, application scenarios of this specification are described.

[0037] The technical solutions provided in this specification are applicable to a scenario in which data backfilling needs to be performed. For example, the technical solutions provided in this specification can be applied to scenarios such as a data source update scenario, a workflow change scenario, a system fault scenario, and a data quality problem scenario.

[0038] In the data source update scenario, an original data source is changed or updated, and data backfilling needs to be performed, to reprocess previous data to ensure data consistency and accuracy. In the workflow change scenario, data processing logic (for example, a loading task) changes, and data backfilling needs to be performed, to apply new logic to historical data. In the

system fault scenario, data in a period of time is not correctly processed or is lost due to a system fault or another reason, and data backfilling needs to be performed, to reprocess data that is not correctly processed or is lost. In the data quality problem scenario, after a data quality problem is found, data backfilling needs to be performed, to reprocess historical data to correct an error.

[0039] It should be noted that the field to which the technical solutions are applicable is not limited in this specification. For example, the technical solutions provided in this specification can be applied to fields such as industrial production, an e-commerce platform, and an enterprise management system. For example, the technical solutions provided in this specification can be applied to a workflow change in industrial production, or can be applied to a data quality problem in the e-commerce platform, or can be applied to the workflow change in the enterprise management system, etc.

[0040] For example, the technical solutions provided in this specification are applied to the e-commerce platform. A data workflow management system can be deployed on the e-commerce platform, to manage a data workflow related to a marketing activity.

[0041] As shown in FIG. 1, a data workflow includes a plurality of tasks, and different tasks can have a dependency relationship. In FIG. 1, a directed arrow connecting two tasks represents a dependency relationship between the two tasks connected by the directed arrow. For example, a directional arrow between "G-table task" and "M-table task" in FIG. 1 represents that "M-table task" depends on "G-table task". In other words, "M-table task" can be executed only after "G-table task" is executed.

[0042] Correspondingly, one task corresponds to one data table, and is used to generate data in the data table. In other words, after a task is executed, data in a data table corresponding to the task can be generated. For example, data in "G-table" is generated after "G-table task" is executed; and data in "M-table" is generated after "M-table task" is executed.

[0043] Relatively, a task can have an upstream task and a downstream task. A K-table task is used as an example. A task on which the K-table task directly depends or indirectly depends can be referred to as an upstream task of the K-table task. For example, with reference to FIG. 1, a B-table task, a C-table task, the G-table task, etc. can be referred to as an upstream task of the K-table task. A task that directly depends or indirectly depends on the K-table task can be referred to as a downstream task of the K-table task. For example, with reference to FIG. 1, the M-table task, a P-table task, an S-table task, etc. can be referred to as a downstream task of the K-table task.

[0044] In the data workflow in the above-mentioned example, data backfilling usually needs to be performed. For example, with reference to FIG. 1, it can be learned that if "User level table" generated by "User level table task" has incorrect data due to a specific reason, theoretically, data tables (for example, a discount transaction table and a W-table) generated by all downstream tasks of "User level table task" may be incorrect. In this case, data backfilling needs to be performed.

[0045] In a related technology, there are two manners of data backfilling: full backfilling and empirical backfilling.

[0046] Full backfilling is performing data backfilling on all data tables or all incorrect data tables. For example, a directed acyclic graph representing a dependency relationship between tasks is pre-built. When data in a data table corresponding to a task is incorrect, the data table corresponding to the task and data tables corresponding to all downstream tasks of the task may be incorrect. In this case, the task and all downstream tasks of the task can be re-executed based on the directed acyclic graph, to perform data backfilling on an incorrect data table.

[0047] For example, with reference to FIG. 1, it can be learned that one rectangular box in FIG. 1 represents one task, and one task corresponds to one data table. It is assumed that "G-table" has incorrect data due to a specific reason, "G-table task" and all downstream tasks of "G-table task" need to be re-executed in a full backfilling manner.

[0048] However, when data backfilling is performed in the full backfilling manner, at least the following disadvantages exist: Because full backfilling needs to be performed on a data table in which an error occurs and all downstream data tables of the data table, resources are wasted, and execution of a normal task (namely, a task corresponding to a data table whose data is correct) is affected. In addition, a priority of a data table is not considered in the full backfilling manner. Therefore, preferential backfilling cannot be implemented for some relatively more important data tables. In other words, pertinent preferential backfilling cannot be implemented.

[0049] Empirical backfilling means that a worker performs, based on rich experience of the worker, data backfilling on a data table on which data backfilling needs to be performed. For example, when data of a data table is incorrect, the worker manually determines, based on working experience of the worker, which tasks in the data workflow are to be re-executed, to implement data backfilling.

[0050] For example, with reference to FIG. 1, it can be learned that if data in "G-table" is incorrect, the worker determines, based on working experience of the worker, that a data table that

may be incorrect because "G-table" is incorrect includes "Full transaction table", "M-table", "User level table", "P-table", and "Discount transaction table". Therefore, the worker re-executes tasks corresponding to "Full transaction table", "M-table", "User level table", "P-table", and "Discount transaction table" based on the above-mentioned analysis result, to implement data backfilling on the above-mentioned data table.

[0051] However, when data backfilling is performed in the empirical backfilling manner, at least the following disadvantages exist: Data backfilling strongly depends on a professional capability of the worker, and is prone to a disadvantage of low accuracy due to subjective impact of a person or poor professional capability. In addition, for a scenario in which there is a large quantity of upstream and downstream data tables, accuracy may be low in empirical backfilling.

[0052] It should be noted that, content of the related technology is merely information known to the inventor, and neither means that the information has entered a public domain before an application date of this specification, nor means that the information can become a conventional technology of this specification.

[0053] Therefore, to avoid at least one of the above-mentioned problems that occur due to full backfilling and empirical backfilling, this specification proposes a creative technical concept: When data backfilling needs to be performed from at least one backfilling start task data table to at least one backfilling end task data table, an end task set of at least one backfilling end task used to generate at least one backfilling end task data table is obtained from a task graph, a start task set of at least one backfilling start task used to generate at least one backfilling start task data table is obtained from the task graph, a minimum task subgraph including the above-mentioned two task sets (the end task set and the start task set) is determined from the task graph, and each task in the minimum task subgraph is executed, to implement data backfilling.

[0054] For example, if a data backfilling request is used to indicate to perform data backfilling from a backfilling start task data table "G-table" to a backfilling end task data table "Y-table", the end task set includes one backfilling end task, and the backfilling end task is specifically "Y-table task" shown in FIG. 1; and the start task set includes one backfilling start task, and the backfilling start task is specifically "G-table task" shown in FIG. 1.

[0055] Correspondingly, a minimum task subgraph including "Y-table task" and "G-table task" can be determined from the task graph shown in FIG. 1, and the minimum task subgraph can be shown in FIG. 2.

[0056] After the minimum task subgraph shown in FIG. 2 is obtained, a data backfilling system can execute each task in the minimum task subgraph, to implement data backfilling. For example, the data backfilling system can sequentially execute "G-table task", "M-table task", "User level table task", "Discount transaction table task", "User discount batch table task", and "Y-table task", to implement data backfilling on "Y-table task", so as to correct incorrect data in "Y-table".

