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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 October 2014 (30.10.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/176363 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/US2014/035187

(22) International Filing Date:
23 April 2014 (23.04.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

61/816,610	26 April 2013 (26.04.2013)	US
61/816,623	26 April 2013 (26.04.2013)	US
13/912,086	6 June 2013 (06.06.2013)	US
13/912,098	6 June 2013 (06.06.2013)	US

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5op7, Redwood Shores, CA 94065 (US).

(72) Inventors: **DE LAVARENE, Jean**; 18 Rue Aristide Briand, F-92300 Levallois-Perret (FR). **ZHOU, Tong**; 1570 Bangs Avenue, Merrick, NY 11566 (US). **SURBER, Douglas**; 192 Hall Drive, Orinda, CA 94563 (US). **FELTS, Stephen**; 3 Cedar Gate Road, Denville, NJ 07834 (US). **MERRILL, David**; 3242 South Court, Palo Alto, CA 94306 (US).

(74) Agents: **MEYER, Sheldon, R.** et al.; MEYER IP LAW GROUP, A Professional Corporation, 410 Pacific Avenue, San Francisco, CA 94133 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: SUPPORT FOR CLOUD-BASED MULTI-TENANT ENVIRONMENTS USING CONNECTION LABELING

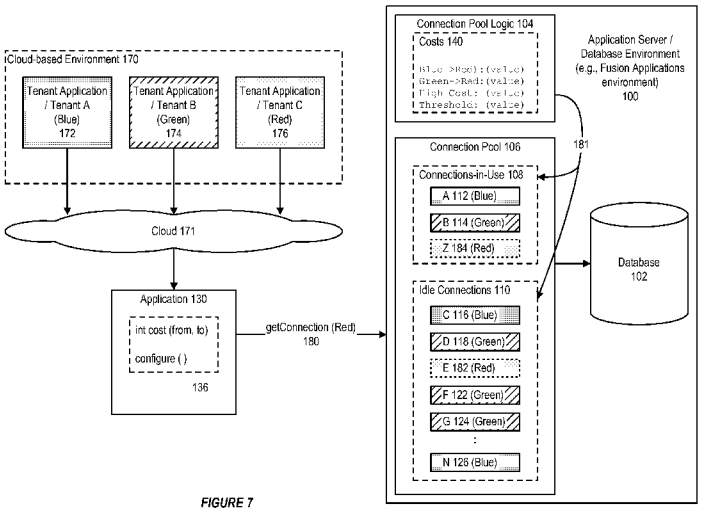

FIGURE 7

(57) Abstract: A system and method for connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling. In accordance with an embodiment, the system comprises a connection pool, including a plurality of connection objects which provide connections that software applications can use to make requests to access the database, wherein each of the connections can be labeled according to the configuration of particular applications; and a connection pool logic that identifies connections labeled as high-cost connections, and controls the creation or repurposing of high-cost connections to serve requests from the multiple tenants or tenant applications. In accordance with an embodiment, the system comprises a connection pool logic that identifies connections labeled as high-cost connections, and avoids using those high-cost connections to serve requests when the total number of connections is below a particular threshold value.

WO 2014/176363 A1

SUPPORT FOR CLOUD-BASED MULTI-TENANT ENVIRONMENTS USING CONNECTION LABELING

Claim of Priority:

[0001] This application claims the benefit of priority to U.S. Provisional Patent Application titled "SYSTEM AND METHOD FOR CONNECTION LABELING FOR USE WITH CONNECTION POOLS" (Attorney Docket No. ORACL-05448US0), Application No. 61/816,610, filed April 26, 2013; U.S. Patent Application titled "SYSTEM AND METHOD FOR CONNECTION LABELING FOR USE WITH CONNECTION POOLS" (Attorney Docket No. ORACL-05448US1), Application No. 13/912,086, filed June 6, 2013; U.S. Provisional Patent Application titled "SUPPORT FOR CLOUD-BASED MULTI-TENANT ENVIRONMENTS USING CONNECTION LABELING" (Attorney Docket No. ORACL-05449US0), Application No. 61/816,623, filed April 26, 2013; and U.S. Patent Application titled "SUPPORT FOR CLOUD-BASED MULTI-TENANT ENVIRONMENTS USING CONNECTION LABELING" (Attorney Docket No. ORACL-05449US1), Application No. 13/912,098, filed June 6, 2013; each of which above applications are herein incorporated by reference.

Field of Invention:

[0002] Embodiments of the invention are generally related to connection pools, and are particularly related to a system and method for connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling.

Background:

[0003] Generally described, a connection pool is a cache of database connection objects. The connection objects represent physical database connections that can be used by a software application to connect to a database. At runtime, an application can request a connection from the pool. If the pool contains a connection that can satisfy the request, it returns the connection to the application. If no connections are found, a new connection can be created and returned to the application. The application uses the connection to access the database to perform work, and then returns the connection to the pool. The connection can then be made available for subsequent connection requests.

[0004] Creating connections can be costly both in terms of time and resources. For example, tasks such as network communication, authentication, transaction enlistment, and memory allocation all contribute to the amount of time and resources it takes to create a connection object. Connection pools allow reuse of such connection objects, and reduce the number of times that objects must be created.

