(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(54) Title: SYSTEM AND METHOD FOR MAINTAINING LARGE-GRAINED DATABASE CONCURRENCY WITH A LOG MONITOR INCORPORATING DYNAMICALLY REDEFINABLE BUSINESS LOGIC
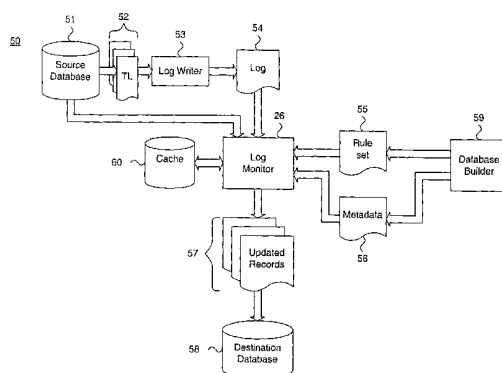
(57) Abstract: A system (10) and method (150) for maintaining large-grained database concurrency with a log monitor (26) incorporating dynamically redefinable business logic (94) are described. Operations expressed in a data manipulation language are executed against a source database (51). At least one operation constitutes a commit operation that completes each database transaction. A current rule set (55) is defined. Each rule includes business logic (94) specifying a data selection criteria for records stored in the stored in the source database. A log entry (70) is periodically generated in a log (54) for each transaction committed to the source database (51). Each log entry (70) identifies an affected record and includes transactional data. The transaction identified in each log entry (70) is evaluated against the data selection criteria specified in the current rule set (55). A new record (57) is built in accordance with metadata (56) describing a destination database (58). The new record (57) contains select transactional data from the log entry (70) of each transaction meeting the selection criteria. The new record (57) is stored into the destination database(58). The data stored in the destination database (58) includes at least a partial subset of the source database (51).

5   SYSTEM AND METHOD FOR MAINTAINING LARGE-GRAINED DATABASE
         CONCURRENCY WITH A LOG MONITOR INCORPORATING DYNAMICALLY
                         REDEFINABLE BUSINESS LOGIC
                  <u>CROSS-REFERENCE TO RELATED APPLICATION</u>

         This patent application claims priority under 35 U.S.C. § 119(e) to U.S. patent application

10   Serial No. 09/804,672, filed March 9, 2001, pending, the disclosure of which is incorporated by

     reference.

                               <u>TECHNICAL FIELD</u>

         The present invention relates in general to database object extraction and, in particular, to

     a system and method for maintaining large-grained database concurrency with a log monitor

15   incorporating dynamically redefinable business logic.

                          <u>BACKGROUND OF THE INVENTION</u>

         Presently, corporate database management systems fall into two categories: production

     and informational. Production databases, including operational data stores, function as

     repositories for real-time or near-real-time data generated by or used in the operation of

20   manufacturing, production, and transactional systems. In contrast, informational databases store

     data periodically obtained from production databases for use in decision support and on-line

     analytical processing systems. Informational databases include data warehouses, often structured

     as enterprise databases and datamarts.

         Typically, data warehouses store both informational data and metadata that describe the

25   database structure. At a minimum, informational databases must maintain a degree of large-

     grained data concurrency with the data stored in the production databases for trend analyses and

     data derivation.

         On–line transaction processing systems are major producers of production data. On-line

     transaction processing systems require a minimum guaranteed response time with uninterrupted

30   availability, particularly in electronic commerce (e-commerce) systems. The high data volume

     and the need for high availability require the use of transaction servers rather than slower

     database servers.

         Production data provide the raw grist for decision support and on-line analytical

     processing systems. These systems analyze data and generate reports for use in the planning and

35   strategic operations of a corporation. The raw production data is transformed into informational

data by data mining, replication, and cleansing tools. Decision support and on-line analytical processing systems can tolerate slower response times. Nevertheless, the data needs of these systems must balance against the autonomy required by production systems.

Frequently updating the informational databases can adversely impact the operation of the production systems. On-line transaction processing systems operate near or at total hardware capacity. For instance, a typical e-commerce site can receive over 500 transactions or "hits" per second. Interrupting production system operation to update the informational databases can exacerbate the problem of maintaining the requisite level of availability and responsiveness.

Periodically, production data must be transformed into informational data through the application of business logic during the data retrieval process. Often, the business logic required to retrieve and transform production data is complex and computationally intensive. As well, the business logic is relatively inflexible and static. These factors can further affect system responsiveness.

In the prior art, two solutions for updating informational databases have been proposed. One solution presents a data replication manager that periodically copies production data while transforming the data. Unfortunately, this solution causes extensive data duplication and can be time consuming.

Another prior art solution introduces a multi-tiered database architecture with periodic updating. Business logic is implemented in queries executed against the production database. Second tiered business logic can utilize the retrieved information to populate and update datamarts using department-specific queries. In a rapidly changing environment, excessive updates can drastically disrupt production system operation.

Therefore, there is a need for a data manager capable of updating an informational database with high-frequency and low overhead. This approach would minimize resource expenditures by substantially avoiding data duplication and inefficient data retrieval.

There is a further need for an approach to retrieving informational data with dynamically redefinable parameters. This approach would allow flexible redefinition of business logic for selecting data in an *ad hoc* fashion.

There is a further need for an approach to non-intrusively updating an informational database. This approach would have minimal effect on a production system operation and respect autonomous operation.

DISCLOSURE OF INVENTION

The present invention provides a system and method for updating a destination database with data indirectly retrieved from a source database through log-based monitoring. A transaction log file is generated as a by-product of transactions committed to a source database

2

by a transaction server. The log file is monitored and evaluated against a dynamic rule set specifying selection criteria implementing business logic. Those log entries satisfying the selection criteria are converted into updated records using metadata describing the schema of a destination database. The rule set and metadata can be dynamically redefined using a database

5   builder tool. The log monitor automatically modifies the selection criteria and record-generation operations. During the data retrieval, the log monitor utilizes information stored in each log entry to indirectly derive informational data with minimal effect on the transaction server operations.