[0057] Based on the above-mentioned analysis, it can be learned that, in this specification, data backfilling is performed by using the technical solution including the above-mentioned technical concept, and the data backfilling request indicates a backfilling start task data table to a backfilling end task data table on which backfilling is to be performed, to implement pertinent data backfilling. Therefore, requirement-based backfilling can be implemented, or data backfilling with a relatively high priority is implemented for an emergency problem caused by a data fault. In addition, according to the technical solution including the above-mentioned technical concept, the minimum task subgraph can be determined in the task graph based on the backfilling start task data table to the backfilling end task data table on which backfilling is to be performed, to perform data backfilling on the minimum task subgraph, without a need to perform full task data backfilling. Therefore, resources can be saved, and data backfilling efficiency can be improved. Further, according to the technical solution including the above-mentioned technical concept, impact on another normal task can be further avoided. For example, a task corresponding to another data table on which data backfilling does not need to be performed can continue to be executed. The technical solution including the above-mentioned technical concept does not need to depend on working experience of a worker. Therefore, subjective impact of a human factor can be avoided, and objectivity and accuracy of data backfilling can be improved.

[0058] FIG. 3 is a schematic diagram of an application scenario of a data backfilling method according to an embodiment of this specification. As shown in FIG. 3, an application scenario 300 can include a target user 301, a client 302, a server 303, and a network 304.

[0059] The target user 301 can be a user who triggers data backfilling. In other words, the target user 301 can be a user who initiates a data backfilling request. For example, the target user 301 can perform a target operation on the client 302, to trigger data backfilling.

[0060] The client 302 can be an electronic device that responds to the target operation performed by the target user 301. In some embodiments, the client 302 can perform data backfilling from at least one backfilling start task data table to at least one backfilling end task data

table in response to the data backfilling request of the target user 301. That is, the data backfilling method described below in this specification can be performed on the client 302. In this case, the client 302 can store data or instructions for performing the data backfilling method described in this specification, and can execute or be configured to execute the data or the instructions. In some embodiments, the client 302 can include a hardware device with a data information processing function and a program needed to drive the hardware device to work, to perform the data backfilling method described in this specification.

[0061] In some embodiments, the client 302 can include a mobile device, a tablet computer, a laptop computer, a device built in a motor vehicle or similar content, or any combination thereof. In some embodiments, the mobile device can include a smart home device, a smart mobile device, a virtual reality device, an augmented reality device or a similar device, or any combination thereof. In some embodiments, the smart home apparatus can include a smart television, a desktop computer, or any combination. In some embodiments, the smart mobile device can include a smartphone, a personal digital assistant, a game device, a navigation device, or any combination thereof. In some embodiments, the apparatus built in the motor vehicle can include a vehicle-mounted computer, a vehicle-mounted television, etc. In some embodiments, the client 302 can include a collection device, configured to collect the data backfilling request.

[0062] In some embodiments, one or more applications (app) can be installed on the client 302. The app can provide the target user 301 with a capability and an interface for interacting with the outside through the network 304. The app includes but is not limited to a web browser app program, a search app program, a chat app program, a shopping app program, a video app program, a financial app program, an instant messaging tool, a mailbox client, social platform software, etc. In some embodiments, a target app can be installed on the client 302. The target app can collect the data backfilling request for the client 302.

[0063] As shown in FIG. 3, the client 302 can be communicatively connected to the server 303. The server 303 can communicate with one client 302, or can communicate with a plurality of clients 302. In some embodiments, the client 302 can interact with the server 303 through the network 304, to receive or send a message, etc. For example, the client 302 can interact with the server 303 through the network 304, to send the data backfilling request to the server.

[0064] The server 303 can be a server that provides various services, for example, a cloud server or a local server.

[0065] In some embodiments, the server 303 can include a hardware device with a data information processing function and a necessary program needed for driving the hardware device to work. Similarly, the server 303 can be communicatively connected to one client 302, and receive data sent by the client 302; or can be communicatively connected to a plurality of clients 302, and receive data sent by each client 302.

[0066] In some embodiments, the data backfilling method can be performed on the server 303. In this case, the server 303 can store data or instructions for performing the data backfilling method described in this specification, and can execute or be configured to execute the data or the instructions.

[0067] The network 304 is a medium configured to provide a communication connection between the client 302 and the server 303. The network 304 can facilitate exchange of information or data. As shown in FIG. 3, the client 302 and the server 303 can be separately connected to the network 304, and transmit information or data to each other through the network 304.

[0068] In some embodiments, the network 304 can be any type of wired or wireless network, or a combination thereof. For example, the network 304 can include a cable network, a wired network, an optical fiber network, a telecommunication network, an intranet, the internet, a local area network (LAN), a wide area network (WAN), a wireless local area network (WLAN), a metropolitan area network (MAN), a public switched telephone network (PSTN), a Bluetooth networkTM, a short-range wireless network (ZigBeeTM), a near field communication (NFC) network, or a similar network.

[0069] In some embodiments, the network 304 can include one or more network access points. For example, the network 304 can include a wired or wireless network access point, for example, a base station or an Internet switching point. Through the access point, one or more components of the client 302 and the server 303 can be connected to the network 304 to exchange data or information.

[0070] It should be noted that quantities of clients 302, servers 303, and networks 304 in FIG. 3 are merely examples. There can be any quantity of clients 302, servers 303, and networks 304 based on implementation needs. In addition, the data backfilling method provided in this specification can be completely performed on the client 302, can be completely performed on the server 303, or can be partially performed on the client 302 and partially performed on the server 303.

[0071] In other words, FIG. 3 and the above-mentioned descriptions for FIG. 3 are merely used to describe an example of an application scenario to which the data backfilling method in this specification may be applied, and cannot be understood as a limitation on the application scenario.

[0072] FIG. 4 is a diagram of a hardware structure of a data backfilling system 400 according to an embodiment of this specification. The data backfilling system 400 can perform a data backfilling method described in this specification. The data backfilling method is described in another part of this specification. When the data backfilling method is performed on a client 302, the data backfilling system 400 can be the client 302. When the data backfilling method is performed on a server 303, the data backfilling system 400 can be the server 303. When the data backfilling method is partially performed on a client 302 and partially performed on a server 303, the data backfilling system 400 can be a system including the client 302 and the server 303.

[0073] As shown in FIG. 4, the data backfilling system 400 can include at least one storage medium 403 and at least one processor 402. In some embodiments, the data backfilling system 400 can further include a communication port 404 and an internal communication bus 401. The data backfilling system 400 can further include an I/O component 405.

[0074] The internal communication bus 401 can be connected to different system components. For example, the internal communication bus 401 can be connected to the storage medium 403, the processor 402, the communication port 404, and the I/O component 405.

[0075] The I/O component 405 supports an input/output between the data backfilling system 400 and another component.

[0076] The communication port 404 is used for data communication between the data backfilling system 400 and the outside. For example, the communication port 404 can be used for data communication between the data backfilling system 400 and the network 404. The communication port 404 can be a wired communication port, or can be a wireless communication port.

[0077] The storage medium 403 can include a data storage apparatus. The data storage apparatus may be a non-transitory storage medium, or may be a transitory storage medium. For example, the data storage apparatus may include one or more of a disk 4031, a read-only storage medium (ROM) 4032, or a random access storage medium (RAM) 4033. The storage medium 403 further includes at least one instruction set stored in the data storage apparatus. The instruction set includes computer program code, and the computer program code can include a program, a routine,

an object, a component, a data structure, a process, a module, etc. for performing the data backfilling method provided in this specification.

