US 20160085794A1

(19) United States

(12) Patent Application Publication

Bengali et al.

(10) Pub. No.: US 2016/0085794 A1

(43) Pub. Date: Mar. 24, 2016

(54) DATA CONSISTENCY AND ROLLBACK FOR CLOUD ANALYTICS

(71) Applicant: **Dell Products L.P.**, Round Rock, TX (US)

(72) Inventors: **Ketan Bengali**, Sunnyvale, CA (US); **Kaniska Mandal**, Sunnyvale, CA (US); **Alex J. Chen**, Fremont, CA (US)

(21) Appl. No.: **14/862,007**

(22) Filed: **Sep. 22, 2015**

Related U.S. Application Data

(63) Continuation of application No. 13/764,446, filed on Feb. 11, 2013, now Pat. No. 9,141,680.

Publication Classification

(51) **Int. Cl.**

G06F 17/30 (2006.01)

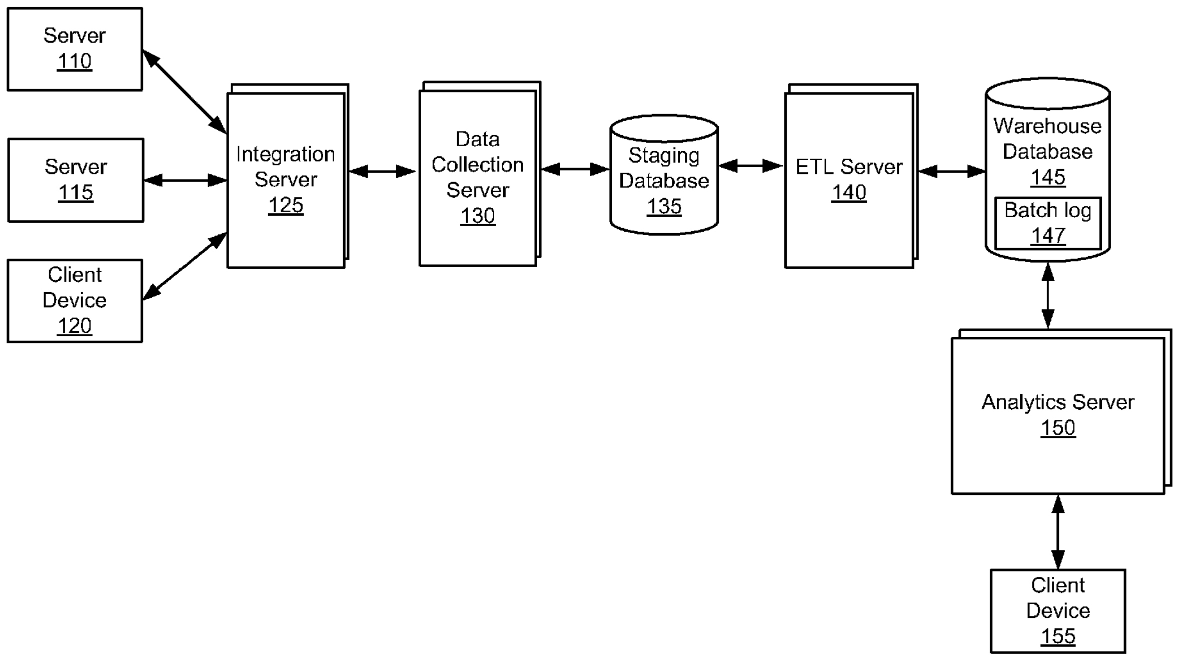
G06F 11/14 (2006.01)

(52) **U.S. Cl.**

CPC ***G06F 17/30371*** (2013.01); ***G06F 17/30563*** (2013.01); ***G06F 11/1469*** (2013.01); *G06F 2201/80* (2013.01)

(57) **ABSTRACT**

An extract-transform-load (ETL) platform fetches consistent datasets in a batch for a given period of time and provides the ability to rollback that batch. The batch may be fetched for an interval of time, and the ETL platform may fetch new or changed data from different cloud/on-premise applications. It will store this data in the cloud or on-premise to build data history. As the ETL platform fetches new data, the system will not overwrite existing data, but rather will create new versions so that change history is preserved. For any reason, if businesses would like to rollback data, they could rollback to any previous batch.



