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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 July 2003 (31.07.2003)

PCT

(10) International Publication Number
WO 03/063027 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US03/01222

(22) International Filing Date: 15 January 2003 (15.01.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/349,433	18 January 2002 (18.01.2002)	US
10/340,301	10 January 2003 (10.01.2003)	US

(71) Applicant: **BEA SYSTEMS, INC.** [US/US]; 2315 North First Street, San Jose, CA 95131 (US).

(72) Inventor: **WHITE, Seth**; 1045 Rivera Street-Apartment B, Sans Francisco, CA 94116 (US).

(74) Agents: **MEYER, Sheldon, R.** et al.; Filesler Dubb Meyer & Lovejoy LLP, Suite 400, Four Embarcadero Center, San Francisco, CA 94111-4156 (US).

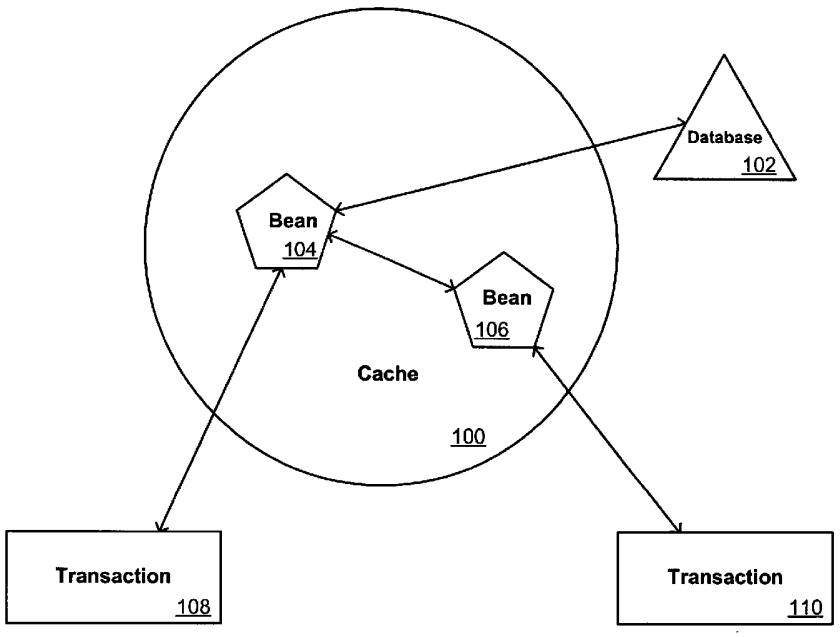
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD FOR READ-ONLY ENTITY BEAN CACHING



(57) Abstract: Separate instances of a data item can be allocated for each transaction in a system. Each instance can be held by an entity bean (104 and 106) capable of updating by either reading a copy of the data item from the database (102) or by reading a copy from another entity bean. When the data item in the database is changed, the system can notify each entity bean that it should read an updated copy od the data item before the next transaction (108 and 110).

WO 03/063027 A1

System and Method for Read-Only Entity Bean Caching

CLAIM OF PRIORITY

This application claims priority to the following applications each of which is hereby incorporated herein by reference:

This application claims priority to U.S. Provisional Patent Application No. 60/349,433, filed January 18, 2002, entitled "System and Method for Read-Only Entity Bean Caching," which is hereby incorporated herein by reference.

U.S. Patent Application No. _____ entitled "System and Method for Read-Only Entity Bean Caching," by Seth White, filed on January 10, 2003.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document of the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

CROSS REFERENCE TO RELATED PATENT DOCUMENTS

The following co-pending U.S. patent documents are assigned to BEA Systems, Inc., the assignee of the present application, and these documents are hereby incorporated herein by reference:

(A) U.S. Patent Application filed _____, to Seth White et al. and entitled, "System and Method for Heterogeneous Caching";and

(B) U.S. Patent Application filed _____, to Seth White et al. and entitled, "System and Method for Optimistic Caching".

(C) U.S. Patent Application filed October 11, 2001, to Dean Jacobs et al. and entitled "Data Replication Protocol".