[0078] The at least one processor 402 can be communicatively connected to the at least one storage medium 403. The at least one processor 402 is configured to execute the at least one instruction set. When the data backfilling system 400 runs, the at least one processor 402 reads the at least one instruction set and performs, based on an indication of the at least one instruction set, the data backfilling method provided in this specification. The processor 402 can perform all steps included in the data backfilling method. The processor 402 can be in a form of one or more processors. In some embodiments, the processor 402 can include one or more hardware processors, for example, a microcontroller, a microprocessor, a reduced instruction set computer (RISC), an application-specific integrated circuit (ASIC), an application-specific instruction set processor (ASIP), a central processing unit (CPU), a graphics processing unit (GPU), a physical processing unit (PPU), a microcontroller unit, a digital signal processor (DSP), a field programmable gate array (FPGA), an advanced RISC machine (ARM), a programmable logic device (PLD), any circuit or processor that can perform one or more functions, or any combination thereof.

[0079] To merely describe a problem, only one processor 402 is shown in the data backfilling system 400 in the accompanying drawings. However, it should be noted that the data backfilling system 400 in this specification can further include a plurality of processors. Therefore, the operations and/or method steps disclosed in this specification can be performed by one processor, or can be jointly performed by a plurality of processors. For example, if the processor 402 in the data backfilling system 400 in this specification performs step A and step B, it should be understood as that step A and step B can be jointly or separately performed by two different processors 402 (for example, a first processor performs step A, and a second processor performs step B, or a first processor and a second processor jointly perform step A and step B).

[0080] FIG. 5 is a schematic diagram of a data backfilling method according to an embodiment of this specification. As shown in FIG. 5, the method includes S501 to S504.

[0081] S501: Obtain a data backfilling request, where the data backfilling request is used to indicate to perform data backfilling from at least one backfilling start task data table to at least one backfilling end task data table.

[0082] For example, the data backfilling method in this embodiment can be performed by a data backfilling system. For descriptions of the data backfilling system, references can be made to

the above-mentioned example. Details are omitted here for simplicity.

[0083] This embodiment sets no limitation on a manner in which the data backfilling system obtains the data backfilling request. An example is as follows:

[0084] In an example, the data backfilling system can be connected to another system, and receive a data backfilling request sent by the another system.

[0085] For example, the above-mentioned example and the application scenario shown in FIG. 3 are used as examples. The data backfilling system can be the server 303 shown in FIG. 3, and the another system can be the client 302 shown in FIG. 3. A target user 301 can input a data backfilling request on the client 302 in an app manner or a non-app manner, to trigger the client 302 to send the data backfilling request to the server 303.

[0086] In another example, the data backfilling system can provide a tool for loading the data backfilling request, and the target user can transmit the data backfilling request to the data backfilling system by using the tool for loading the data backfilling request.

[0087] The tool for loading the data backfilling request can be an interface used to be connected to an external device, for example, an interface used to be connected to another storage device, to obtain, through the interface, the data backfilling request transmitted by the external device. The tool for loading the data backfilling request can also be a display apparatus, for example, the data backfilling system can output, on the display apparatus, an interface of a function of loading the data backfilling request. The target user can import the data backfilling request to the data backfilling system in the interface. The data backfilling system obtains the imported data backfilling request.

[0088] In some embodiments, S501 can be understood as follows: When the target user finds that data in an upstream task data table is incorrect and data in a downstream task data table is affected, to correct incorrect data in the upstream task data table to the downstream task data table, so that data in the upstream task data table to the downstream task data table is accurate and reliable data, the target user can initiate the data backfilling request to the data backfilling system, and correspondingly, the data backfilling system obtains the data backfilling request.

[0089] This specification sets no limitation on a reason for a need to perform data backfilling. For example, data backfilling needs to be performed because data in the upstream task data table to the downstream task data table is incorrect, or data backfilling needs to be performed because data in the upstream task data table to the downstream task data table is missing, or data backfilling

needs to be performed from the upstream task data table to the downstream task data table due to another reason.

[0090] There can be one or more backfilling start task data tables and/or backfilling end task data tables. This specification sets no limitation thereto. In an actual application scenario, the user can combine an actual scenario requirement.

[0091] With reference to the above-mentioned descriptions of S501, it can be learned that in this embodiment, a data backfilling process is triggered by the data backfilling request, and the data backfilling request can be specific to a specific backfilling start task data table to a specific backfilling end task data table. To be specific, the data backfilling system can perform data backfilling on the backfilling start task data table to the backfilling end task data table that are indicated by the data backfilling request, and does not need to perform full backfilling on full data tables, to implement pertinence and flexibility of data backfilling, and improve data backfilling efficiency.

[0092] S502: Determine an end task set and a start task set in a pre-built task graph based on the data backfilling request, where the task graph is used to represent a plurality of tasks and a dependency relationship between the plurality of tasks, the end task set includes at least one backfilling end task in the plurality of tasks that is used to generate the at least one backfilling end task data table, and the start task set includes at least one backfilling start task in the plurality of tasks that is used to generate the at least one backfilling start task data table.

[0093] In this specification, the backfilling start task is a task used to generate the backfilling start task data table. One backfilling start task data table corresponds to one backfilling start task. The backfilling end task is a task used to generate the backfilling end task data table. One backfilling end task data table corresponds to one backfilling end task. The data backfilling request can include one or more backfilling start task data tables, or can include one or more backfilling end task data tables.

[0094] When there is one backfilling end task data table, the end task set can include one backfilling end task corresponding to the one backfilling end task data table. For example, with reference to the above-mentioned example, that the backfilling end task data table is "Y-table" is used as an example. A task used to generate the backfilling end task data table is "Y-table task". In this case, the data backfilling system can obtain a backfilling end task "Y-table task" corresponding to the backfilling end task data table "Y-table" from the task graph shown in FIG. 1. In this case,

the end task set includes one backfilling end task, namely, "Y-table task".

[0095] When there are a plurality of backfilling end task data tables, the end task set can include backfilling end tasks respectively corresponding to the plurality of backfilling end task data tables. For example, that there are two backfilling end task data tables is used as an example. It is assumed that the backfilling end task data table includes "Y-table" and "W-table", a task used to generate "Y-table" is "Y-table task", and a task used to generate "W-table" is "W-table task". In this case, the data backfilling system can obtain a backfilling end task "Y-table task" corresponding to "Y-table" and obtain a backfilling end task "W-table task" corresponding to "W-table" from the task graph shown in FIG. 1. In this case, the end task set includes two backfilling end tasks: "Y-table task" and "W-table task".

[0096] Correspondingly, when there is one backfilling start task data table, the start task set can include one backfilling start task corresponding to the backfilling start task data table. For example, with reference to the above-mentioned example, that the backfilling start task data table is "G-table" is used as an example. A task used to generate the backfilling end task data table is "G-table task". In this case, the data backfilling system can obtain a backfilling start task "G-table task" in the task graph shown in FIG. 1. In this case, the start task set includes one backfilling start task, namely, "G-table task".