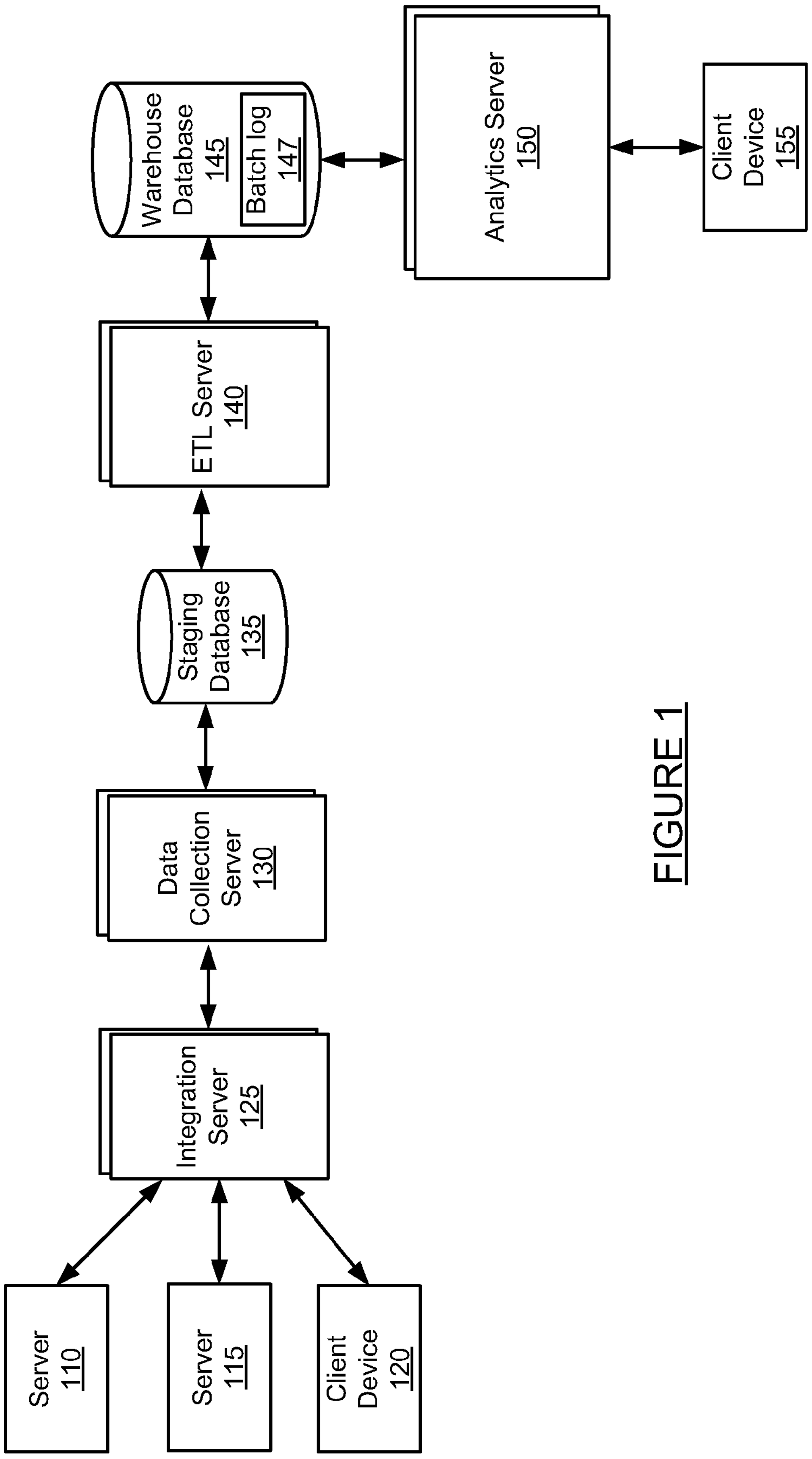

Server 110
Server 115
Client Device 120
Integration Server 125
Data Collection Server 130
Staging Database 135
ETL Server 140
Warehouse Database 145
Batch log 147
Analytics Server 150
Client Device 155

