

(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 03/062987 A1

- (51) International Patent Classification⁷: **G06F 9/45**
- (21) International Application Number: PCT/US03/01211
- (22) International Filing Date: 15 January 2003 (15.01.2003)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
60/349,432 18 January 2002 (18.01.2002) US
10/341,107 13 January 2003 (13.01.2003) US
- (71) Applicant: **BEA SYSTEMS, INC.** [US/US]; 2315 North First Street, San Jose, CA 95131 (US).
- (72) Inventors: **SHINN, Matthew**; 714 Hayes Street, San Francisco, CA 94102 (US). **WHITE, Seth**; Apartment B, 1045 Rivera Street, San Francisco, CA 94116 (US). **WOOLEN, Rob**; 2531 14th Avenue, San Francisco, CA 94127 (US).
- (74) Agents: **MEYER, Sheldon, R.** et al.; Fliesler 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: SYSTEMS AND METHODS FOR DYNAMIC QUERYING

(57) Abstract: ABSTRACT A user can generate queries dynamically at runtime without having to redeploy the appropriate EJB or hard-code the query into the user application. A properties object can be generated to accept the query settings from the user. These settings can be extracted at runtime when the appropriate finder method is invoked, such that the desired query statement, such as a SQL statement, can be generated and executed against the database. This description is not intended to be a complete description of, or limit the scope of, the invention. Other features, aspects, and objects of the invention can be obtained from a review of the specification, the figures, and the claims.



WO 03/062987 A1

SYSTEMS AND METHODS FOR DYNAMIC QUERYING

CLAIM OF PRIORITY

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

5 This application claims priority to U.S. Provisional Patent Application No. 60/349,432, filed January 18, 2002, entitled "SYSTEMS AND METHODS FOR DYNAMIC QUERYING," which is hereby incorporated herein by reference.

10 U.S. Patent Application No. _____ entitled "SYSTEMS AND METHODS FOR DYNAMIC QUERYING," by Matt Shinn, et al., filed on January 10, 2003.

COPYRIGHT NOTICE

15 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.

FIELD OF THE INVENTION

20 The present invention relates to executing queries against a database.

BACKGROUND

25 The Enterprise JavaBean (EJB) specification, published by Sun Microsystems, Inc. of Palo Alto, CA, describes ways in which a user can execute queries against a database, as well as ways in which a user can communicate queries to an EJB container. Presently, the EJB 2.0

specification forces users to hard-code finder queries into a deployment descriptor for an EJB. A user develops a query before deploying the EJB. Once the EJB is deployed, the user is able to execute the query. A problem exists with this approach, however, in that it is necessary to
5 redeploy the EJB every time the user wishes to run a new query.

BRIEF SUMMARY

Systems and methods in accordance with one embodiment of the present invention can allow a user to dynamically generate a query to be
10 executed against a database. A properties object can be generated that holds settings for the query, which can be specified by a user at runtime. When the query is to be executed, the user or application can invoke an appropriate finder method. The server receiving the call from the finder method can extract the user-specified settings from the properties object
15 and parse the finder method in order to generate a query statement. The server can then execute the query statement on the database and return the appropriate results. The generating of the properties object and the query statement can happen at runtime.

Other features, aspects, and objects of the invention can be
20 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.

25 Figure 2 is a flowchart showing the steps of a method that can be used with the system of Figure 1.

DETAILED DESCRIPTION

Systems and methods in accordance with one embodiment of the present invention can allow a user to define a query programmatically
30 rather than defining the query statically. Static queries are defined, for

example, by hard-coding the static query into the deployment descriptor for an EJB. Programmatic queries, or “dynamic queries”, allow users to construct and execute queries in their application code. This can provide several benefits over static queries which utilize static finder methods.

5 One such benefit is the ability to create and execute new queries without having to update and redeploy an EJB. When deploying an EJB with static queries, each query is read and parsed in order to generate the SQL to be sent to the database. Finder methods can be utilized in
10 executing the query, which can be defined in the home interface of an entity bean. An example of a finder method is `findByPrimaryKey()`, which can accept an instance of a primary key and return an instance of that entity type (or throw an exception). Additional finder methods can be defined in local home or remote home interfaces, with each finder method being associated with a query in the deployment descriptor. With dynamic
15 queries, however, the query and corresponding SQL can be generated at runtime.

 Another benefit is that the size of an EJB deployment descriptor is reduced. Since the finder queries can be created dynamically, they do not have to be statically defined in the deployment descriptor. For some
20 applications this approach may be a little slower, but the added flexibility will outweigh the slight hit in performance for many users.