[0097] When there are a plurality of backfilling start task data tables, the start task set can include backfilling start tasks respectively corresponding to the plurality of backfilling start task data tables. For example, that there are two backfilling start task data tables is used as an example. It is assumed that the backfilling start task data table includes "G-table" and "L-table". A task used to generate "G-table" is "G-table task", and a task used to generate "L-table" is "L-table task". In this case, the data backfilling system can obtain a backfilling start task "G-table task" corresponding to "G-table" and obtain a backfilling start task "L-table task" corresponding to "L-table" from the task graph shown in FIG. 1. In this case, the start task set includes two backfilling start tasks: "G-table task" and "L-table task".

[0098] With reference to the above-mentioned descriptions of S502, it can be learned that in this embodiment, data backfilling is performed based on the backfilling start task to the backfilling end task, and no other task needs to be considered. In this way, unnecessary data backfilling is avoided while data backfilling accuracy is ensured, to improve data backfilling efficiency and save data backfilling resources.

[0099] With reference to the above-mentioned analysis, it can be learned that the task graph can be built based on each task in a data workflow and a dependency relationship between the tasks. A specific form of the task graph is not limited in this embodiment.

[0100] In some embodiments, the task graph can be a directed acyclic graph. The directed acyclic graph includes a plurality of nodes and a plurality of directed edges, one node corresponds to one task, and a directed edge connecting two nodes is used to represent a dependency relationship between tasks corresponding to the two nodes.

[0101] For example, as shown in FIG. 6, the directed acyclic graph includes a node 1 and a node 2 to a node 8, and includes seven directed edges.

[0102] With reference to FIG. 1, it can be learned that a task corresponding to the node 1 can be "User level table task", a task corresponding to the node 2 can be "P-table task", a task corresponding to a node 3 can be "Activity industry configuration table task", a task corresponding to a node 4 can be "L-table task", a task corresponding to a node 5 can be "Discount transaction table task", a task corresponding to a node 6 can be "User discount batch table task", a task corresponding to a node 7 can be "Y-table task", and a task corresponding to the node 8 can be "W-table task".

[0103] A directed edge (a directed arrow) between the node 5 and the node 6 is used to represent that the task (for example, "User discount batch table task") corresponding to the node 6 needs to depend on the task (for example, "Discount transaction table task") corresponding to the node 5. In other words, the directed edge between the node 5 and the node 6 is used to represent that the task corresponding to the node 6 needs to be executed after the task corresponding to the node 5 is executed.

[0104] In some embodiments, the data backfilling system can provide an interaction interface. The task graph is displayed in the interaction interface. The target user can select the backfilling end task and the backfilling start task in the task graph. Correspondingly, the data backfilling system can determine the end task set and the start task set based on a selection operation of the target user.

[0105] It should be noted that another interaction manner can also be used in the interaction interface provided by the data backfilling system. This embodiment sets no limitation thereto. For example, the interaction interface can display each task in a form of a table in the interaction interface, and the target user can select the backfilling end task and the backfilling start task from

the table. For another example, the interaction interface can display a plurality of data tables, and the target user can select the backfilling start task data table and the backfilling end task data table from the plurality of data tables. For another example, the interaction interface can include an input box, and the target user can input the backfilling end task and the backfilling start task in the input box in a text input manner.

[0106] For example, that the target user initiates the data backfilling request in the text input manner is used as an example:

[0107] The data backfilling system can output an interface shown in FIG. 7. As shown in FIG. 7, the interface includes an input box (the input box can be referred to as a first input box for ease of distinguishing) of the backfilling end task and an input box (similarly, the input box can be referred to as a second input box for ease of distinguishing) of the backfilling start task.

[0108] The target user can enter the backfilling end task in the first input box. When there are a plurality of backfilling end tasks, the target user can tap a virtual button "+" shown in FIG. 7, to add more first input boxes, so as to specify more backfilling end tasks.

[0109] In some other embodiments, the target user can alternatively enter a scenario identifier of the backfilling end task in the first input box. Correspondingly, the data backfilling system can obtain all backfilling end tasks (which may be one or more backfilling end tasks) of the scenario identifier. The scenario identifier is used to distinguish between different service scenarios or data processing procedures. For example, if q (q is an integer greater than or equal to 1) tables are directly related to final data that can be seen on a home page of the target user, the q tables can be marked with a scenario identifier "XXX-to-visitor display task".

[0110] The target user can enter the backfilling start task in the second input box. When there are a plurality of backfilling start tasks, the target user can tap the virtual button "+" shown in FIG. 7, to add more second input boxes, so as to specify more backfilling start tasks.

[0111] As shown in FIG. 7, the interface further includes a virtual button "OK". After specifying the backfilling end task and backfilling start task in the interface, the target user can tap the virtual button "OK", to determine to send the data backfilling request to the data backfilling system. Correspondingly, the data backfilling system obtains the data backfilling request.

[0112] S503: Determine, in the task graph, a minimum task subgraph including the end task set and the start task set.

[0113] Correspondingly, when the end task set and the start task set are obtained, the data

backfilling system can determine, from the task graph, a minimum task subgraph that includes both the end task set and the start task set and includes a smallest quantity of tasks.

[0114] Similarly, in this step, the minimum task subgraph is determined based on the end task set and the start task set, so that a task included in the minimum task subgraph is a smallest quantity of necessary tasks used to correct data in the at least one backfilling start task data table to the at least one backfilling end task data table, to improve data backfilling validity in a case of a smallest data backfilling amount.

[0115] In some embodiments, S503 can include step 1 and step 2:

[0116] Step 1: Determine an end task in the task graph based on the end task set, and determine a start task in the task graph based on the start task set.

[0117] The end task can be understood as a final (namely, last or downstream) task of data backfilling, or a cut-off task of data backfilling. The start task can be understood as an earliest (namely, initial or uppermost) task of data backfilling, or a start task of data backfilling.

[0118] In some embodiments, the data backfilling system can determine the end task in the task graph based on the end task set in Manner 1, Manner 2, or Manner 3:

[0119] Manner 1: If there is one backfilling end task in the end task set, the backfilling end task in the end task set is determined as the end task.

[0120] For example, with reference to the above-mentioned analysis, it can be learned that the end task set may include only one backfilling end task. In this case, the data backfilling system can directly determine the backfilling end task as the end task.

[0121] Manner 2: If there are a plurality of backfilling end tasks in the end task set, a backfilling end virtual task is generated in the task graph, where the backfilling end virtual task depends on each backfilling end task in the end task set, and the backfilling end virtual task is determined as the end task.

[0122] For example, with reference to the above-mentioned analysis, it can be learned that the end task set may include a plurality of backfilling end tasks. In this case, the data backfilling system can create a virtual task in the task graph (for ease of distinguishing, the task can be referred to as a backfilling end virtual task). It should be noted that the backfilling end virtual task needs to satisfy the following condition: The backfilling end virtual task has a dependency relationship with each backfilling end task in all backfilling end tasks, and the backfilling end virtual task is a downstream task of each backfilling end task.