FIGURE 1

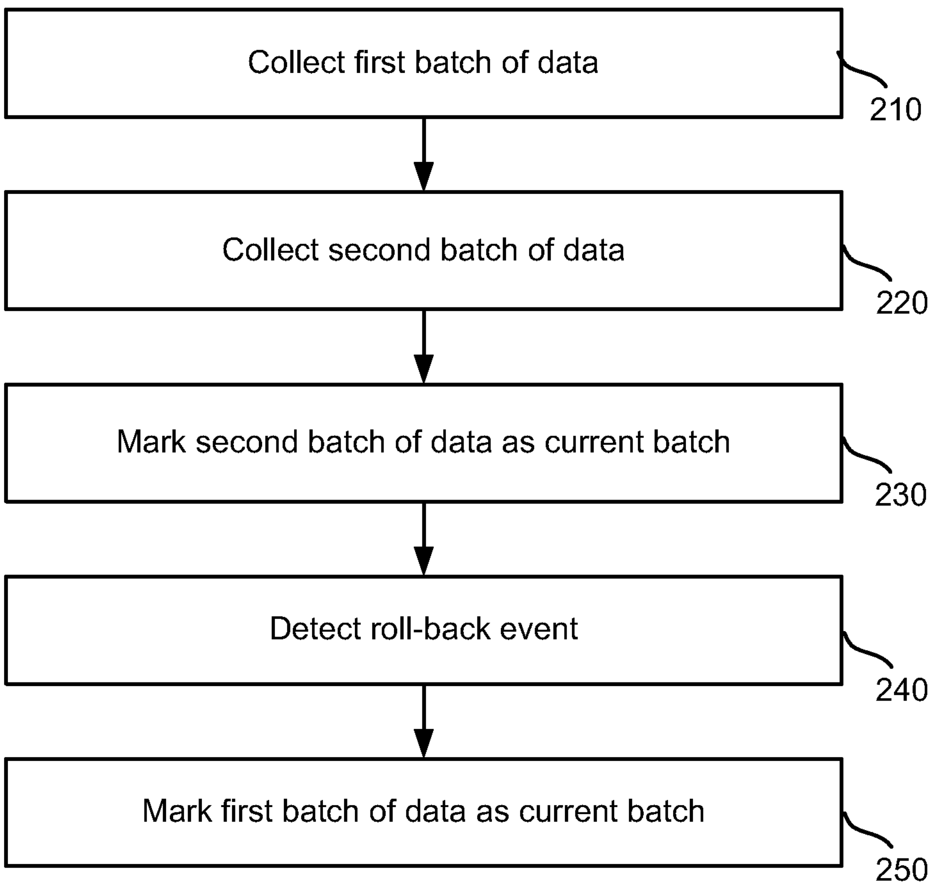

Collect first batch of data
210
Collect second batch of data
220
Mark second batch of data as current batch
230
Detect roll-back event
240
Mark first batch of data as current batch
250