FIELD OF THE INVENTION

The invention relates generally to the caching of data, such as for transactions.

BACKGROUND

In an environment where multiple users can concurrently access data, many systems currently hold a copy of each data item in a read-only entity bean. A single instance of each read-only bean is kept in memory so that transactions can access the read-only bean instead of hitting the database. In order to prevent the transactions from stepping on each other, the transactions are able to utilize an exclusive lock such that the transaction can lock that read-only bean for its own exclusive use during each method call on the bean. The current transaction can be suspended, a new transaction can be started, and the bean can be locked on behalf of the new transaction and the method on the bean called by the new transaction. One problem with this approach is that it requires a lot of overhead. It is necessary for the system to manage the suspending, resuming, and committing of transactions.

Exclusive locking, such as in an entity bean container, also limits scalability. Exclusive locking is undesirable in many applications because the data is read-only and many users wish to be able to view the data at the same time and not have to wait for earlier transactions to finish.

BRIEF SUMMARY

Systems and methods in accordance with one embodiment of the present invention can provide for improved data caching for transactions. Each transaction can require the reading of a data item that is contained in a database. The data item can be read into a cache capable of caching instances of the data item. An allocation algorithm can allocate each instance of the data item to one of the transactions. Each instance can be held by an entity bean stored in the cache, for example, such as an entity bean container. Any other mechanism for storing an instance of a data item can also be used in accordance with embodiments of the present invention. Each bean can have associated with it a time-out value that can help to determine how long the bean should hold the instance of the data item. After the time-out value has expired, the bean can obtain a new instance of the data item by reading from the database or from another bean in a bean container. The system can also notify each bean in the container when the data item in the database is changed.

Other features, aspects, and objects of the invention can be obtained from a review of the specification, the figures, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a diagram of a system in accordance with one embodiment of the present invention.

Figure 2 is a diagram of a system in accordance with another embodiment of the present invention.

Figure 3 is a flowchart showing the steps of a method in accordance with the embodiment of the Figure 1.

DETAILED DESCRIPTION

Systems and methods in accordance with one embodiment of the present invention can allow multiple users to concurrently view data by allocating multiple instances of the same bean. Each transaction can be given its own instance, such that the transactions can read and use the data in parallel. The system can activate and manage the multiple copies, and can keep the copies consistent with the underlying data. The transactions can then read the data from the bean instances, such as for example entity bean, session bean, or Enterprise JavaBean instances, without having to access the database. For backward compatibility, the system can also include an option enabling exclusive locking for certain transactions. For instance, an existing application might require an exclusive lock on a data item. The system can allow for an exclusive lock where it is needed, but disallow locking for all other cases. Where the data is not going to change, there will be no anomalies or inconsistencies generated by allowing users to view the data concurrently. For example, there is no reason why a thousand users requesting the price of a new release compact disc at an online store should have to wait "their turn" to view the price.

One way to keep the beans reasonably synchronized with data in the database is to associate a "time-out" or "time-to-live" value with each bean. After a bean has been in existence for a period of time equal to its "time-out" value, the bean will expire and the transaction will have to acquire a new instance of the bean that contains a potentially newer version of the data. In some systems, however, this approach may still be a little too resource intensive. Also, this approach might be most useful for systems in which this type of data is not often changed or is not changed by these types of transactions.

Systems and methods in accordance with another embodiment of the present invention can reduce the number of hits to the database even further by using an algorithm to find recent copies of the data. Such an algorithm might include checking version numbers or indices of each copy, checking the age of each copy, or checking information from a list holding information for each copy. Using such an algorithm, a bean that reaches its time-out period can look to see if there is another bean in the system that contains a more recent copy of the data held by that bean. If the bean finds a more recent copy, or a bean in a more recent state, the timed-out bean can read data from the bean with the more recent version instead of going to the database. The system can also keep a list of beans, or at least the identity of the most recent bean, so that timed-out beans can easily identify the most recent beans. This approach, however, can result in beans not having the most recent version of the data, as beans will only read current data from the database when there is not a more recent copy of the data in the system.