[0123] For example, with reference to the above-mentioned example and FIG. 8, there are two backfilling end tasks: "Y-table task" and "W-table task". The data backfilling system generates "Backfilling end virtual task" in the task graph. As shown in FIG. 8, "Backfilling end virtual task" has a dependency relationship with each of "Y-table task" and "W-table task", and "Backfilling end virtual task" is a downstream task of "Y-table task" and "W-table task". In other words, "Backfilling end virtual task" depends on "Y-table task" and "W-table task".

[0124] Manner 3: If there are a plurality of backfilling end tasks in the end task set, each backfilling end task in the end task set is separately determined as one end task. In other words, one backfilling end task is determined as one end task.

[0125] With reference to Manner 2 and Manner 3, it can be learned that, when there are a plurality of backfilling end tasks, the data backfilling system can virtualize the plurality of backfilling end tasks to obtain one backfilling end virtual task, that is, can virtualize the plurality of backfilling end tasks into one end task. Alternatively, the data backfilling system can determine one backfilling end task as one end task, without performing virtualization processing.

[0126] With reference to the above-mentioned descriptions of Manner 1, Manner 2, and Manner 3, it can be learned that in this specification, the data backfilling system determines the end task in different manners, to implement diversity and flexibility of determining the end task. Compared with Manner 3, when Manner 2 is used in the data backfilling system, a plurality of end tasks can be simplified to a single end task, to perform a subsequent operation such as determining the minimum task subgraph, that is, achieve a technical effect of a simplified operation.

[0127] Similarly, in some embodiments, the data backfilling system can determine the start task in the task graph based on the start task set in Manner 1, Manner 2, or Manner 3:

[0128] Manner 1: If there is one backfilling start task in the start task set, the backfilling start task in the start task set is determined as the start task.

[0129] For example, with reference to the above-mentioned analysis, it can be learned that the start task set may include only one backfilling start task. In this case, the data backfilling system can directly determine the backfilling start task as the start task.

[0130] Manner 2: If there are a plurality of backfilling start tasks in the start task set, a backfilling start virtual task is generated in the task graph, where each backfilling start task in the start task set depends on the backfilling start virtual task, and the backfilling start virtual task is determined as the start task.

[0131] For example, with reference to the above-mentioned analysis, it can be learned that the start task set may include a plurality of backfilling start tasks. In this case, the data backfilling system can create a virtual task in the task graph (for ease of distinguishing, the task can be referred to as a backfilling start virtual task). It should be noted that the backfilling start virtual task needs to satisfy the following condition: The backfilling start virtual task has a dependency relationship with each backfilling start task in all backfilling start tasks, and the backfilling start virtual task is an upstream task of each backfilling start task.

[0132] For example, with reference to the above-mentioned example and FIG. 8, there are two backfilling start tasks: "User level table task" and "P-table task". The data backfilling system generates "Backfilling start virtual task" in the task graph. As shown in FIG. 8, "Backfilling start virtual task" has a dependency relationship with each of "User level table task" and "P-table task", and "Backfilling start virtual task" is an upstream task of "User level table task" and "P-table task". In other words, "User level table task" and "P-table task" depend on "Backfilling start virtual task".

[0133] Manner 3: If there are a plurality of backfilling start tasks in the start task set, each backfilling start task in the start task set is separately determined as one start task. In other words, one backfilling start task is determined as one start task.

[0134] With reference to Manner 2 and Manner 3, it can be learned that, when there are a plurality of backfilling start tasks, the data backfilling system can virtualize the plurality of backfilling start tasks to obtain one backfilling start virtual task, that is, can virtualize the plurality of backfilling start tasks into one start task. Alternatively, the data backfilling system can determine one backfilling start task as one start task, without performing virtualization processing.

[0135] With reference to the above-mentioned descriptions of Manner 1, Manner 2, and Manner 3, it can be learned that in this specification, the data backfilling system determines the start task in different manners, to implement diversity and flexibility of determining the start task. Compared with Manner 3, when Manner 2 is used in the data backfilling system, a plurality of start tasks can be simplified to a single start task, to perform a subsequent operation such as determining the minimum task subgraph, that is, achieve a technical effect of a simplified operation.

[0136] Step 2: Determine the minimum task subgraph in the task graph based on the start task and the end task.

[0137] Correspondingly, when the start task and the end task are determined, the data

backfilling system can determine, from the task graph, a minimum task subgraph starting with the start task and ending with the end task; or the data backfilling system can determine, from the task graph, a minimum task subgraph starting with the end task and ending with the start task.

[0138] With reference to the above-mentioned analysis of step 1 and step 2, it can be learned that in this specification, the data backfilling system can determine the start task and the end task, and then determine, from the task graph, a minimum task subgraph that includes both the start task and the end task and that includes a quantity of tasks that satisfy a minimum task requirement of data backfilling, to improve validity and reliability of the determined minimum task subgraph.

[0139] In some embodiments, when there is one start task and there is one end task, step 2 can include sub-step 1 and sub-step 2 in the following:

[0140] Sub-step 1: Perform a path search in the task graph by using the start task as a path start and using the end task as a path end, to obtain at least one data backfilling path.

[0141] For example, the data backfilling system can perform a path search in the task graph in a preset path search method by using the start task as a search start and using the end task as a search end, to obtain one or more data backfilling paths.

[0142] The path search method includes one of depth-first search (DFS), recursive backtracking, and dynamic planning.

[0143] For example, that the data backfilling system determines at least one data backfilling path based on the depth first search is used as an example, including (1) to (6):

[0144] (1) Initialization:

[0145] Two lists are created: List 1 and List 2. List 1 is used to store a currently searched path, and List 2 is used to record all reachable paths (namely, data backfilling paths) from a path start to a path end.

[0146] (2) Depth-first search:

[0147] A current task is added to List 1. Whether the current task is a path end is checked. If the current task is a path end, a copy of the list is added to List 2, and is used as a reachable path. If the current task is not a path end, (3) continues to be performed.

[0148] (3) Explore a downstream node

[0149] All downstream tasks of the current task are traversed, and if the downstream task does not exist in List 1, a depth-first search function is recursively invoked for the downstream node.

[0150] (4) Backtracking

[0151] After all downstream tasks of the current task are explored, the current task is removed from List 1, and a task of a previous step is performed again to continue to explore another possible path.

[0152] (5) (1) to (4) are repeated, until all tasks in the task graph are searched recursively, and corresponding reachable paths are recorded or backtracked, to obtain all data backfilling paths.

[0153] (6) All the data backfilling paths are returned.

[0154] After a depth-first search starting from the start task is completed, List 2 includes all reachable paths from the path start to the path end, and List 2 is returned and used as an output of an algorithm.

[0155] Sub-step 2: Determine the minimum task subgraph based on the at least one data backfilling path.

[0156] Correspondingly, after determining all the reachable paths, the data backfilling system can determine the minimum task subgraph based on all the reachable paths.

[0157] With reference to sub-step 1 and sub-step 2, it can be learned that in this embodiment, the minimum task subgraph is determined by using the start task as a start and using the end task as an end. Therefore, requirement-based backfilling can be implemented based on an actual search requirement (for example, the end task is a data backfilling requirement) of the user, and no full backfilling needs to be performed. In addition, a task included in the minimum task subgraph is a smallest quantity of necessary tasks used to correct data in the at least one backfilling start task data table to the at least one backfilling end task data table. Therefore, normal running of another task outside the minimum task subgraph is not affected.