FIGURE 2

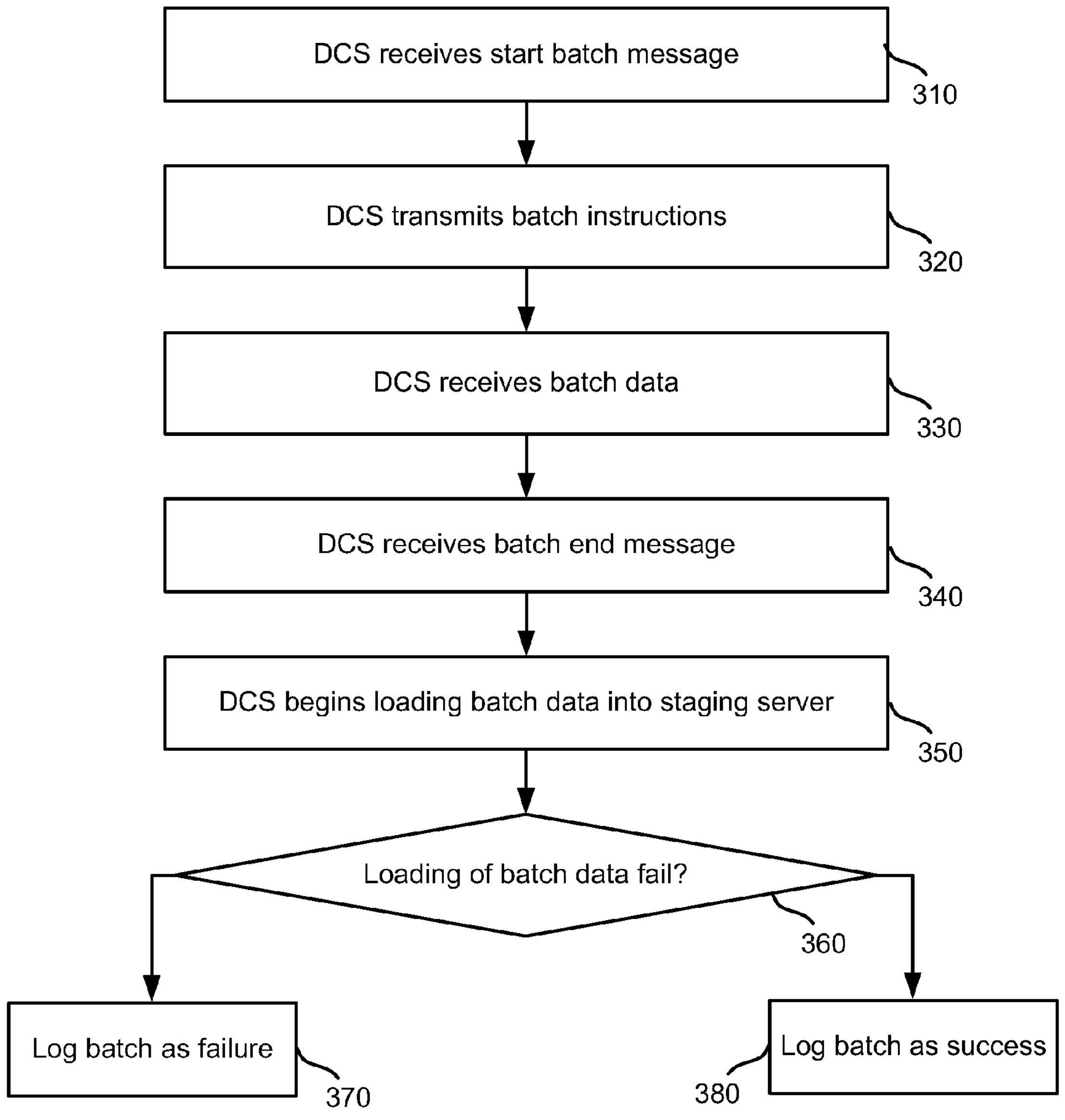

DCS receives start batch message
310
DCS transmits batch instructions
320
DCS receives batch data
330
DCS receives batch end message
340
DCS begins loading batch data into staging server
350
Loading of batch data fail?
360
Log batch as failure
370
Log batch as success
380

FIGURE 3

Before/After	Key	Amount	Start	End	Batch ID	Current
Before Chg	~~1~~	~~500~~	~~1/1/1900~~	~~12/31/2099~~	~~1~~	~~Y~~
After Chg	1	500	1/1/1900	7/31/2012	1	N
After Chg	1	1000	8/1/2012	12/31/2099	2	Y

FIGURE 4A

Before/After	Key	Amount	Start	End	Batch ID	Current
Before Rllbk	~~1~~	~~500~~	~~1/1/1900~~	~~7/31/2099~~	~~1~~	~~N~~
After Rllbk	~~1~~	~~1000~~	~~8/1/2012~~	~~12/31/2099~~	~~2~~	~~Y~~
After Rllbk	1	500	1/1/1900	12/31/2099	1	Y

FIGURE 4B

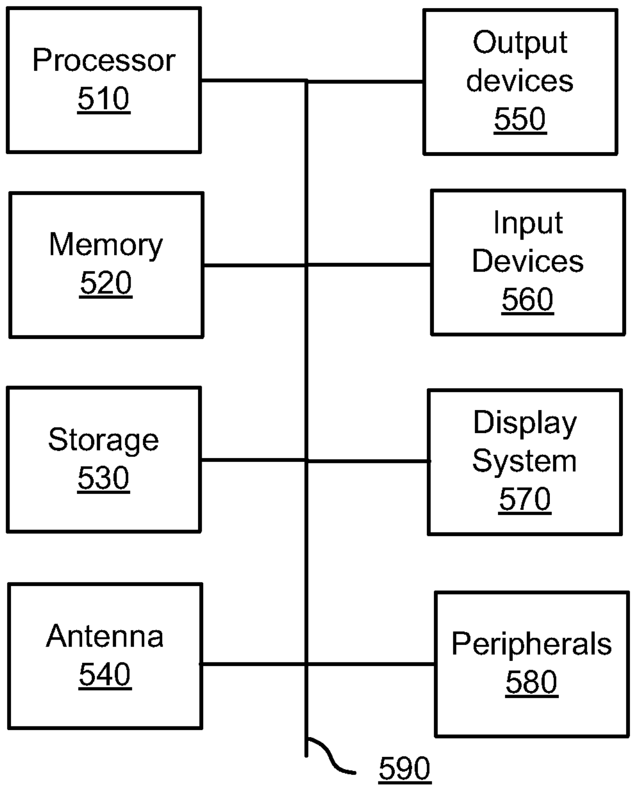

Processor
510
Memory
520
Storage
530
Antenna
540
Output devices
550
Input Devices
560
Display System
570
Peripherals
580
590

FIGURE 5

DATA CONSISTENCY AND ROLLBACK FOR CLOUD ANALYTICS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is a continuation and claims the priority benefit of U.S. patent application Ser. No. 13/764,446 filed Feb. 11, 2013, now U.S. Pat. No. 9,141,680, the disclosure of which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

[0002] Businesses must process large amounts of data to make strategic decisions and be successful. The data is often provided in formats such as reports. To build a meaningful report, businesses are relying on multi-tenanted software as a service (SAAS) analytic companies. Building and providing meaningful analytics typically require a large amount of resources and have a high cost.

[0003] In order to reduce cost, more and more businesses are adapting to cloud based SAAS application models. For example, businesses may store sales data in “Salesforce” applications, accounting data in “NetSuite” applications, and billing data in “Zuora” applications. It is important to have detailed information about a company’s performance and positions. Unfortunately, analytic applications do not consolidate data from different SAAS applications and provide a single view. Analytic applications also do not provide data consistency within data collection of different data types. What is needed is an improved analytics system that improves upon the analytic systems of the prior art.