An embodiment of the present invention is a system and method for refreshing an

10   informational database through log-based transaction monitoring. A production database is maintained and includes one or more tables. Each table stores records of production data generated by a transaction processing system. Log entries are periodically stored into a log file. At least one log entry is generated for each transaction committed to the production database. An informational database including one or more tables is maintained. Each table stores records of

15   informational data for use by a decision support system. The log entries stored into the log file are dynamically analyzed using a rule set that specifies a data selection criteria. The updated records generated from production data satisfying the data selection criteria are stored into the informational database.

A further embodiment is a system and method for maintaining large-grained database

20   concurrency with a log monitor incorporating dynamically redefinable business logic. Operations expressed in a data manipulation language are executed against a source database. At least one operation constitutes a commit operation that completes each database transaction. A current rule set is defined. Each rule includes business logic specifying a data selection criteria for records stored in the source database. A log entry is periodically generated in a log for each

25   transaction committed to the source database. Each log entry identifies an affected record and includes transactional data. The transaction identified in each log entry is evaluated against the data selection criteria specified in the current rule set. A new record is built in accordance with metadata describing a destination database. The new record contains select transactional data from the log entry of each transaction meeting the selection criteria. The new record is stored

30   into the destination database. The data stored in the destination database includes at least a partial subset of the source database.

One benefit of the present invention is the ability to dynamically redefine business logic implemented as rules interpreted by a transaction log monitor. A further benefit is harnessing the metadata intrinsic to a data warehouse to intelligently populate a database and to allow an

35   additional level of responsiveness to changes in the structure of the database.

Still other embodiments of the present invention will become readily apparent to those skilled in the art from the following detailed description, wherein is described embodiments of the invention by way of illustrating the best mode contemplated for carrying out the invention.

As will be realized, the invention is capable of other and different embodiments and its several

5   details are capable of modifications in various obvious respects, all without departing from the spirit and the scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive.

DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram showing a distributed computer environment, including a

10   system for maintaining large-grained database concurrency with a log monitor incorporating dynamically redefinable business logic, in accordance with the present invention.

FIGURE 2 is a functional block diagram showing a prior art multi-tiered database architecture.

FIGURE 3 is a block diagram showing the system for maintaining large-grained database

15   concurrency of FIGURE 1.

FIGURE 4 is a data structure diagram showing a log entry used in the system of FIGURE 3.

FIGURE 5 is a process flow diagram showing informational database updating through log-based transaction monitoring.

20   FIGURE 6 is a functional block diagram showing the software modules of the system of FIGURE 3.

FIGURE 7 is a data structure diagram showing a rule entry.

FIGURE 8 is a flow chart showing a method for maintaining large-grained database concurrency with a log monitor incorporating dynamically redefinable business logic in

25   accordance with the present invention.

BEST MODE FOR CARRYING OUT THE INVENTION

FIGURE 1 is a block diagram showing a distributed computing environment 10, including a system for maintaining large-grained database concurrency with a log monitor incorporating dynamically redefinable business logic, in accordance with the present invention.

30   An operational data store (ODS) 11 stores production data generated by a production system 12. The production system 12 can constitute an on-line transaction processing system for transacting electronic commerce (e-commerce), reservations, point of sale transactions, inventory control, factory and manufacturing operations, and similar types of activities. For efficiency, the production system 12 maintains a local production database 13 within which production data is

35   maintained.

To maintain a high level of responsiveness and availability, the operational data store 11 is coupled to a dedicated transaction server 14 that provides a high throughput interface to the operational data store 11. The production system 12 and transaction server 14 are interconnected via a network 15, such as an internetwork or an intranetwork, as are known in the art.

5   The production data stored in the operational data store 11 is periodically replicated into an enterprise data warehouse 16. Unlike the operational data store 11, which only contains production data, the enterprise data warehouse 16 contains both direct and derivative data values, known as informational data, for use by decision support and on-line analytical processing systems. A database server 17 is coupled to the enterprise data warehouse 16 for executing data

10   manipulation language (DML) queries against the enterprise data warehouse 16. The database server 17 is also interconnected to the other systems via the network 15.

The enterprise data warehouse 16 can be hierarchically structured with secondary databases, such as a workgroup datamart 18. While the enterprise data warehouse 16 contains informational data pertaining to all aspects of the corporate activities, each work group datamart

15   18 is a subset presenting a specialized view of the informational data for a specific subject area, such as employee records, sales revenue figures, and the like. The work group datamart 18 could be structured in a local network environment with a dedicated local database server 19 interconnected with a plurality of clients 20 over a local network 21. In turn, the local network 21 can be interconnected with the corporate network 15, via a series of hubs 22.

20   Finally, the enterprise data warehouse 16 and the work group datamart 18 can both be remotely accessed by a remote client 23, for instance via an internetwork 24, such as the Internet, coupled to the corporate network 15 through a gateway 25.

The operational data store 11 is characterized by a high degree of volatility and change with 100% availability and guaranteed response times. Conversely, the enterprise data

25   warehouse 16 performs flexible operations responsive to *ad hoc* queries posed by the various clients 20 and remote clients 23.

The informational data stored in the enterprise data warehouse 16 must be periodically refreshed with production data retrieved from the operational data store 11. Depending upon the system load on the operational data store 11, the retrieval of production data may be impractical

30   during peak operational times and at best tolerated during off-peak periods. Consequently, a log monitor 26 can provide large-grained database concurrency between the operational data store 11 and enterprise data warehouse 16 by indirectly updating the informational data.

The log monitor 26 reads log entries generated by the transaction server 14 as a by-product of transaction processing. Individual log entries are retrieved and analyzed by applying

35   selection criteria implementing business logic into a dynamically redefinable rule set, as further

5

described below with reference to FIGURE 3. Updated records are generated from those log entries satisfying the selection criteria for updating the informational data in the enterprise data warehouse 16.

5

While the specific forms of databases, including operational data store 11, enterprise data warehouse 16, and workgroup datamart 18, are referenced in relation to the described embodiment, one skilled in the art would recognize that other forms of structured databases could also be used within the general parameters and characteristics outlined herein. In addition, other networked topologies and system configurations can also be used.