 One system and method for implementing dynamic queries utilizes the generation of a class such as an `ejbHome` class. Such a class can be used to implement an extra interface with a method that can execute the
25 query. As shown in **Figure 1**, when a user **100** wants to execute a query against a database **112** at runtime, an object such as a Java properties object **104** can be generated that can be populated with the settings for the finder method **106**, such as a container-managed or bean-managed finder method. The finder method **106** can then be invoked on the query home
30 of the appropriate EJB **108**, which can be stored on a server **110** or EJB container in communication with, and capable of executing SQL queries

against, a database **112**. Once the call makes it into the server **110**, the properties object **104** can be inspected and the user settings extracted. The finder method **106** can be parsed and the SQL query statement generated that is to be sent to the database **112**. The query is executed and, depending on the user settings, the use of the results can be determined. One possible result of such a query is a collection of EJBs. Another possible result is a number of values or fields on certain EJBs that match the query.

A method that can be used in accordance with the system of Figure 1 is shown in the flowchart of **Figure 2**. In the method, a properties object is generated that contains user-specified settings for the query or the finder method, as can be implemented through a user interface of an ejbHome class **200**. The appropriate finder method is invoked when the user or application wishes to execute the query **202**. The settings are extracted from the properties object and the finder method is parsed in order to generate the appropriate SQL query statement, although other database or data source querying language statements may be generated by the method **204**. The SQL query statement is then executed against the database **206**.

One embodiment can be implemented through a simple API. To enable the use of dynamic queries, users can add an element to their deployment descriptor, such as:

<!ELEMENT enable-dynamic-queries (#PCDATA)>

The enable-dynamic-queries element can be a sub-element of a descriptor such as entity-descriptor. The value of enable-dynamic-queries can be either "true" or "false" in this embodiment. Invoking a dynamic query when dynamic queries have not been enabled can result in an exception being thrown, such as `java.rmi.AccessException` or `javax.ejb.AccessLocalException`, depending on whether it was invoked from a Remote or Local interface.

A generated implementation class, such as `HomeImpl` that can be

used for all EJB 2.0 Container-Managed Persistence (CMP) beans, can implement a new interface such as QueryHome. A QueryHome interface can declare a single method, such as:

```
5      public Object executeQuery(String query, Properties props) throws
      FinderException, RemoteException;
```

There can also be a local version of QueryHome which may be referred to as QueryLocalHome. The only difference between the interfaces can be the "throws" clause of the executeQuery method. The QueryLocalHome interface can declare a single method:

```
      public Object executeQuery(String query, Properties props) throws
      FinderException, EJBException;
```

15 The application code can make use of this interface as follows:

```
20      InitialContext ic = new InitialContext();
      FooHome fh = (FooHome)ic.lookup("fooHome");
      QueryHome qh = (QueryHome)fh;
      String query = "SELECT OBJECT(e) FROM EmployeeBean e WHERE e.name
      = 'rob' ";
      Properties props = new Properties();
      props.setProperty(DynamicQuery.MAX_ELEMENTS, "10");
      Collection results = (Collection)qh.executeQuery(query, props);
```

25 All options that can currently be specified for a static finder can be set in a Properties object passed to the executeQuery method. The Properties key for all valid options can be defined in a DynamicQuery interface. A list of some valid entries is as follows:

	Property:	Value:	Description:
	GROUP_NAME	String	The name of the field-group whose fields are to be loaded into the cache upon execution of the query. Note that in order for this to work, a finders-load-bean or equivalent option may need to be enabled for the EJB.
	MAX_ELEMENTS	int	The max-elements attribute is used to specify the maximum number of elements that should be returned by a multi-valued query. This option can be similar to the maxRows feature of JDBC.
5	INCLUDE_UPDATES	boolean	The include-updates tag is used to specify that updates made during the current transaction must be reflected in the result of a query.
	SQL_SELECT_DISTINCT	boolean	Used to control whether the generated SQL 'SELECT' will contain a 'DISTINCT' qualifier. Use of the DISTINCT qualifier will cause the RDBMS to return unique rows.
	RETURN_TYPE	String	Indicates the return type of the executeQuery method. Legal values include Collection, CursoredCollection, and ResultSet. The default value is java.util.Collection.
	NEW_TRANSACTION	boolean	Indicates whether a new transaction should be started for the execution of the DynamicQuery
	ISOLATION_LEVEL	String	Indicates the isolation level to be used if a new transaction is started
10	RESULT_TYPE_MAPPING	String	Indicates whether EJBObjects or EJBLocalObjects should be returned. The legal values are Local and Remote. If the query was executed on QueryHome, EJBObjects will always be returned. If the query was executed on QueryLocalHome, EJBLocalObjects will be returned by default. A result-type-mapping of Remote can be specified in this case if EJBObjects are desired.