SUMMARY OF THE INVENTION

[0004] The present system includes an extract-transform-load (ETL) platform which fetches consistent datasets in a batch for a given period of time and provides the ability to rollback that batch. The batch may be fetched for an interval of time, and the ETL platform may fetch new or changed data from different cloud/on-premise applications. It will store this data in the cloud or on-premise to build data history. As the ETL platform fetches new data, the system will not overwrite existing data, but rather will create new versions so that change history is preserved. For any reason, if businesses would like to rollback data, they could rollback to any previous batch.

[0005] In an embodiment, a method for collecting data may include collecting a first batch of data by a server from one or more tenant applications and associated with a first period of time. A second batch of data may be collected by the server from the one or more tenant applications and associated with a second period of time subsequent to the first period of time. The second batch of data may be marked as the current batch of data. A rollback event may be detected, and the first batch of data may be marked as the current batch of data after the rollback request.

[0006] In an embodiment, a system for collecting data may include a memory, a processor and one or more modules stored in memory and executable by the processor. The modules may be executable to collect a first batch of data from one or more tenant applications and associated with a first period of time, collect a second batch of data from the one or more tenant applications and associated with a second period of time subsequent to the first period of time, and mark the second batch of data as the current batch of data. The modules may further be executable to detect a rollback event and mark the first batch of data as the current batch of data after the rollback request.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] FIG. **1** is a block diagram of an exemplary data analytics system.

[0008] FIG. **2** is an exemplary method for performing a rollback of data.

[0009] FIG. **3** is an exemplary method for collecting a batch of data.

[0010] FIG. **4**A is an exemplary batch log with a data change.

[0011] FIG. **4**B is an exemplary batch log with a roll back.

[0012] FIG. **5** is a block diagram of a device for implementing the present technology.

DETAILED DESCRIPTION

[0013] The present system includes an ETL platform which fetches consistent datasets in a batch for a given period of time and provides the ability to rollback that batch. The batch may be fetched for an interval of time, and the ETL platform may fetch new or changed data from different cloud/on-premise applications. It will store this data in the cloud or on-premise to build data history. As the ETL platform fetches new data, it will not overwrite existing data, but rather will create new versions so that change history is preserved. For any reason, if businesses would like to rollback data, they could rollback to any previous batch.

[0014] The present system has many advantages over the prior art. Though some prior art SAAS analytic providers provide analytics, these systems don’t provide history tracking or snapshot trending features. Building analytics with history tracking or snapshot trending feature requires complex ETL and rollback features. Once data is fetched and placed in an SAAS analytics system and a hardware failure occurs at the SAAS application vendor, if SAAS application vendor rolls back its system to some previous point-in-time, data in the SAAS analytic system will become inconsistent with the SAAS application. In this case, the only possible fix is to erase everything in SAAS analytic, re-provision data from SAAS application and erase all history.

[0015] Data consistency is also an issue for previous SAAS analytics systems. All SAAS applications provide an API to access their data. However fetching data using their supplied APIs doesn’t provide data consistency. For example, when fetching data from Salesforce, it is possible that the process is fetching all accounts at time T1, and then fetching all opportunities at time T10. Fetching all opportunities at time T10 would result in getting new Opportunities created between time T1 and T10 whose account information is not fetched yet. It may result in inconsistent dataset as receiving an opportunity without account information may produce wrong report output.

[0016] When the present system restores itself from any failure (Business application/Application integrator/Data Collector/ETL system); the ETL system can automatically restart the data extraction process as it continuously polls a data collection status and looks up for newer successful batches of data based on timestamps. The ETL system can replay the data extraction process from anytime in the past, without any manual intervention required. This auto-restorability and auto-restartability features ensure data consistency. This system ensures complete data isolation for multiple tenants and guarantees consistent delivery to heterogeneous persistence data stores.

[0017] FIG. **1** is a block diagram of an exemplary data analytics system. The system of FIG. **1** includes tenant servers **110** and **115**, tenant clime **120**, integration server **125**, data collection server (DCS) **130**, staging database **135**, extract-transform-load (ETL) server **140**, warehouse database **145**, analytics server **150** and client device **150**. Each of devices **110-155** may communicate with each other over a network (not shown). The network may be implemented as a private network, public network, Wi-Fi network, WAN, LAN, an intranet, the Internet, a cellular network, or a combination of these networks.

[0018] Servers **110** and **115** and client device **120** may each be associated with a tenant (client organization) in a multi-tenancy. Each tenant of the multi-tenancy may include one or more servers and client devices. Each server and client may include data to be collected by data collection server **130** via integration server **125**. In embodiments, integration server **125** may communicate with different SAAS providers, whether provided from a cloud or a particular machine, and communicate with data collection server **130**. Client **120** may be implemented as a desktop, laptop, notebook, tablet computer, smart phone, or some other computing device.