10

The individual computer systems, including production system 12, transaction server 14, database server 17, local database server 19, clients 20 and remote client 23, are general purpose, programmed digital computing devices consisting of a central processing unit (CPU), random access memory (RAM), non-volatile secondary storage, such as a hard drive or CD-ROM drive, network interfaces, and peripheral devices, including user-interfacing means, such as a keyboard and display. Program code, including software programs, and data are loaded into the RAM for

15

execution and processing by the CPU and results are generated for display, output, transmittal, or storage.

FIGURE 2 is a functional block diagram showing a prior art multi-tiered database architecture 30. Production and informational databases are structured into several tiers to distribute the database updating and retrieval workload, such as described in R. Orfali,

20

"Client/Server Survival Guide," Chs. 12-13, John Wiley & Sons, Inc. (3d ed.1999), the disclosure of which is incorporated by reference. A plurality of production databases 13 are served by a transaction server 14. Production data is stored and retrieved using the transaction server 14 into and from an operational data store 11 at high volume with maximum availability (step 31).

25

Periodically, a database server 17 "taps" informational data from the operational data store 11 (step 32). The raw production data copied, purified and cleansed into informational data maintained in an enterprise data warehouse 16. The retrieval of production data from the operational data store 11 is performed by a scheduled recurring process, such as a *cron* job, that periodically awakens to refresh the production data from the operational data store 11 into the

30

enterprise data warehouse 16.

The process of downloading the production data into the enterprise data warehouse 16 is nontrivial and taxes computational, storage and network interfacing resources. The informational data retrieval and transformation executes business logic for selecting the appropriate data values from the operational data store 11. Each update can potentially implicate a massive volume of

35

replicated information.

To mitigate the load on the operational data store 11, a series of specialized datamarts, such as, by way of example, a promotional datamart 34, an affinity datamart 35, and a market basket datamart 36, are created as a subset of the enterprise data warehouse 16 (step 33). Like the enterprise data warehouse 16, the datamarts 34-36 must also be periodically refreshed with

5 informational data by a local data server 19. The use of the local database server 19 offloads a part of the specialized business logic necessary to further refine the informational data into formats usable by decision support and on-line transaction processing system (not shown).

This prior art approach attempts to balance the needs of the decision support and on-line analytical processing systems against the autonomous operation of the production systems by

10 distributing and offloading the data replication and processing operations. However, the multi-tiered architecture 30 replicates the information needed in each of the specialized databases. In addition, multiple layers of business logic implementations are required to refresh and update the informational data. These layers can potentially include duplicated queries. A less duplicative solution is needed.

15 FIGURE 3 is a block diagram showing the system 50 for maintaining large-grained database concurrency of FIGURE 1. The core functionality of the system is performed by log monitor 26. A source database 51, typically an operational datastore 11 (shown in FIGURE 2), or alternatively an enterprise data warehouse 16 or work group datamart 18, stores production (or informational) data. The stored data values change as transactions are committed to the source

20 database 51. Upon commitment, a log writer 53 generates a log entry into a transaction log 54 to journal the transacted event. Each log entry includes transactional data, as further described below with reference to FIGURE 4, that identifies the table, record and operations performed. The log writer 53 "flushes" a running series of transaction logs 52 (TL), generated on a continuous basis.

25 As log entries are written by the log writer 53 into the log 54, the log monitor 26 applies selection criteria incorporating business logic to the journaled log entries.

For each log entry that satisfies the selection criteria, the log monitor 26 retrieves the associated production (or informational) data from the source database 51 to generate updated records 57 for a destination database 58. The selection criteria utilized by the log monitor 26 is

30 implemented as a rule set 58. The rule set incorporates business logic for selecting the transactions affecting production (or informational) data of interest to a decision support or on-line analytical processing system. In addition, the log monitor 26 utilizes metadata 56 describing the schema employed by the destination database 58.

The rule set 55 and metadata 56 can be dynamically redefined through a database builder

35 tool 59. An example of a database builder tool 59 suitable for use in the present invention is the

Oracle Warehouse Builder product, licensed by Oracle Corporation, Redwood Shores, California. A user can redefine the business logic and metadata using the database builder tool 59. Redefinitions are regularly forwarded to the log monitor 26. Individual rules in the rule set 58 implement the business logic as data manipulation language (DML) operations. The metadata 56

5   is used by the log monitor 26 to structure the updated records 57 into a format used by tables stored in the destination database 58. A structure of the rule set 55 is further described below with reference to FIGURE 7.

Preferably, the metadata 56 includes four components, as follows. First, the metadata 56 describes the architectures of the source database 51 and destination database 58. The metadata

10   56 also includes network information describing the database links and aliases to other networks. The log 54 can also chronicle transactions performed on logically joined databases, such as in a distributed database environment. In addition, the metadata 56 includes job control and scheduling information specifying a frequency of execution for the implemented business logic against the journaled transaction entries. In the described embodiment, the job control

15   information is specified in a higher order database manipulation language known as PL/SQL. Finally, the metadata 56 includes the actual source code for executing a selection criteria analysis. Other forms of metadata could also be used, for both production (or informational) data retrieval or recovery.

As an optimization to performance, the log monitor 26 can also include a cache 60 for

20   staging information, including log entries and updated records 57. In the described embodiment, the log monitor 26 is implemented as a process separate from the transaction server 14 and database servers 17, 19 (shown in FIGURE 1). The log monitor 26 operates in accordance with a sequence of process steps, further described below with reference to FIGURE 8.

25   FIGURE 4 is a data structure diagram showing a log entry 70 used in the system 50 of FIGURE 3. Each log entry 70 includes, by way of example, five fields of information. A time stamp 71 chronicles the date and time that the associated transaction was committed to the source database 51. The table identifier 72 and record identifier 73 specify the source table and record entry or entries against which the transaction was committed. The operation type 74 identifies

30   the committed transactions. Finally, the *undo* information 75 optionally describes the actual changes applied to the data in a format that allows the committed transaction to be unrolled. In the described embodiment, the *undo* information 75 identifies the table space contents and storage locations for the associated record. Using the *undo* information 75, the committed transaction can be unrolled and the original record entry recovered. The *undo* information 75 can

also be used by the log monitor 26 (shown in FIGURE 3) to determine whether the selection criteria of the rule set 55 has been satisfied for the current log entry 70.