[0158] In the above-mentioned example, how to determine the minimum task subgraph is described by using one start task and one end task as an example. In some other embodiments, when there is one end task, there can be a plurality of start tasks, and step 2 can include sub-step 1 to sub-step 3 in the following:

[0159] Sub-step 1: Perform a path search in the task graph by using the 1st start task as a path start and using the end task as a path end, to obtain at least one data backfilling path corresponding to the 1st start task, and generate an intermediate task subgraph based on the at least one data backfilling path corresponding to the 1st start task.

[0160] The 1st start task is any of the plurality of start tasks. To be specific, the data backfilling system can use any of the plurality of start tasks as the 1st start task.

[0161] Correspondingly, the data backfilling system can determine one or more corresponding data backfilling paths in the task graph for the 1st start task and the end task, and then determine a corresponding task graph with reference to the one or more data backfilling paths. For ease of distinguishing, the task graph is referred to as an intermediate task subgraph.

[0162] Sub-step 2: For the ith start task, where a value of i is successively an integer from 2 to N, separately perform the following operations: performing a path search in the task graph by using the ith start task as a path start and using the end task as a path end, to obtain at least one data backfilling path corresponding to the ith start task, and updating the intermediate task subgraph based on the at least one data backfilling path corresponding to the ith start task; and N is a quantity of the plurality of start tasks.

[0163] With reference to the above-mentioned analysis, when the plurality of start tasks are N start tasks, the data backfilling system can label the N start tasks as the 1st start task and the 2nd start task to the Nth start task.

[0164] Correspondingly, the data backfilling system can first determine the intermediate task subgraph corresponding to the 1st start task based on sub-step 1. Then, the data backfilling system can determine one or more data backfilling paths corresponding to the 2nd start task, and update the intermediate task subgraph based on the one or more data backfilling paths corresponding to the 2nd start task. Then, the data backfilling system can determine one or more data backfilling paths corresponding to the 3rd start task, and update, based on the one or more data backfilling paths corresponding to the 3rd start task, an intermediate task subgraph (namely, an intermediate task subgraph obtained through updating based on the 2nd start task) obtained through previous updating. By analogy, the data backfilling system can determine one or more data backfilling paths corresponding to the Nth start task, and update, based on the one or more data backfilling paths corresponding to the Nth start task, an intermediate task subgraph (namely, an intermediate task subgraph obtained through updating based on the $(N-1)^{th}$ start task) obtained through previous updating.

[0165] Sub-step 3: Determine an updated intermediate task subgraph as the minimum task subgraph.

[0166] Correspondingly, the data backfilling system can determine that a result of updating the intermediate task subgraph based on the one or more data backfilling paths corresponding to the Nth start task is the minimum task subgraph.

[0167] With reference to sub-step 1 to sub-step 3, it can be learned that in this embodiment, the data backfilling system determines the intermediate task subgraph based on one of the plurality of start tasks, and then updates the intermediate task subgraph based on another task node of the plurality of start tasks. An intermediate task subgraph that may include each of the plurality of start tasks is fully considered, to avoid omitting a task in the minimum task subgraph, and improve validity and reliability of the minimum task subgraph. In addition, disadvantages such as high resource consumption and cumbersome processes caused by separately constructing a task subgraph based on each of the plurality of start tasks can be avoided, to improve efficiency of determining the minimum task subgraph.

[0168] With reference to the above-mentioned analysis, it can be learned that there may be further a plurality of end tasks. When there are a plurality of end tasks, there may be one or more start tasks.

[0169] For example, there are a plurality of end tasks, and there is one start task. For each end task, the data backfilling system can determine a data backfilling path between the end task and the start task. After the data backfilling paths corresponding to each end task are obtained, the data backfilling system can de-duplicate all obtained data backfilling paths, to obtain the minimum task subgraph.

[0170] Similarly, for example, there are a plurality of end tasks and a plurality of start tasks: For a start-end task pair including each end task and each start task, the data backfilling system can determine a data backfilling path between the start-end task pair. After the data backfilling paths corresponding to each start-end task pair are obtained, the data backfilling system can de-duplicate all obtained data backfilling paths, to obtain the minimum task subgraph.

[0171] S504: Execute each task in the minimum task subgraph, to implement data backfilling.

[0172] Correspondingly, with reference to the above-mentioned analysis, it can be learned that, when the minimum task subgraph is determined, the data backfilling system can execute each task in the minimum task subgraph, and does not need to perform full backfilling on all tasks, to improve pertinence and accuracy of data backfilling when data backfilling resources are saved.

[0173] In some embodiments, S504 can include step 1 and step 2 in the following:

[0174] Step 1: Determine an execution sequence of the tasks in the minimum task subgraph based on a dependency relationship between the tasks in the minimum task subgraph.

[0175] The execution sequence determined in step 1 satisfies the following condition: For two

tasks that have a dependency relationship in the minimum task subgraph, for example, if the P-table task depends on the K-table task, an execution sequence of the K-table task is before an execution sequence of the P-table task. This specification sets no limitation on an execution sequence of different tasks that do not have a dependency relationship in the minimum task subgraph.

[0176] For example, with reference to the above-mentioned example and the minimum task subgraph shown in FIG. 9, it can be learned that if the backfilling start task data table is "G-table", the start task set includes one backfilling start task, and the backfilling start task is specifically "G-table task" shown in FIG. 1; if the backfilling end task data table includes "Y-table" and "W-table", the end task set includes two backfilling end tasks, and the two backfilling end tasks include "Y-table task" and "W-table task" shown in FIG. 1; and the minimum task subgraph can include a plurality of data backfilling paths, for example, three data backfilling paths: the 1st data backfilling path, the 2nd data backfilling path, and the 3rd data backfilling path.

[0177] Correspondingly, in this step, for any data backfilling path, the data backfilling system can determine the execution sequence of the tasks in the data backfilling path based on the dependency relationship between the tasks in the data backfilling path. In addition, for the three data backfilling paths, the data backfilling system can determine an execution sequence of the three data backfilling paths.

[0178] For example, for the 1st data backfilling path shown in FIG. 9, when determining an execution sequence, the data backfilling system constructs a corresponding sequence queue based on a dependency relationship between tasks in the 1st data backfilling path: G-table task -> M-table task -> K-table task -> User level table task -> Discount transaction table task -> User discount batch table task -> Y-table. A sequence of the K-table task and the M-table task is not limited in this embodiment. Merely example descriptions are provided here.

[0179] For the 1st data backfilling path and the 2nd data backfilling path shown in FIG. 9, when determining an execution sequence, the data backfilling system can determine that the 1st data backfilling path and the 2nd data backfilling path are executed in parallel, or can determine to execute one of the data backfilling paths and then execute the other one of the data backfilling paths.

[0180] That is, for tasks in different data backfilling paths, the data backfilling system can determine that an execution sequence of the tasks in different data backfilling paths are parallel

execution, or can be sequential execution. For tasks in any data backfilling path, an execution sequence can be determined based on a dependency relationship between the tasks in the task data backfilling path.

[0181] Step 2: Execute the tasks in the minimum task subgraph in the execution sequence, to implement data backfilling.