[0019] Data collection server **130** may collect data from one or more tenant applications on devices **110-120** through integration server **125** and store the data in a staging database **135**. The Data collection server may send batch instructions to integration server **125** in response to receiving a start batch request. Data collection server may provide any portion of the staging data to ETL server **140**, for example upon receiving a data request from ETL server **140**. When data is collected, it is stored and maintained. Batches of data are not overwritten with newly collected data.

[0020] ETL server **140** receives staging data from data collection server **130** and may transform the data to a format more useful to a user. For example, the data transformation may include selecting only certain columns to load into a star format, translating coded values, deriving new calculated values, sorting data, aggregating data, transposing or pivoting data, splitting a column into multiple columns, and other processing. Once data is transformed by ETL server **140**, it may be provided to data warehouse **155** for future analytics.

[0021] Warehouse database **145** may receive transformed data from ETL server **140** and provide the data to analytics server **150** for processing. When data is loaded into warehouse database **145**, the data is stored in a star schema and maintained. Transformed data is not overwritten in warehouse database **145**. This allows rollback to previous batches of data if needed. A batch log **147** may be stored at warehouse database **147**. The batch log may be updated and maintained to track information about each batch of data and may be used in the rollback of data. The batch log may be stored in table format and may include attributes for each batch such as batch ID, tenant ID, data start date and time, data end date and time, DCS processing status, staging database ID, ETL processing status, and ETL server ID. The DCS processing status and ETL processing status may include not started, in-progress, success and failure. The batch log may be updated by ETL server **140**, data collection server **130**, and other servers of the system of FIG. **1**. Though illustrated as being stored in warehouse database **145**, batch log **147** may be stored on another serer or database within the system of FIG. **1**.

[0022] Analytics server **150** may retrieve transformed data stored in a star schema in data warehouse **155** and perform analytics to the data. The results of the analytics may be provided in the form of charts, graphs, reports or other formats to a user at client device **170**.

[0023] Though illustrated as one server or one device, each of the servers and clients of the system of FIG. **1** may be implemented using one or more actual or logical machines, servers and client devices. One or more blocks of the system of FIG. **1** may also be combined. Further, though examples of communications are shown using arrows, it is intended and should be understood that each of the servers and clients in the system of FIG. **1** may communicate over network, and therefore with each other.

[0024] FIG. **2** is an exemplary method for performing a rollback of data. A first batch of data is collected at step **210**. The batch of data may be collected for a period of time by data collection serve **130** from tenant devices **110-120** via integration server **125**. A batch log may be updated during and after the batch data collection, and the collected data is marked with the batch ID information. Collecting a first batch of data is discussed in more detail below with respect to the method of FIG. **3**.

[0025] After a batch of data is collected, the batch may be stored or "staged" at staging database **135**. Once staged, the batch may be transformed at ETL server **140**. ETL server **140** performs transformation functions per batch. The transformed batch of data is then provided to warehouse database **145**. Analytics server **150** may access transformed data at warehouse database **145** to generate charts, graphs, reports and other tools for analyzing the transformed data.

[0026] A second batch of data is collected at step **220**. The second batch of data may include the same data objects as the first batch (sales information, opportunity information, and so forth), but will cover a different period of time. In some embodiments, the second batch will automatically include data with a start time just after the end time of the previous successful batch. Collecting a second batch of data is performed as described with respect to FIG. **3**. The second batch of data may be staged in staging database **135** without overwriting or deleting the first batch or any other batch. Similarly, when the second batch is transformed and stored in warehouse database **145**, no other data is overwritten or deleted when the second batch of data is stored.

[0027] The second batch is marked as the current batch at step **230**. The batch may be marked as the current batch in the batch log **147**. The second batch is the most up to date batch and will likely be used for performing analytics. An example of a batch log having a second batch marked as the current batch is provided in FIG. **4**A.

[0028] A rollback event is detected at step **240**. A rollback event may include receiving input from a user that a rollback should be performed. For example, an administrator may determine that the most current batch should not be used, and may request a rollback to the previous batch. A rollback event may also include an automated event not initiated from a user. For example, if a data load into warehouse database **145** fails, a rollback event may be automatically triggered by the failure.

[0029] Upon detecting a rollback event, a first batch (i.e., previous successful batch) is designated as the current batch at step **250**. In some embodiments, the current batch information is deleted from the batch log **147** and the previous batch is marked as the current batch. An example of a batch log reflecting a rollback is illustrated in FIG. **4**B.