Users can balance the need to conserve resources with the need to have the most recent copy of the data when deciding between the above two exemplary systems. For an application in which there are many users, but for which the data does not often change, it might be better to allow beans to read from each other to conserve resources. In applications where it is important that users get the most up-to-date data, or where the data changes more often, it might be desirable for each bean to read from the database.

An example of such a system is shown in **Figure 1**. A cache **100**, such as an entity bean container, can cache beans **104** and **106**. Beans **104** and **106** are separate instances that are associated with the same data in the database **102**. Bean **104** is allocated by the system to transaction **108**, and bean **106** is allocated to transaction **110**. If bean **104**

is timed-out, it can obtain a more recent copy of the data by reading from the database **102** or reading the data from bean **106**. Bean **104** can read from bean **106**, if bean **106** has a newer copy of the data, is a "younger" bean, or if it has the same version of the data, but has an updated state. If bean **106** does not meet any of these criteria, bean **104** can read the data from the database **102**. The system can also have a more intelligent algorithm that allows bean **104** to read from bean **106** when there are many users on the system, and allows bean **104** to read from the database when there are few users of the system. This allows a user to get the most recent data when the resources are available.

Either of the above systems can utilize a notification algorithm. A notification algorithm can be any algorithm useful in notifying other beans of updates, such as by multicasting a message to the beans or by contacting each relevant bean instance directly by a point-to-point protocol. Messages can be sent any of a number of places in a cluster or on a network, such as from a cluster server, a server manager, a cluster manager, a bean, a database, and a database manager. For example, in **Figure 2** a transaction **202** causes a bean **204** in cache **200** to update data in the database **206**. In this example, the bean **204** also sends an update message to other beans **208**, **210**, **212** in the cache **200** holding an instance of the data item being held and updated by the bean **204**. This message can alert the beans **208**, **210**, **212** that the data has been updated, can tell them to drop their instance, or can include the updated data itself so the beans can automatically have the updated version.

When using a notification algorithm, any bean that receives an updated copy of the data can notify all other beans associated with that data that the data has changed. This can allow the system to propagate the new data more quickly, as beans can know that they need to update

data before beginning a new transaction, whether the beans read the data from the database or from another bean. Also, the system can send out invalidation messages, such as by multicasting, so each bean can know to re-read the data the next time a transaction needs the data, or when a bean times-out.

An example of such a method is shown in **Figure 3**. A data item is read from a database for a transaction **300**. An entity bean is allocated by the system to hold an instance of that data item for the transaction **302**. An invalidation message is received by the entity bean indicating that the data item in the database has changed **304**. The entity bean drops its instance of the data item **306**. A new instance of the data item is read from another entity bean that holds a current copy of the data item **308**. This can be, for example, the entity bean that caused the data item to be updated. If no entity beans hold a current version of the data item, the entity bean can re-read the item from the database.

The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to one of ordinary skill in the relevant arts. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims and their equivalence.

What is claimed is:

1. A system for improved data caching for transactions, comprising:

a database capable of containing a data item;

a cache capable of caching at least one instance of the data item in the database; and

an allocation algorithm adapted to allocate each instance of the data item to a transaction.

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

a plurality of entity beans that can hold an instance of the data item.

3. A system according to claim 1, wherein:

the cache comprises an entity bean container.

4. A system according to claim 1, further comprising:

an entity bean for holding each instance of the data item, each entity bean being capable of updating the instance of the data item by reading from the database.

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

a plurality of beans, each bean capable of holding an instance of the data item, each bean further being capable of updating the instance of the data item by reading from another bean holding an instance of the data item.

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

a plurality of beans, each bean being capable of holding an instance of the data item, each bean having a time-out value that determines how long the bean holds that instance of the data item.

7. A system according to claim 1, further comprising:
a notification algorithm capable of notifying each instance of the data item when the data item in the database is changed.

8. A system for improved data caching for transactions, comprising:
a database capable of containing a data item;
a cache capable of caching instances of the data item; and
a bean for each instance of the data item, each bean adapted to hold an instance of the data item and being capable of updating the instance of the data item by reading an instance from another bean in the cache.