Ideally, dynamic queries execute nearly as fast as static queries. Dynamic queries can invariably be somewhat slower since the queries can require parsing at runtime, whereas static queries are parsed during deployment. The speed of dynamic queries can be increased, such as by
5 extending them to take query parameters and caching the parsed query String.

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
10 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
15 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 dynamically generating a query to be executed on a database, comprising:

5 a properties object adapted to contain settings for a query as specified by a user;

a finder method for initiating the query; and

10 a server capable of querying the database, the server capable of receiving a call from the finder method and reading the settings from the properties object in order to generate the appropriate SQL query statements to be sent to the database.

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

15 a user interface adapted to allow a user to specify the settings in the property object.

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

a client containing the properties object and the finder method.

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

20 a bean associated with the database upon which the finder method can be invoked.

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

25 an element in a deployment descriptor for enabling dynamic querying.

6. A method for dynamically generating a query to be executed against a database, comprising:

30 generating a properties object containing settings for a query;

invoking a finder method;

extracting the settings from the properties object and parsing the

finder method in order to generate a query statement; and
executing the query statement on the database.

7. A method according to claim 6, wherein:

5 generating a properties object occurs at runtime.

8. A method according to claim 6, wherein:

 extracting the settings from the properties object and parsing the
finder method in order to generate a SQL query statement occurs at
10 runtime.

9. A method according to claim 6, further comprising:

 enabling dynamic querying by setting an element in a deployment
descriptor.
15

10. A method according to claim 6, further comprising:

 allowing the user to specify the settings for the query at runtime.

11. A method for generating dynamic queries, comprising:

20 specifying settings to be used in generating the query using a user
interface;

 generating a properties object to hold the settings;

 invoking a finder method; and

 parsing the finder method and reading the settings from the property
25 object in order to generate the query.

12. A system for generating dynamic queries, comprising:

 means for generating a properties object to hold settings to be used
in generating a query;

30 means for invoking a finder method; and

 means for parsing the finder method and reading the settings from

the property object in order to generate the query.

13. A computer-readable medium, comprising:

5 means for generating a properties object to hold settings to be used
in generating a query;

means for invoking a finder method; and

means for parsing the finder method and reading the settings from
the property object in order to generate the query.

10 14. A computer program product for execution by a server computer for
generating dynamic queries, comprising:

computer code for generating a properties object to hold settings to
be used in generating a query;

computer code for invoking a finder method; and

15 computer code for parsing the finder method and reading the
settings from the property object in order to generate the query.

15. A computer system comprising:

a processor;

20 object code executed by said processor, said object code configured
to:

generate a properties object to hold settings to be used in
generating a query;

invoke a finder method; and

25 parse the finder method and reading the settings from the
property object in order to generate the query.

16. A computer data signal embodied in a transmission medium,
comprising:

30 a code segment including instructions to generate a properties
object to hold settings to be used in generating a query;

a code segment including instructions to invoke a finder method;
and

a code segment including instructions to parse the finder method
and reading the settings from the property object in order to generate the
5 query.