FIGURE 5 is a process flow diagram showing informational database updating through log-based transaction monitoring. Data is updated as updated records 57 from the source

5   database 51 into the destination database 58. A committed transaction 91 is stored by the transaction server 92 into the source database 51. A log entry 93 is generated upon the commitment of each transaction 91 and is stored into the log 54 by a log writer 53 (shown in FIGURE 3). The log monitor 26 applies business logic 94, as implemented in a rules set 55. Each log entry 93 satisfying the selection criteria is used to generate an updated record 95 that is

10   stored into the destination database 58 by the database server 96.

Unlike the informational data update approach used in the prior art multi-tiered database architecture 30 (shown in FIGURE 2), the present approach generates updated records 95 by indirectly deriving the informational data through the log entries 93. The log entries 93 are generated as a by-product of the transaction server 92. Consequently, the updated records 95 are

15   generated at minimal cost to the transaction server 92 and avoids the data replication and potential duplicity of data and business logic inherent in the prior art multi-tiered database hierarchy.

FIGURE 6 is a functional block diagram showing the software modules 110 of the system 50 of FIGURE 3. The log monitor 26 includes two primary modules: an evaluation module 111

20   and a record-generation module 112. The evaluation module 111 receives the log entries in the log 53 as input. The transaction data included in each log entry 70 (shown in FIGURE 4) is evaluated against the selection criteria presented by the rules set 55. Those log entries 113 that satisfy the selection criteria are forwarded to the record generation module 112. Using the schema description of the destination database 58, stored in the metadata 56, the record

25   generation module 112 creates updated records 57 for updating the destination database 58.

In the described embodiment, the updated records 57 are sent via a file transfer process, such as in accordance with the File Transfer Protocol (FTP) or similar network transport protocol. Each module of the log monitor is a computer program, procedure or module written as source code in a conventional programming language, such as the C++ programming language,

30   and is presented for execution by the CPU as object or byte code, as is known in the art. The various implementations of the source code and object and byte codes can be held on a computer-readable storage medium or embodied on a transmission medium in a carrier wave.

FIGURE 7 is a data structure diagram showing a rule entry 130. By way of example, and at a minimum, each rule 130 identifies the table 131 within the source database 51 to which the

35   selection criteria is to be applied. A set of data manipulation language (DML) statements 132

implements the business logic through which the selection criteria are expressed. As well, a threshold 133 can be included to allow the log monitor 26 (shown in FIGURE 3) to quickly and efficiently filter the log entries based on a predefined boundary condition. Other rule formats are feasible.

5   FIGURE 8 is a flow diagram of a method 150 for maintaining large-grained database concurrency with a log monitor 26 incorporating dynamically redefinable business logic in accordance with the present invention. Preliminarily, the log file 54 is opened by the log monitor 26 (block 151) preparatory to the application of the business logic. Each log entry (shown in FIGURE 4) is then iteratively processed as follows.

10   During each iteration, each log entry 70 is first read by the log monitor 26 (block 152). The log entry 70 is evaluated to determine whether the entry describes a committed transaction 91 (shown in FIGURE 5) affecting a table 131 (shown in FIGURE 7) to which the selection criteria applies (block 153). If the log entry 70 does apply to a listed table 131 (block 153), the rule, as implemented in the data manipulation language procedure 132, is evaluated (block 154).

15   If the selection criteria of the rule is satisfied (block 155), an updated record 57 is generated by the log monitor 26 using the metadata 56 (block 156) and the updated record is sent to the destination database 58 (block 157). Iterative processing continues (blocks 152-158) while there are more log entries 70 (block 158). Upon the processing of the last log entry 70 (block 158), the log file 54 is closed (block 159) and the routine terminates.

20   While the invention has been particularly shown and described as referenced to the embodiments thereof, those skilled in the art will understand that the foregoing and other changes in form and detail may be made therein without departing from the spirit and scope of the invention.

CLAIMS

1   1.   A system (10) for refreshing an informational database through log-based

2   transaction monitoring, comprising:

3   a production database (13) comprising one or more tables each storing records of

4   production data generated by a transaction processing system (14);

5   a log writer (53) periodically storing log entries (70) into a log file (54) with at least one

6   log entry (70) generated for each transaction committed to the production database (13);

7   an informational database (58) comprising one or more tables each storing records of

8   informational data for use by a decision support system; and

9   a log monitor (26) dynamically analyzing the log entries (70) stored into the log file (54)

10   using a rule set (55) that specifies a data selection criteria and storing updated records (57)

11   generated from production data satisfying the data selection criteria into the informational

12   database (58).

1   2.   A system according to Claim 1, further comprising:

2   the informational database (58) further comprising metadata (56) describing the structure

3   of the informational database (58); and

4   the log monitor (26) converting the production data into the updated records (57) based

5   on the metadata (56) for the informational database.

1   3.   A system according to Claim 2, further comprising:

2   a database builder (59) generating the metadata (56) and defining the rule (55) set from

3   the data selection criteria structured as provided by the metadata (56).

1   4.   A system according to Claim 3, wherein the metadata (56) comprises at least one

2   parameter selected from the group comprising the production database (13) and the informational

3   database (58) architectures, network topology information, job control information, and program

4   code for performing the data selection.

1   5.   A system according to Claim 1, further comprising:

2   the log monitor (26) storing the updated records (57) into the informational database (58)

3   on a substantially continuous basis as a consistent data set of the production database (13).

1   6.   A system according to Claim 1, further comprising:

2   the log monitor (26) creating replicas of at least one table of records selected from the

3   informational database (58).

1   7.   A system according to Claim 1, further comprising:

2   the log monitor (26) performing job control over each refreshment of the informational

3   database (58) with the updated records (57).

1   8.   A system according to Claim 1, further comprising:

2   a log monitor cache (60) staging the updated records (57) generated by the log monitor

3   (26) as a transient data set shared with the informational database (58).