[0030] FIG. **3** is an exemplary method for collecting a batch of data. The DCS **130** receives a start batch message from integration server **125** at step **310**. The start batch message may be received periodically or initiated by the integration serer **125** in response to a user request.

[0031] In response to the request, the DCS **130** transmits batch instructions to integration server **125** at step **320**. The batch instructions may indicate the data start time and date, data end time and date, the data to be collected, and the batch ID. For example, the batch instructions may indicate to collect employee records, sales records, and revenue records created or changed during a time period of Jan. 1, 2013 at 8:00 AM to Jan. 1, 2013 at 10:00 AM, and to call the data batch no. 001. The batch log may be updated by DCS **130** to indicate the batch ID and that DCS processing of the batch is “not started.”

[0032] DCS **130** receives batch data at step **330**. In some embodiments, DCS **130** may receive all batch data requested, a portion of the data, or none of the data. While data is received from integration server **125** by DCS **130**, the DCS processing status may indicate “in-progress.” Once the batch data has been provided to DCS server **130**, integration server **125** provides a batch end message to DCS **130** at step **340**. The request for a batch of data may specify that all new data and changed data maintained by a tenant be collected. If no tenant data has changed or been updated for the specified period of time, no data will be provided and no new batch is created.

[0033] DCS sever **130** may stage the collected data for the batch in staging database **135** at step **350**. A determination is then made by DCS **130** if the batch data staging has failed or succeeded. The batch data staging is marked as “successful” in batch log **147** at step **380** if all batch data received by DCS **130** is staged or loaded into staging database **135**. If any portion of the batch data is not loaded into staging database **135**, the batch status is set to “failure” at step **370**. If a batch is listed as a failure, the batch is removed from the batch log and the next batch will attempt to collect the same data for the same time period. In some embodiments, the batch log may be updated by script generated and executed by DCS **130**, ETL **135** or other parts of the system of FIG. **1**.

[0034] FIG. **4A** is an exemplary batch log with a data change. The batch table of FIG. **4A** include seven columns with headings of “Before/After”, “Key”, “Amount”, “Start”, “End”, “Batch ID”, and “Current.” The Key through current record columns may be added to all data stored in staging database **135** and warehouse database **145**. In the example of FIG. **4A**, an opportunity in the batch data has changed from $500 to $1,000. In the original batch collection, the key has a value of 1, the amount of the opportunity is 500, the batch data starts at Jan. 1, 1900 and ends at Dec. 31, 2099, the data has a batch ID of 1 and is marked as the current data.

[0035] After a change that occurs on Aug. 1, 2012 is detected, the original batch of row 1 is replaced (hence, the strikeout of the data in row 1) and is replaced with two batches, as indicated in the second row and third row of data in the log. The second row of data indicates that the business key is 1, the amount is 500, the data begins on Jan. 1, 1900 and ends at Jul. 31, 2012, the batch ID is 1 and that the batch is not the current record. The third column indicates a business key of 1, an amount of 1000, a start date of Aug. 1, 2012, an end date of Dec. 31, 2099, a batch ID of 2 and that the batch is the current record.

[0036] FIG. **4B** is an exemplary batch log with a roll back. The columns in FIG. **4B** have the same headings as those in the batch log of FIG. **4A**. FIG. **4B** illustrates a batch log in the case of a rollback that causes the current record to be changed to batch ID 1 from batch ID 2. The first two rows of the batch log in FIG. **4B** match the last two rows of the batch log from FIG. **4A**. Both rows are replaced (hence, the strikethrough), with the first row from FIG. **4A**, which is now made the current record.

[0037] FIG. **5** is a block diagram of a device for implementing the present technology. FIG. **5** illustrates an exemplary computing system **500** that may be used to implement a computing device for use with the present technology. System **500** of FIG. **5** may be implemented in the contexts of the likes includes tenant servers **110** and **115**, tenant clime **120**, integration server **125**, DCS **130**, staging database **135**, ETL server **140**, warehouse database **145**, analytics server **150** and client device **150**. The computing system **500** of FIG. **5** includes one or more processors **510** and memory **520**. Main memory **520** may store, in part, instructions and data for execution by processor **510**. Main memory can store the executable code when in operation. The system **500** of FIG. **5** further includes a storage **520**, which may include mass storage and portable storage, antenna **540**, output devices **550**, user input devices **560**, a display system **570**, and peripheral devices **580**.

[0038] The components shown in FIG. **5** are depicted as being connected via a single bus **590**. However, the components may be connected through one or more data transport means. For example, processor unit **510** and main memory **520** may be connected via a local microprocessor bus, and the storage **530**, peripheral device(s) **580** and display system **570** may be connected via one or more input/output (I/O) buses.

[0039] Storage device **530**, which may include mass storage implemented with a magnetic disk drive or an optical disk drive, may be a non-volatile storage device for storing data and instructions for use by processor unit **510**. Storage device **530** can store the system software for implementing embodiments of the present invention for purposes of loading that software into main memory **510**.