9. A system according to claim 8, wherein:
the cache comprises a bean container.

10. A system according to claim 8, wherein:
the bean is further capable of updating the instance of the data item by reading from the database.

11. A system according to claim 8, further comprising:
a time-out value for the bean that determines how long the bean holds that instance of the data item.

12. A system according to claim 8, further comprising:
a notification algorithm adapted to notify each bean when the data item in the database is changed.

13. A method for improved data caching for transactions, comprising:
allocating a bean for each transaction to hold an instance of the data item;

reading an instance of the data item from a database to each bean; and

updating a bean by reading a copy of the data item from the database.

14. A method according to claim 13, further comprising:

assigning a time-out value for each bean, the time-out value determining when the bean is updated.

15. A method according to claim 13, further comprising:

notifying a bean holding an instance of the data item when the data item in the database is changed.

16. A method for improved data caching for transactions, comprising:

reading a data item from a database for each transaction;

allocating a bean for each transaction to hold an instance of the data item; and

updating a bean by reading a copy of the data item from another bean holding an instance of the data item.

17. A method according to claim 16, further comprising:

assigning a time-out value for each bean, the time-out value determining when the bean is updated.

18. A method according to claim 16, further comprising:

notifying each bean holding an instance of the data item when the data item in the database is changed.

19. The system of claim 1, further comprising:

a plurality of beans, each of the plurality of beans holding an instance of the data item, each bean having a time-out value that determines how

long the bean holds the instance of the data item and thereafter the bean updates the instance of the data item by one of reading from another bean holding an instance of the data item or reading from the database.

20. The system of claim 1, further comprising:

an entity bean capable of holding an instance of the data item in the cache.

21. The system of claim 8, further comprising:

a time-out value for each bean, the time-out value capable of determining how long a bean holds the instance of the data item, the bean capable of thereafter updating the instance of the data item by one of reading from another bean holding an instance of the data item or reading from the database.

22. The system of claim 1 wherein:

each bean is an entity bean.

23. The method of claim 13, wherein:

allocating further includes allocating an entity bean for each transaction

24. The system of claim 1, further comprising:

a plurality of beans holding each instance of the data item, wherein each bean is capable of updating a respective instance of the data item by reading from another bean holding an instance of the data item, as well as by reading from the database.

25. The method of claim 13, wherein:

each bean is an entity bean.

26. The method of claim 16, wherein:

each bean is an entity bean.

27. The method of claim 13, wherein:

each bean is capable of being updated by at least one of reading from another bean and by reading from the database.

28. The method of claim 16, wherein:

each bean is capable of being updated by at least one of reading from another bean and by reading from the database.

29. The method of claim 13, further comprising:

notifying each bean each time the data item in the database has changed.

30. A system for improved data caching for transactions, comprising:

a database adapted to contain a data item;

a cache capable of caching multiple instances of the data item, wherein each of said instances can be associated with a transaction;

a read-only bean associated with each instance of the data item in the cache;

wherein the data item in each read-only bean can be updated by both of reading from the database and reading from another bean; and

wherein multiple transactions can access multiple respectively associated read-only beans in a non-serial manner in order to access the data item.

31. The system of claim 30, wherein:

a transaction can be carried on using the data in the read-only bean without requiring a lock on the bean.

32. The system of claim 5, further comprising:

an index having a list of beans with instances of the same data item.

33. The system of claim 5, further comprising:

an index having a list of beans with updated instances of the same data item.

34. The system of claim 8, further comprising:

an index having a list of beans with instances of the same data item.

35. The system of claim 8, further comprising:

an index having a list of beans with updated instances of the same data item.

36. The system of claim 24, wherein:

the system preferentially reads from the database in order to update an instance of a data item when the demand for access to the database is reduced.

37. The system of claim 27, wherein:

the system preferentially reads from the database in order to update an instance of a data item when the demand for access to the database is reduced.

38. The system of claim 28, wherein:

the system preferentially reads from the database in order to update an instance of a data item when the demand for access to the database is reduced.