1   9.   A method (150) for refreshing an informational database (58) through log-based

2   transaction monitoring, comprising:

3   maintaining a production database (13) comprising one or more tables each storing

4   records of production data generated by a transaction processing system (14);

5   periodically storing log entries (70) into a log file (54) with at least one log entry (70)

6   generated for each transaction committed to the production database (13);

7   maintaining an informational database (58) comprising one or more tables each storing

8   records of informational data for use by a decision support system; and

9   dynamically analyzing the log entries (70) stored into the log file (54) using a rule set (55)

10   that specifies a data selection criteria and storing updated records (57) generated from production

11   data satisfying the data selection criteria into the informational database (58).

1   10.   A method according to Claim 9, further comprising:

2   storing metadata (56) describing the structure of the informational database (58) into the

3   informational database; and

4   converting the production data into the updated records (57) based on the metadata (56)

5   for the informational database.

1   11.   A method according to Claim 10, further comprising:

2   generating the metadata (56) and defining the rule set (55) from the data selection criteria

3   structured as provided by the metadata (56).

1   12.   A method according to Claim 11, wherein the metadata (56) comprises at least

2   one parameter selected from the group comprising the production database (13) and the

3   informational database (58) architectures, network topology information, job control information,

4   and program code for performing the data selection.

1   13.   A method according to Claim 9, further comprising:

2   storing the updated records (57) into the informational database (58) on a substantially

3   continuous basis as a consistent data set of the production database (13).

1   14.   A method according to Claim 9, further comprising:

2   creating replicas of at least one table of records selected from the informational database

3   (58).

1   15.   A method according to Claim 9, further comprising:

2   performing job control over each refreshment of the informational database (58) with the

3   updated records (57).

1   16.   A method according to Claim 9, further comprising:

2   staging the updated records (57) generated by the log monitor (26) as a transient data set

3   shared with the informational database (58).

1   17.   A computer-readable storage medium holding code for performing the method of

2   Claim 9.

1   18.   A system for maintaining large-grained database concurrency with log monitoring

2   (26) means incorporating dynamically redefinable business logic (94), comprising:

3   a source database engine (12) executing operations expressed in a data manipulation

4   language against a source database (51) with at least one operation constituting a commit

5   operation that completes each database transaction;

6   a database builder (59) defining a current rule set (55) with each rule comprising business

7   logic (94) specifying a data selection criteria for records stored in the source database;

8   means for periodically generating (53) a log entry (70) in a log (54) for each transaction

9   committed to the source database, each log entry (70) identifying an affected record and

10   including transactional data;

11   means for monitoring the log (26), comprising:

12   an evaluation module (111) evaluating the transaction identified in each log entry

13   (70) against the data selection criteria specified in the current rule set (55); and

14   a record generation module (112) building a new record (57) in accordance with

15   metadata (56) describing a destination database (58) and containing select transactional data from

16   the log entry (70) of each transaction meeting the selection criteria; and

17   a destination database engine (96) storing the new record (57) into the destination

18   database (58) with the data stored in the destination database (58) comprising at least a partial

19   subset of the source database (51).

1   19.   A system according to Claim 18, further comprising:

2   the database builder (59) dynamically redefining the current rule set (55) comprise

3   business logic (94) to specify a revised data selection criteria.

1   20.   A system according to Claim 18, further comprising:

2   the destination database engine creating a new source database (51) as specified by the

3   metadata (56).

1   21.   A system according to Claim 18, further comprising:

2   a temporary data store (60) caching each new record.

1   22.   A system according to Claim 18, wherein the transaction data comprises

2   information selected from the group comprising a timestamp (71), table identifier (72), record

3   identifier (73), operation type (74), and undo information (75).

1   23.   A method for maintaining large-grained database concurrency with a log monitor

2   (26) incorporating dynamically redefinable business logic (94), comprising:

3   executing operations expressed in a data manipulation language against a source database

4   (51) with at least one operation constituting a commit operation that completes each database

5   transaction;

6   defining a current rule set (55) with each rule comprising business logic (94) specifying a

7   data selection criteria for records stored in the source database;

8   periodically generating a log entry (70) in a log (54) for each transaction committed to the

9   source database, each log entry (70) identifying an affected record (73) and including

10   transactional data;

11   evaluating the transaction identified in each log entry (70) against the data selection

12   criteria specified in the current rule set (55);

13   building a new record (57) in accordance with metadata (56) describing a destination