[0040] Portable storage device of storage **530** operates in conjunction with a portable non-volatile storage medium, such as a floppy disk, compact disk or Digital video disc, to input and output data and code to and from the computer system **500** of FIG. **5**. The system software for implementing embodiments of the present invention may be stored on such a portable medium and input to the computer system **500** via the portable storage device.

[0041] Antenna **540** may include one or more antennas for communicating wirelessly with another device. Antenna **516** may be used, for example, to communicate wirelessly via Wi-Fi, Bluetooth, with a cellular network, or with other wireless protocols and systems. The one or more antennas may be controlled by a processor **510**, which may include a controller, to transmit and receive wireless signals. For example, processor **510** execute programs stored in memory **512** to control antenna **540** transmit a wireless signal to a cellular network and receive a wireless signal from a cellular network.

[0042] The system **500** as shown in FIG. **5** includes output devices **550** and input device **560**. Examples of suitable output devices include speakers, printers, network interfaces, and monitors. Input devices **560** may include a touch screen, microphone, accelerometers, a camera, and other device. Input devices **560** may include an alpha-numeric keypad, such as a keyboard, for inputting alpha-numeric and other information, or a pointing device, such as a mouse, a trackball, stylus, or cursor direction keys.

[0043] Display system **570** may include a liquid crystal display (LCD), LED display, or other suitable display device. Display system **570** receives textual and graphical information, and processes the information for output to the display device.

[0044] Peripherals **580** may include any type of computer support device to add additional functionality to the computer system. For example, peripheral device(s) **580** may include a modem or a router.

[0045] The components contained in the computer system **500** of FIG. **5** are those typically found in computing system, such as but not limited to a desk top computer, lap top computer, notebook computer, net book computer, tablet computer, smart phone, personal data assistant (PDA), or other computer that may be suitable for use with embodiments of the present invention and are intended to represent a broad category of such computer components that are well known in the art. Thus, the computer system **500** of FIG. **5** can be a personal computer, hand held computing device, telephone, mobile computing device, workstation, server, minicomputer, mainframe computer, or any other computing device. The computer can also include different bus configurations, networked platforms, multi-processor platforms, etc. Various operating systems can be used including Unix, Linux, Windows, Macintosh OS, Palm OS, and other suitable operating systems.

[0046] The foregoing detailed description of the technology herein has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the technology to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. The described embodiments were chosen in order to best explain the principles of the technology and its practical application to thereby enable others skilled in the art to best utilize the technology in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the technology be defined by the claims appended hereto.

1. (canceled)

2. A system for retrieving consistent datasets, comprising:

a staging database for storing batches of data corresponding to a period of time, wherein a batch of data includes one or more distinct datasets; and

a plurality of tenant devices, wherein each tenant device of the plurality of tenant devices includes a processor that executes instructions stored in memory to:

collect a current batch of data associated with a first period of time from one or more sources, wherein collection of the first batch of data includes instructions to collect new or changed data compared to a marked current batch of data that has been previously stored in memory,

assign identification information for the collected current batch of data, wherein the collected current collected current batch of data includes at least one new or changed dataset compared to the marked current batch of data that is previously stored in a batch log,

store the collected current batch of data in the staging database, wherein the stored current batch of data does not overwrite previously stored batches of data listed in the batch log, and wherein a location associated with the stored current batch of data is updated in the batch log,

mark the collected current batch of data as the current batch of data in the batch log,

detect a rollback event, wherein the rollback event indicates that the marked current batch of data should not be used,

select a previously stored batch of data as the current batch using the batch log, wherein information for the previously stored batch of data is included in the batch log,

retrieve the previously stored batch of data from the staging database using the identification information in the batch log, wherein the retrieved previously stored batch of data is used to overwrite the first batch of data, and

deleting information pertaining to the first batch of data from the batch log.

3. The system of claim **2**, wherein the collected batches of data from the one or more sources corresponds with one or more applications associated with the tenant.

4. The system of claim **2**, wherein the current batch of data and the previously stored batches of data include the same data objects.

5. The system of claim **2**, wherein the detected rollback event corresponds to an automated event.

6. The system of claim **5**, wherein the automated event includes a generated failure output if a batch of data cannot be stored in memory.

7. The system of claim **2**, wherein collection of the current batch of data from the one or more sources includes instructions that indicates the period of time associated with the data to be collected and what data is to be collected.

8. The system of claim **2**, wherein collection of the current batch of data from the one or more sources includes updating the batch log with a status associated with the collection.

9. The system of claim **8**, wherein the collection status updated in the batch log includes “not started”, “in-progress”, “successful”, and “failure.”

* * * * *