5

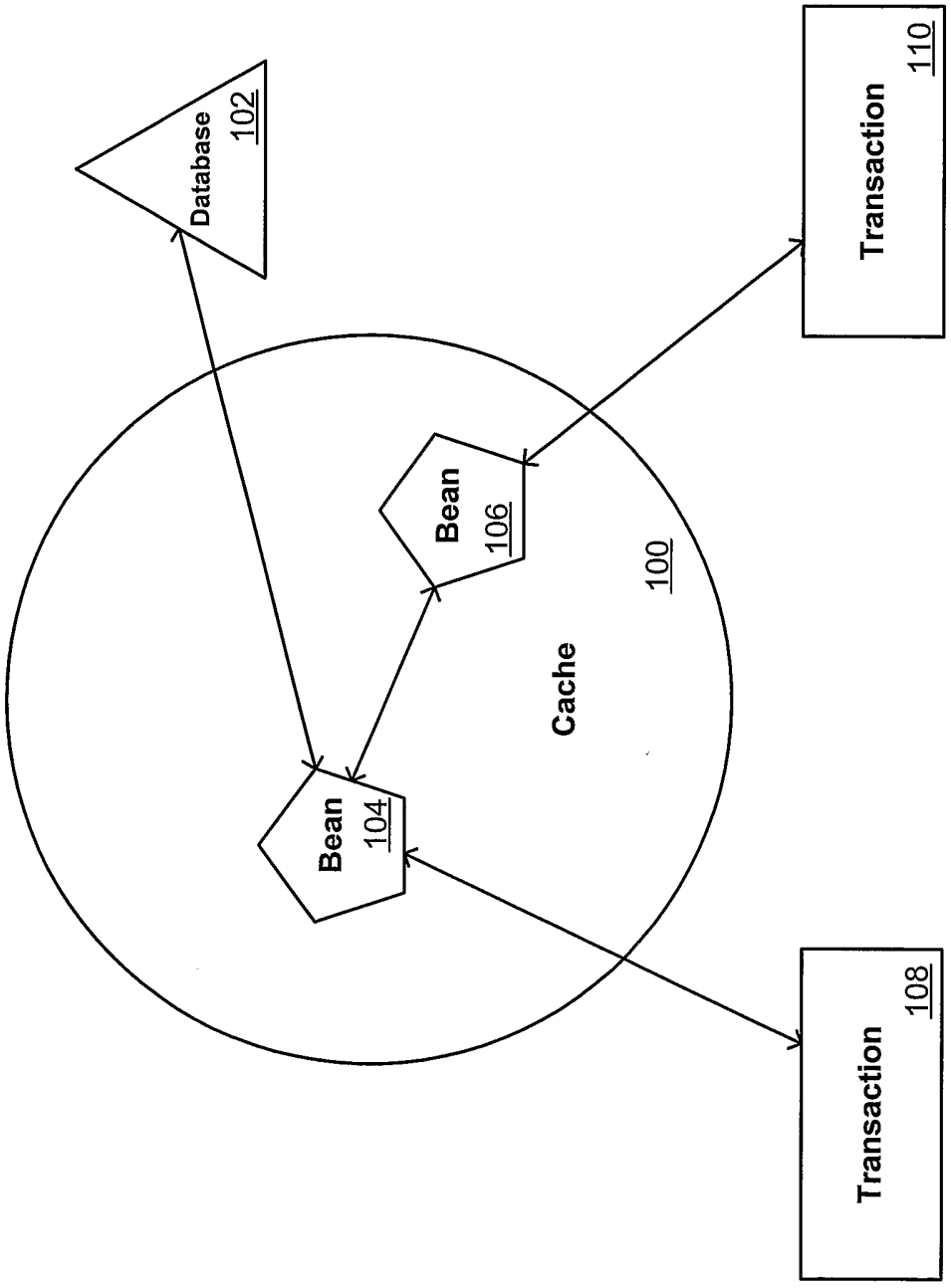

Database
102
Transaction
110
Bean
106
Cache
100
Bean
104
Transaction
108

Figure 1

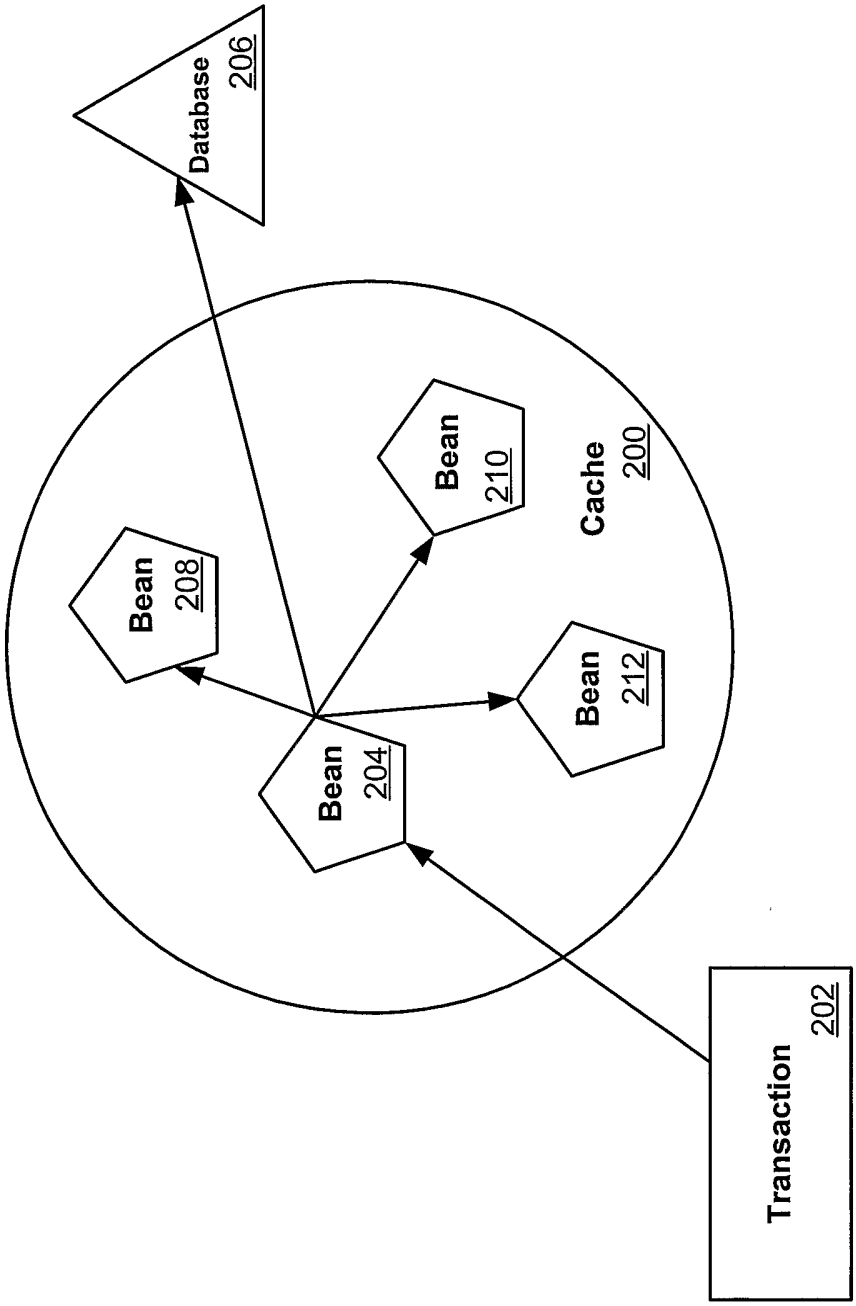

Database
206
Cache
200
Bean
208
Bean
210
Bean
212
Bean
204
Transaction
202

Figure 2

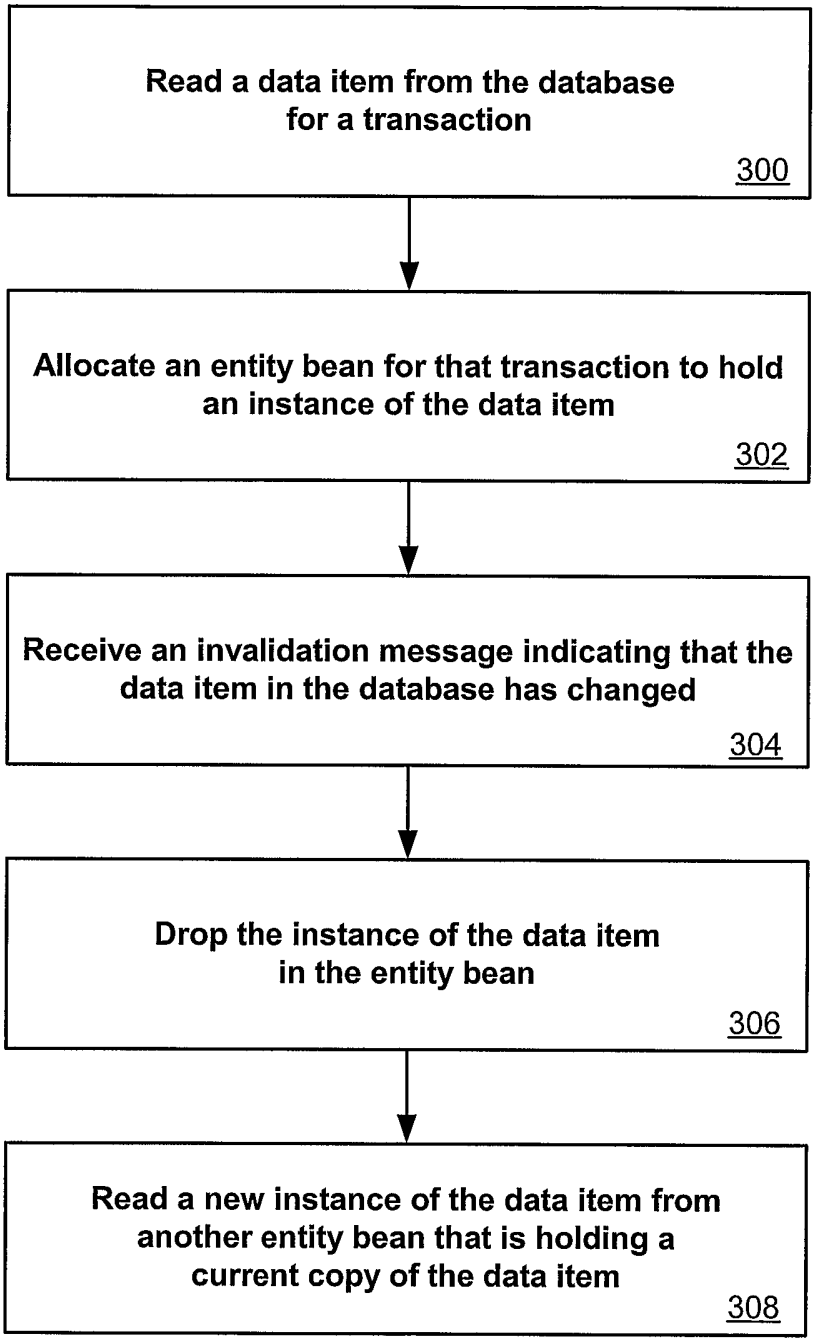

Read a data item from the database for a transaction
300
Allocate an entity bean for that transaction to hold an instance of the data item
302
Receive an invalidation message indicating that the data item in the database has changed
304
Drop the instance of the data item in the entity bean
306
Read a new instance of the data item from another entity bean that is holding a current copy of the data item
308

Figure 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/01222

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/30
US CL : 707/8, 9, 10, 201

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/8, 9, 10, 201

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST search; East Search, Internet Search

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US 6,457,065 B1 (RICH et al) 24 September 2002 (24 .9.2002), columns 6-21.	1-29
A,P		30-38
Y	US 6,298,478 B1 (NALLY et al) 02 October 2001 (02.10.2001), columns 5-20.	1-29
A		30-38

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 March 2003 (21.03.2003)	01 APR 2003
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Haythim J. Alaubaidi Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)