14   database (58) and containing select transactional data from the log entry (70) of each transaction
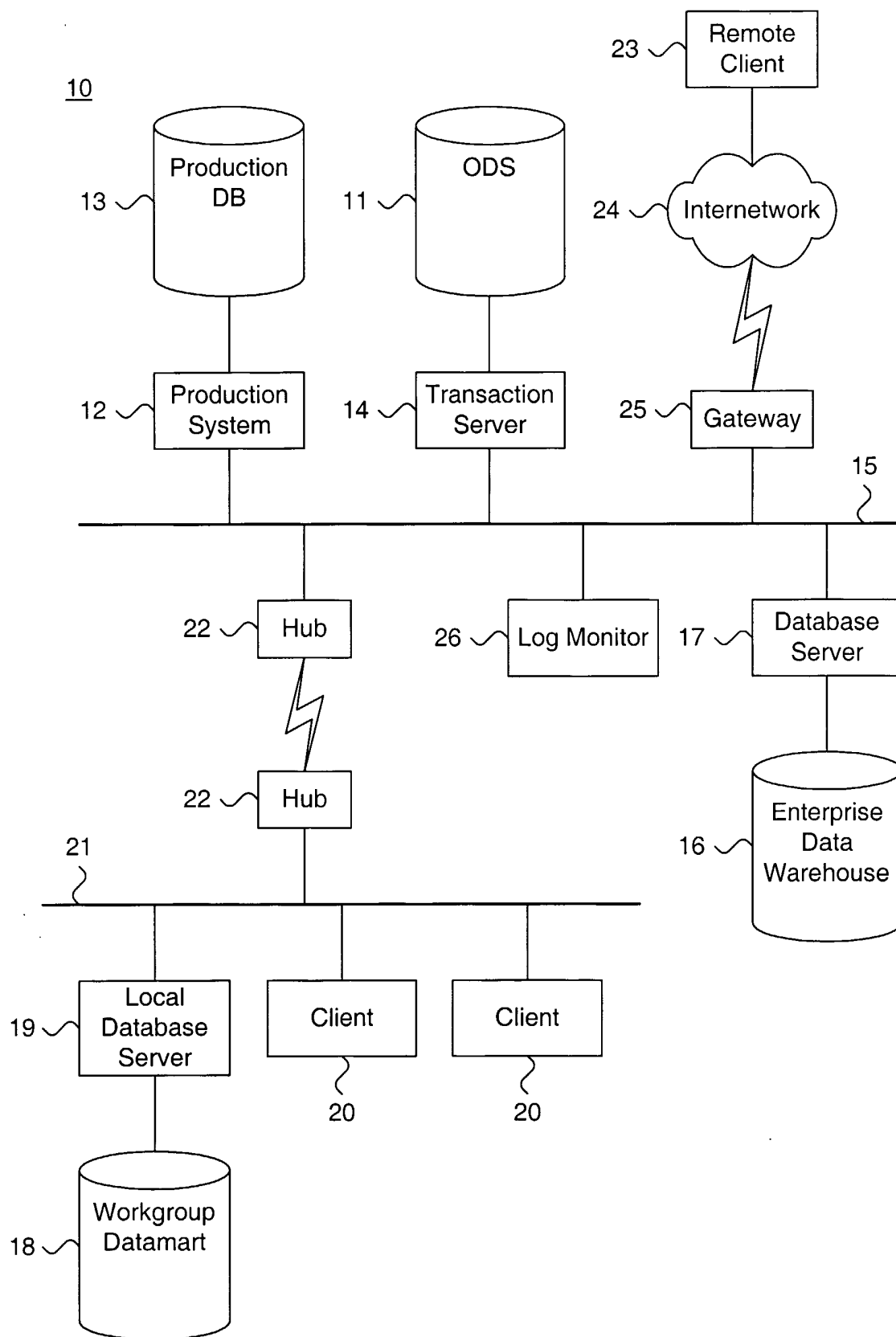
15   meeting the selection criteria; and

16   storing the new record (57) into the destination database (58) with the data stored in the

17   destination database (58) comprising at least a partial subset of the source database (51).

1   24.   A method according to Claim 23, further comprising:

2   dynamically redefining the current rule set (55) comprise business logic (94) to specify a
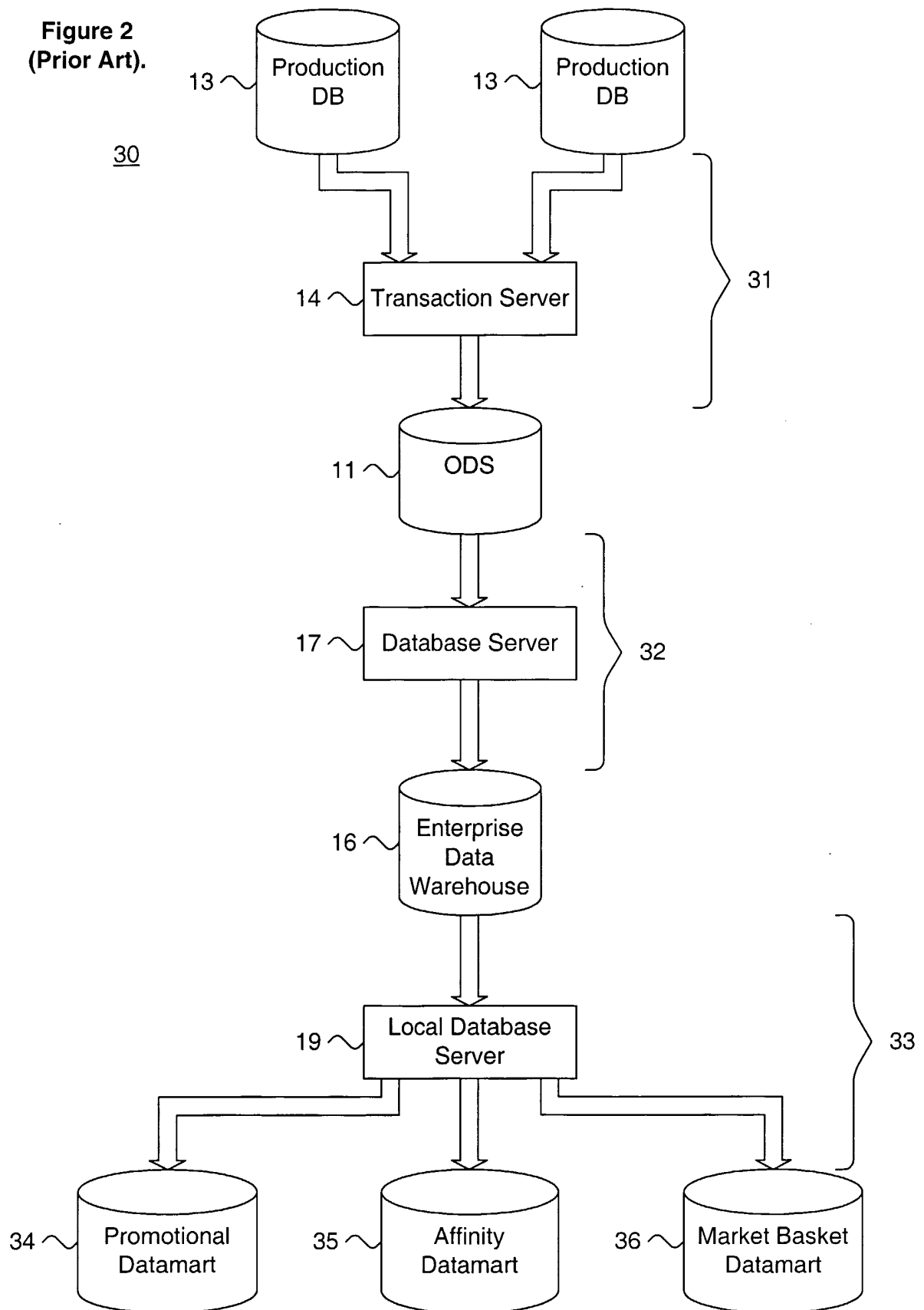
3   revised data selection criteria.