[0182] Correspondingly, with reference to the above-mentioned example, after determining the execution sequence, the data backfilling system can perform an execution operation on the tasks in the minimum task subgraph based on the execution sequence. When executing a task, the data backfilling system updates a data table corresponding to the task. In this way, after the tasks in the minimum task subgraph are sequentially executed, data in the at least one backfilling start task data table to the at least one backfilling end task data table can be updated, to complete data backfilling from the at least one backfilling start task data table to the at least one backfilling end task data table.

[0183] With reference to the above-mentioned analysis of step 1 and step 2, it can be learned that in this embodiment, the data backfilling system determines the execution sequence based on the dependency relationship, and then executes the corresponding tasks based on the execution sequence, so that the tasks are sequentially executed, to improve validity and reliability of target data backfilling.

[0184] It should be noted that, the above-mentioned examples are merely used to describe examples of a possible implementation of the data backfilling method in this specification, and cannot be understood as a limitation on the implementation of the data backfilling method in this specification. For example, based on the above-mentioned technical concept, some of the above-mentioned technical features can be combined to obtain a new embodiment, a new technical feature can be added based on the above-mentioned example to obtain a new embodiment, some technical features can be reduced based on the above-mentioned example to obtain a new embodiment, some technical features in the above-mentioned example may be replaced with other technical features, or some technical features and orders in the above-mentioned example can be adjusted to obtain a new embodiment. Examples are not listed one by one here.

[0185] According to the above-mentioned technical concept, this specification further provides a computer-readable non-transitory storage medium. The computer-readable non-transitory storage medium stores at least one instruction set, and when the at least one instruction

set is executed by a processor, the steps of the data backfilling method described in this specification are implemented.

[0186] In some possible implementations, aspects of this specification can further be implemented in a form of a program product, including program code. When the program product runs on the data backfilling system 400, the program code is used to enable the data backfilling system 400 to perform the steps of the data backfilling method described in this specification. A program product configured to implement the above-mentioned method can be a portable compact disk read-only memory (CD-ROM), including program code, and can run in the data backfilling system 400. However, the program product in this specification is not limited thereto. In this specification, a readable storage medium can be any tangible medium that includes or stores a program, and the program can be used by or in combination with an instruction execution system. The program product can be any combination of one or more readable media. The readable medium can be a readable signal medium or a readable storage medium. The readable storage medium can be, for example, an electrical system, apparatus, or device, a magnetic system, apparatus, or device, an optical system, apparatus, or device, an electromagnetic system, apparatus, or device, an infrared system, apparatus, or device, or a semiconductor system, apparatus, or device, or any combination thereof. More specific examples of the readable storage medium include: an electrical connection including one or more conducting wires, a portable disk, a hard disk, a random access memory (RAM), a read-only memory (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 component, a magnetic storage component, or any proper combination thereof. The computer-readable storage medium can include a data signal propagated in a baseband or as part of a carrier, and carries readable program code. The propagated data signal can be in a plurality of forms, and includes but is not limited to an electromagnetic signal, an optical signal, or any proper combination thereof. The readable storage medium can further be any readable medium other than the readable storage medium, and the readable medium can be used to send, propagate, or transmit a program used by or in combination with an instruction execution system, apparatus, or component. The program code included in the readable storage medium can be transmitted in any proper medium, including but not limited to a wireless medium, a wired medium, optical cable, an RF medium, or any proper combination thereof. Program code for performing an operation of this specification can be written in any combination of one or more

program design languages. The program design language includes an object-oriented program design language, for example, Java or C++, and further includes a conventional procedural program design language, for example, "C" language, or a similar program design language. The program code can be completely executed on the data backfilling system 400, partially executed on the data backfilling system 400, executed as an independent software package, partially executed on the data backfilling system 400 and partially executed on a remote computing device, or completely executed on the data backfilling system 400.

[0187] It should be noted that, collection, storage, use, processing, transmission, provision, disclosure, etc. of personal information of a user in the technical solutions in this specification all comply with related laws and regulations, and do not violate public order and good morals.

[0188] Specific embodiments of this specification are described above. Other embodiments fall within the scope of the appended claims. In some cases, the actions or steps described in the claims can be performed in an order different from that in the embodiments, and the desired results can still be achieved. In addition, the process depicted in the accompanying drawings does not necessarily need a particular order or consecutive order to achieve the desired results. In some implementations, multitasking and parallel processing are possible or may be advantageous.

[0189] In conclusion, after reading the detailed disclosure, a person skilled in the art can understand that the detailed disclosure can be presented only in an example manner, and can impose no limitation. Although it is not explicitly stated here, a person skilled in the art can understand that the requirements of this specification include various proper changes, improvements, and modifications to the embodiments. These changes, improvements, and modifications are intended to be presented in this specification, and fall within the spirit and scope of the example embodiments of this specification.

[0190] In addition, some terms in this specification are used to describe the embodiments of this specification. For example, "an embodiment", "embodiments", and/or "some embodiments" mean that a particular feature, structure, or feature described with reference to the embodiments can be included in at least one embodiment of this specification. Therefore, it can be emphasized and understood that two or more of "embodiments", "one embodiment", or "alternative embodiments" in each part of this specification do not necessarily indicate the same embodiment. In addition, specific features, structures, or features can be properly combined in one or more embodiments of this specification.

[0191] It should be understood that in the above-mentioned descriptions of the embodiments of this specification, to help understand a feature, for the purpose of simplifying this specification, various features are combined in a single embodiment, accompanying drawing, or description thereof in this specification. However, this does not mean that the combination of these features is necessary. It is entirely possible for a person skilled in the art to mark some devices and understand the devices as separate embodiments when reading this specification. That is, the embodiments in this specification can also be understood as an integration of a plurality of secondary embodiments. Content of each secondary embodiment is also true when features are less than all features in one of the above-mentioned disclosed embodiments.

[0192] Each patent, patent application, publication of a patent application, and other materials, for example, articles, books, instructions, publications, documents, and products, that are cited in this specification and that are other than any document that is inconsistent or conflict with this document or that imposes limiting impact on the widest scope of the claims can be incorporated here by reference and used for all purposes now or later associated with this document. In addition, the term in this document shall prevail when there is any inconsistency or conflict between the description, definition and/or use of a related term in any material and the description, definition and/or use of a related terms in this document.

[0193] Finally, it should be understood that the principles of the implementation solutions of this specification are described in the implementation solutions of the application disclosed in this specification. Other modified embodiments also fall within the scope of this specification. Therefore, the embodiments disclosed in this specification are merely examples and not limitations. A person skilled in the art can use an alternative configuration according to the embodiments of this specification to implement the application in this specification. Therefore, the embodiments of this specification are not limited to the embodiments accurately described in the application.

CLAIMS

What is claimed is:

1. A data backfilling method, comprising:

obtaining a data backfilling request, wherein the data backfilling request is used to indicate to perform data backfilling from at least one backfilling start task data table to at least one backfilling end task data table;

determining an end task set and a start task set in a pre-built task graph based on the data backfilling request, wherein the task graph is used to represent a plurality of tasks and a dependency relationship between the plurality of tasks, the end task set comprises at least one backfilling end task in the plurality of tasks that is used to generate the at least one backfilling end task data table, and the start task set comprises at least one backfilling start task in the plurality of tasks that is used to generate the at least one backfilling start task data table;

determining, in the task graph, a minimum task subgraph comprising the end task set and the start task set; and

executing each task in the minimum task subgraph, to implement data backfilling.