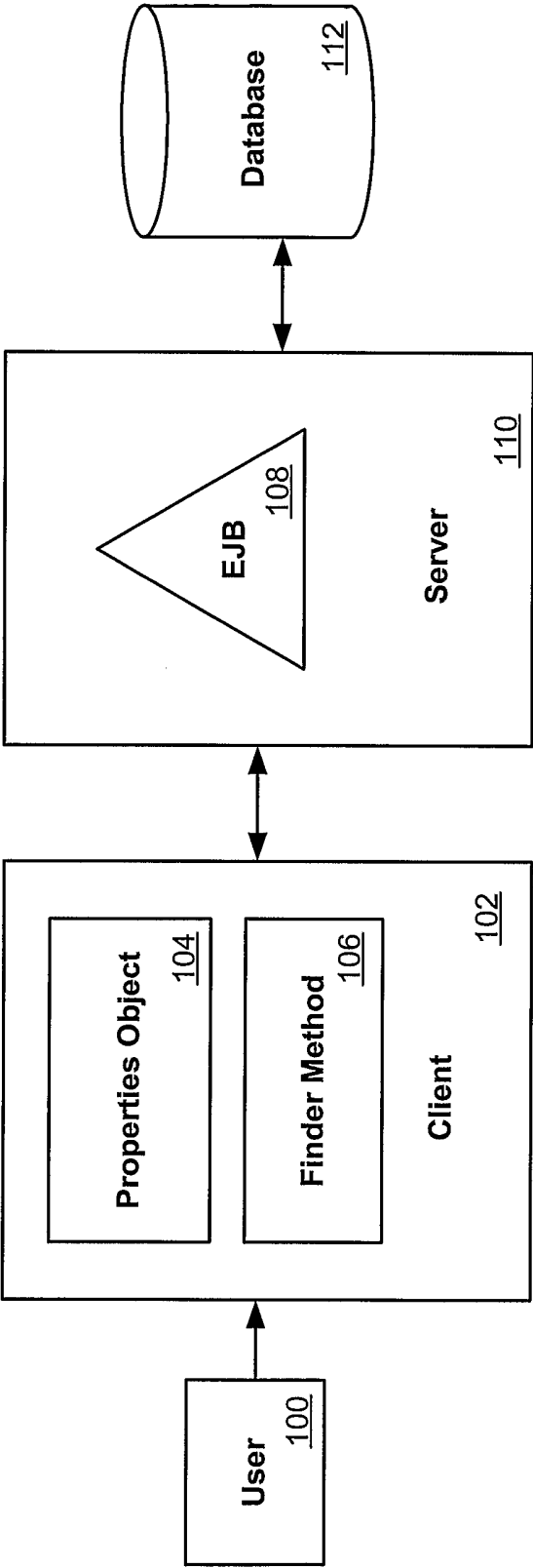


Figure 1

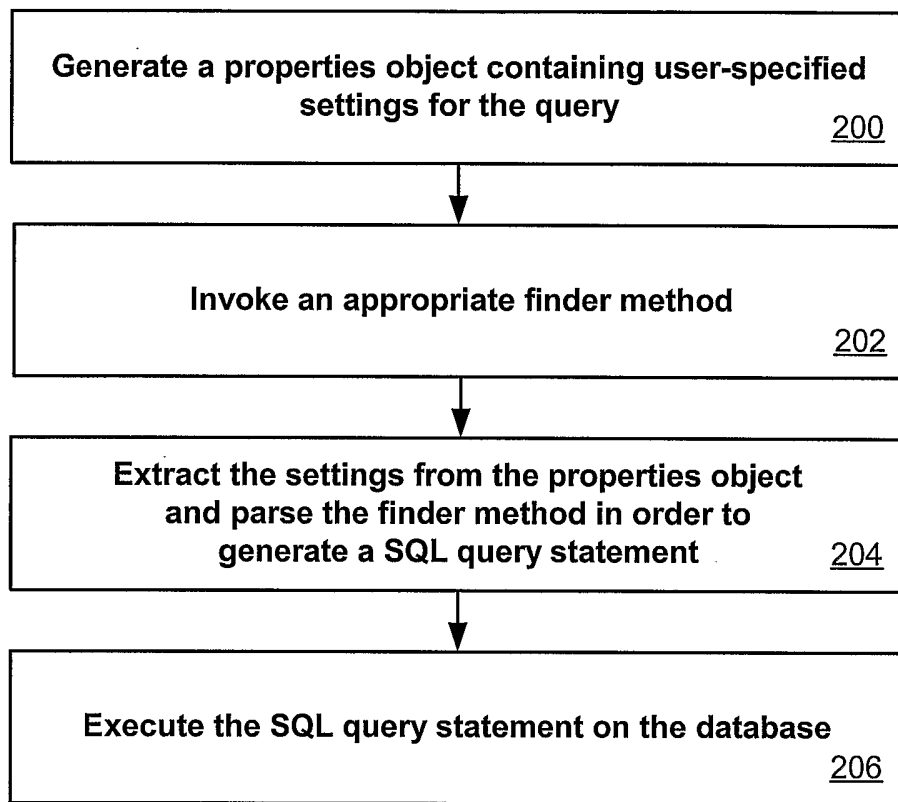


Figure 2

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US03/01211

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 9/45

US CL : 707/3

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. : 707/3, 4, 10, 100-102; 717/1-3, 5; 709/100-332

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)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,199,195 B1 (GOODWIN at al.) 06 March 2001 (06.03.2001), col. 1, line 50 - col. 2, line 20; col. 2, lines 29-56; col. 2, line 64 - col. 3, line 2, lines 14-25; col. 7, lines 39-65; col. 8, lines 6-36; Figures 2-7.	1-16
Y	US 5,499,371 A (HENNINGER at al.) 12 March 1996 (12.03.1996), col. 2, lines 56-65; col. 3, lines 5-22; Figures 1-8B.	1-16
A,P	US 6,466,933 B1 (HUANG at al.) 15 October 2002 (15.10.2002), All	1-16



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	"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
"E" earlier application or patent published on or after the international filing date	"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
"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)	"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
"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	"&" document member of the same patent family

Date of the actual completion of the international search

14 April 2003 (14.04.2003)

Date of mailing of the international search report

28 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

Te Y Chen

Telephone No. (703)308-6296