1   25.   A method according to Claim 23, further comprising:

2   creating a new source database (51) as specified by the metadata (56).

1   26.   A method according to Claim 23, further comprising:

2   caching each new record (57) in a temporary data store.

1   27.   A method according to Claim 23, wherein the transaction data comprises

2   information selected from the group comprising a timestamp (71), table identifier (72), record

3   identifier (73), operation type (74), and undo information (75).

1   28.   A computer-readable storage medium holding code for performing the method of
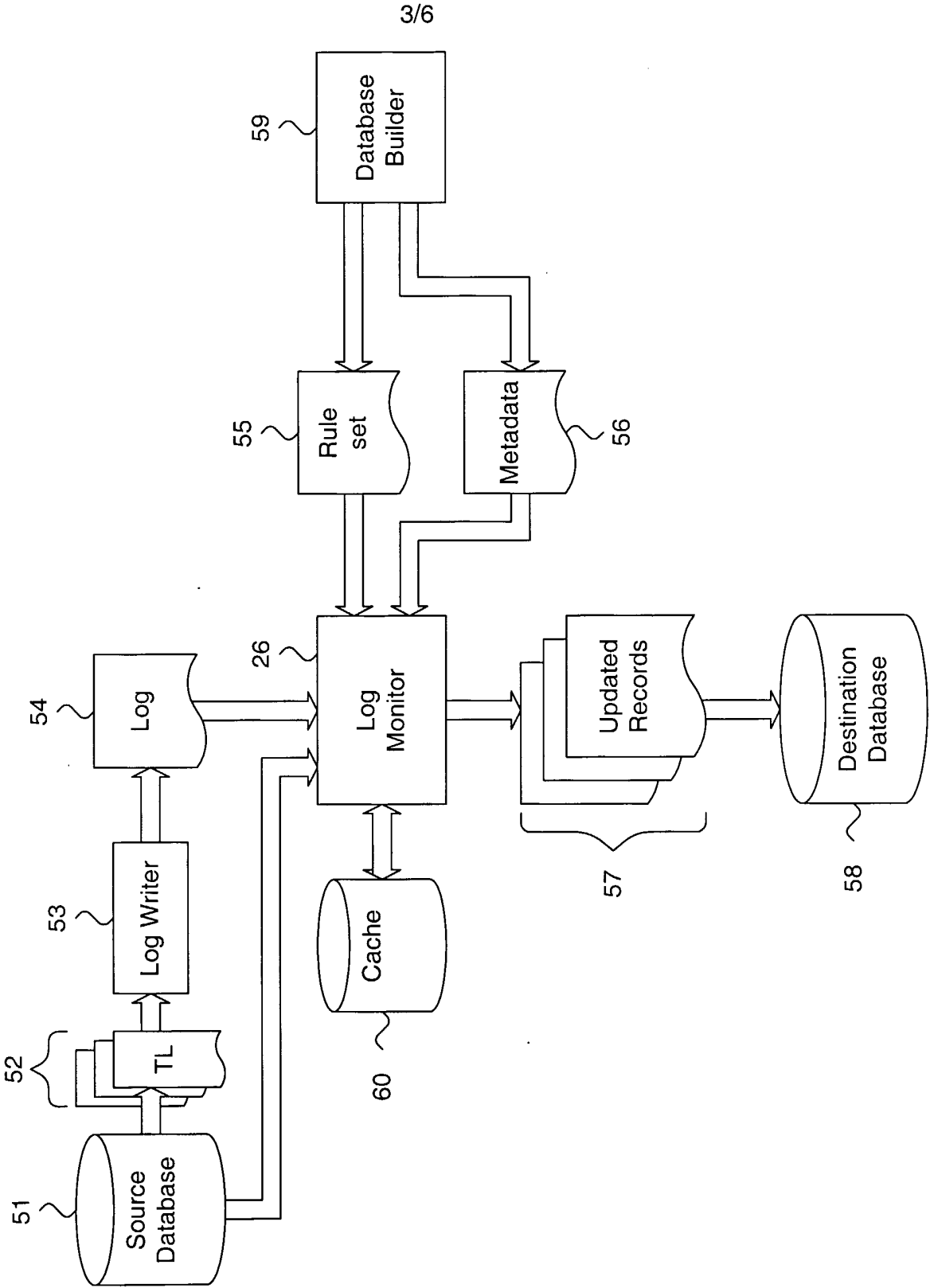
2   Claim 23.