2. The method according to claim 1, wherein the determining, in the task graph, a minimum task subgraph comprising the end task set and the start task set comprises:

determining an end task in the task graph based on the end task set, and determining a start task in the task graph based on the start task set; and

determining the minimum task subgraph in the task graph based on the start task and the end task.

3. The method according to claim 2, wherein the determining an end task in the task graph based on the end task set comprises one of the following:

if there is one backfilling end task in the end task set, determining the backfilling end task in the end task set as the end task; or

if there are a plurality of backfilling end tasks in the end task set, generating a backfilling end virtual task in the task graph, wherein the backfilling end virtual task depends on each backfilling end task in the end task set, and determining the backfilling end virtual task as the end task; or

if there are a plurality of backfilling end tasks in the end task set, separately determining each backfilling end task in the end task set as one end task.

4. The method according to claim 2, wherein the determining a start task in the task graph based on the start task set comprises one of the following:

if there is one backfilling start task in the start task set, determining the backfilling start task in the start task set as the start task; or

if there are a plurality of backfilling start tasks in the start task set, generating a backfilling start virtual task in the task graph, wherein each backfilling start task in the start task set depends on the backfilling start virtual task, and determining the backfilling start virtual task as the start task; or

if there are a plurality of backfilling start tasks in the start task set, separately determining each backfilling start task in the start task set as one start task.

5. The method according to claim 2, wherein there is one start task, there is one end task, and the determining the minimum task subgraph in the task graph based on the start task and the end task comprises:

performing a path search in the task graph by using the start task as a path start and using the end task as a path end, to obtain at least one data backfilling path; and

determining the minimum task subgraph based on the at least one data backfilling path.

6. The method according to claim 2, wherein there are N start tasks, N is an integer greater than 1, there is one end task, and the determining the minimum task subgraph in the task graph based on the start task and the end task comprises:

performing a path search in the task graph by using the 1st start task as a path start and using the end task as a path end, to obtain at least one data backfilling path corresponding to the 1st start task, and generating an intermediate task subgraph based on the at least one data backfilling path corresponding to the 1st start task;

for the ith start task, wherein a value of i is successively an integer from 2 to N, separately performing the following operations: performing a path search in the task graph by using the ith start task as a path start and using the end task as a path end, to obtain at least one data backfilling path corresponding to the ith start task, and updating the intermediate task subgraph based on the at least one data backfilling path corresponding to the ith start task; and

determining an updated intermediate task subgraph as the minimum task subgraph.

7. The method according to claim 1, wherein the executing each task in the minimum task subgraph, to implement data backfilling comprises:

determining an execution sequence of the tasks in the minimum task subgraph based on a dependency relationship between the tasks in the minimum task subgraph; and

executing the tasks in the minimum task subgraph in the execution sequence, to implement data backfilling.

8. The method according to claim 1, wherein the task graph is a directed acyclic graph, and the directed acyclic graph comprises a plurality of nodes and a plurality of directed edges; and

one node corresponds to one task, and a directed edge connecting two nodes is used to represent a dependency relationship between tasks corresponding to the two nodes.

9. A data backfilling system, comprising:

at least one storage medium, storing at least one instruction set to perform task processing; and

at least one processor, communicatively connected to the at least one storage medium, wherein when running, the at least one processor reads the at least one instruction set, and performs the following operations based on an indication of the at least one instruction set:

obtaining a data backfilling request, wherein the data backfilling request is used to indicate to perform data backfilling from at least one backfilling start task data table to at least one backfilling end task data table;

determining an end task set and a start task set in a pre-built task graph based on the data backfilling request, wherein the task graph is used to represent a plurality of tasks and a dependency relationship between the plurality of tasks, the end task set comprises at least one backfilling end task in the plurality of tasks that is used to generate the at least one backfilling end task data table, and the start task set comprises at least one backfilling start task in the plurality of tasks that is used to generate the at least one backfilling start task data table;

determining, in the task graph, a minimum task subgraph comprising the end task set and the start task set; and

executing each task in the minimum task subgraph, to implement data backfilling.

10. The system according to claim 9, wherein to determine, in the task graph, the minimum task subgraph comprising the end task set and the start task set, the at least one processor performs the following operations:

determining an end task in the task graph based on the end task set, and determining a start task in the task graph based on the start task set; and

determining the minimum task subgraph in the task graph based on the start task and the end task.

11. The system according to claim 10, wherein to determine the end task in the task graph based on the end task set, the at least one processor performs one of the following:

if there is one backfilling end task in the end task set, determining the backfilling end task in the end task set as the end task; or

if there are a plurality of backfilling end tasks in the end task set, generating a backfilling end virtual task in the task graph, wherein the backfilling end virtual task depends on each backfilling end task in the end task set, and determining the backfilling end virtual task as the end task; or

if there are a plurality of backfilling end tasks in the end task set, separately determining each backfilling end task in the end task set as one end task.

12. The system according to claim 10, wherein to determine the start task in the task graph based on the start task set, the at least one processor performs one of the following:

if there is one backfilling start task in the start task set, determining the backfilling start task in the start task set as the start task; or

if there are a plurality of backfilling start tasks in the start task set, generating a backfilling start virtual task in the task graph, wherein each backfilling start task in the start task set depends on the backfilling start virtual task, and determining the backfilling start virtual task as the start task; or

if there are a plurality of backfilling start tasks in the start task set, separately determining each backfilling start task in the start task set as one start task.

13. The system according to claim 10, wherein there is one start task, there is one end task, and to determine the minimum task subgraph in the task graph based on the start task and the end task, the at least one processor performs the following operations:

performing a path search in the task graph by using the start task as a path start and using the end task as a path end, to obtain at least one data backfilling path; and

determining the minimum task subgraph based on the at least one data backfilling path.

14. The system according to claim 10, wherein there are N start tasks, N is an integer greater than 1, there is one end task, and to determine the minimum task subgraph in the task graph based on the start task and the end task, the at least one processor performs the following operations:

performing a path search in the task graph by using the 1st start task as a path start and using

the end task as a path end, to obtain at least one data backfilling path corresponding to the 1st start task, and generating an intermediate task subgraph based on the at least one data backfilling path corresponding to the 1st start task;

for the ith start task, wherein a value of i is successively an integer from 2 to N, separately performing the following operations: performing a path search in the task graph by using the ith start task as a path start and using the end task as a path end, to obtain at least one data backfilling path corresponding to the ith start task, and updating the intermediate task subgraph based on the at least one data backfilling path corresponding to the ith start task; and

determining an updated intermediate task subgraph as the minimum task subgraph.

15. The system according to claim 9, wherein to execute each task in the minimum task subgraph, to implement data backfilling, the at least one processor performs the following operations:

determining an execution sequence of the tasks in the minimum task subgraph based on a dependency relationship between the tasks in the minimum task subgraph; and

executing the tasks in the minimum task subgraph in the execution sequence, to implement data backfilling.

16. A computer-readable non-transitory storage medium, wherein the computer-readable non-transitory storage medium stores at least one instruction set, and when the at least one instruction set is executed by at least one processor, the method according to any one of claims 1 to 8 is implemented.