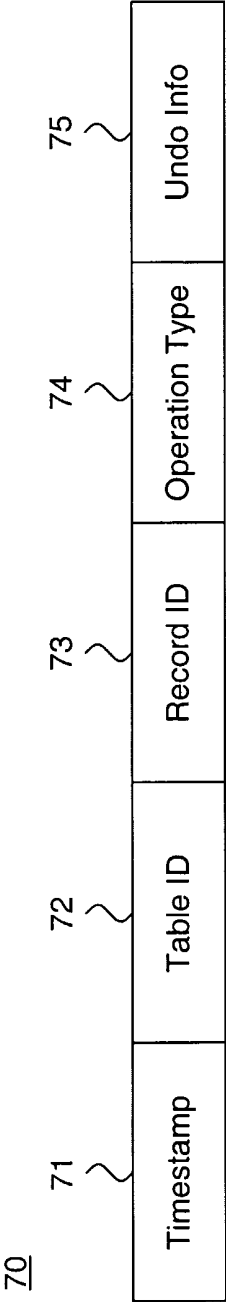**Figure 1.**

**Figure 2
(Prior Art).**
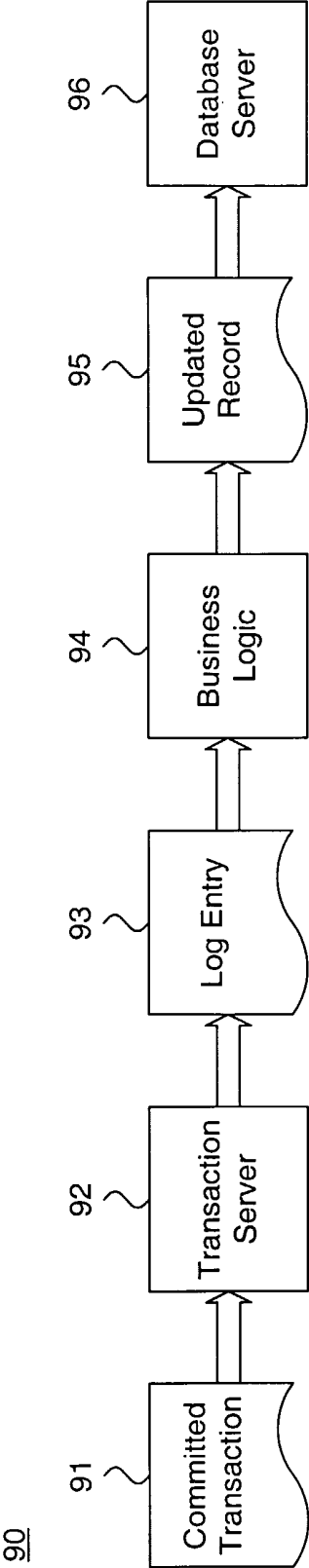
30

3/6

Figure 3.   50

Figure 4.

70

Timestamp	Table ID	Record ID	Operation Type	Undo Info
71	72	73	74	75

Figure 5.

90

Committed Transaction (91) → Transaction Server (92) → Log Entry (93) → Business Logic (94) → Updated Record (95) → Database Server (96)

Figure 7.

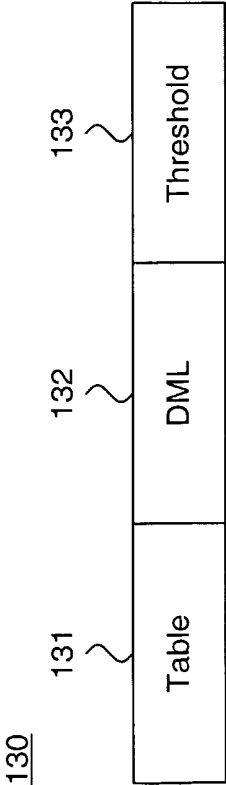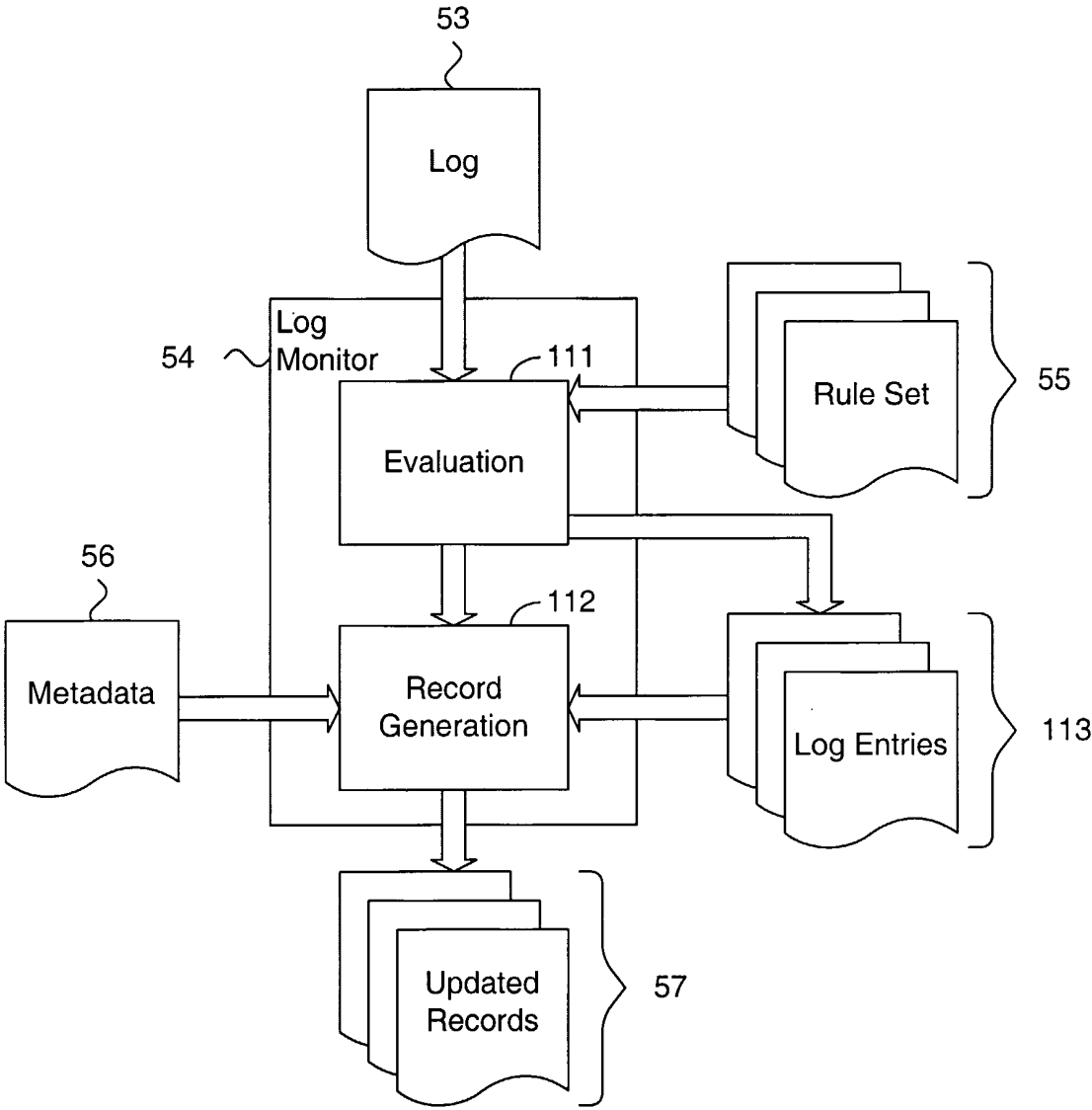130

Table	DML	Threshold
131	132	133

5/6

Figure 6.

<u>110</u>

6/6

Figure 8.