[0005] One example of a connection pool is Oracle Universal Connection Pool (UCP), which provides a connection pool for caching JDBC connections. Java applications that are database-intensive can use the connection pool to improve performance utilization of system resources. A UCP connection pool can use any JDBC driver to create physical connections that are then maintained by the pool. The connection pool can be configured with properties that are used to optimize pool behavior, based on the performance and availability requirements of an application.

Summary:

[0006] In the context of a multi-tenant environment, such as a cloud environment, or a Fusion Applications multi-tenant environment, connection types may be very complex, due to the need to accommodate multiple tenants, and, e.g., to maintain security between each different tenant's access to the database. These complex connections are considered high-cost connections. Approaches to handling high-cost connections can be useful in improving system performance, and/or the performance of applications operating within a cloud environment.

[0007] In accordance with an embodiment, described herein is a system and method for connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling. In accordance with an embodiment, the system comprises a connection pool, including a plurality of connection objects which provide connections that software applications can use to make requests to access the database, wherein each of the connections can be labeled according to the configuration of particular applications; and a connection pool logic that identifies connections labeled as high-cost connections, and controls the creation or repurposing of high-cost connections to serve requests from the multiple tenants or tenant applications. In accordance with an embodiment, the system comprises a connection pool logic that identifies connections labeled as high-cost connections, and avoids using those high-cost connections to serve requests when the total number of connections is below a particular threshold value.

Brief Description of the Figures:

[0008] **Figure 1** illustrates a system for connection labeling for use with connection pools, in accordance with an embodiment.

[0009] **Figure 2** further illustrates a system for connection labeling for use with connection pools, in accordance with an embodiment.

[0010] **Figure 3** further illustrates a system for connection labeling for use with connection pools, in accordance with an embodiment.

[0011] **Figure 4** further illustrates a system for connection labeling for use with connection pools, in accordance with an embodiment.

[0012] **Figure 5** is a flowchart that illustrates a process of connection labeling for use with connection pools, in accordance with an embodiment.

[0013] **Figure 6** illustrates a system for connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling, in accordance with an embodiment.

[0014] **Figure 7** further illustrates a system for connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling, in accordance with an embodiment.

[0015] **Figure 8** is a flowchart that illustrates a process of connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling, in accordance with an embodiment.

Detailed Description:

[0016] In accordance with an embodiment, described herein is a system which includes a connection pool, wherein the system can identify high-cost connections, and avoid using those high-cost connections to serve requests when the total number of connections is below a particular threshold value. In accordance with an embodiment, the system can be used with, or provide support for a cloud-based or multi-tenant cloud environment that allows access to a database via a connection pool.

[0017] In accordance with an embodiment, this allows the connection pool to use new physical connections to serve connection requests from different applications, such as from different tenant applications, without incurring a reinitialization overhead on other connections (e.g., other tenant connections) that may be already pooled.

Connection Labeling

[0018] **Figure 1** illustrates a system for connection labeling for use with connection pools, in accordance with an embodiment. As shown in Figure 1, an application server / database environment 100, such as a Fusion Applications environment, can include or provide access to a database 102. As further shown in Figure 1, the system also includes a connection pool logic 104, which controls 105 the creation and use of objects in a connection pool 106, including connections that are currently in use 108, and connections that are idle

110.

[0019] Software applications 109 may initialize connections 111 retrieved from a connection pool before using the connection. Examples of initialization include simple state re-initializations that require method calls within the application code, or more complex initializations including database operations that require round trips over a network. The cost of these latter types of initialization may be significant.

[0020] Some connection pools, such as the Oracle Universal Connection Pool (UCP), allow their connection pools to be configured using connection pool properties. The properties have get and set methods that are available through a pool-enabled data source instance. These methods are a convenient way to programmatically configure a pool. If no pool properties are set, then a connection pool uses default property values.

[0021] **Figure 2** further illustrates a system for connection labeling for use with connection pools, in accordance with an embodiment.

[0022] In accordance with an embodiment, labeling connections allows an application to attach arbitrary name/value pairs to a connection. The application can then request a connection with a desired label from the connection pool. By associating particular labels with particular connection states, an application can retrieve an already initialized connection from the pool and avoid the time and cost of re-initialization. Connection labeling does not impose any meaning on user-defined keys or values; the meaning of any user-defined keys and values is defined solely by the application.

[0023] For example, as shown in Figure 2, the connection pool can include a plurality of connections that are currently in use, here indicated as connections A 112 and B 114. Each of the connections can be labeled. In the example shown in Figure 2, connection A 112 is labeled (Blue) and connection B 114 is labeled (Green). These labels/colors are provided for purposes of illustration. In accordance with various embodiments, different types of labels can be used to distinguish between different connection types.

[0024] As further shown in Figure 2, the connection pool can also include a plurality of connections that are idle, here indicated as connections C 116, D 118, E 120, F 122, G 124 and N 126. Each of the idle connections can be similarly labeled, in this illustration as (Blue) or (Green), and again these labels/colors are provided for purposes of illustration.

[0025] As further shown in Figure 2, in accordance with an embodiment, if a software application 130 wishes to make a request on the database, using a particular type of connection, for example a (Red) connection, it can make a getConnection(Red) request 132. In response, the connection pool logic will either create a new (Red) connection, here indicated as X 134 (Red); or repurpose an existing idle connection from (Blue or Green) to (Red), here indicated as E 135 (Red).

[0026] **Figure 3** further illustrates a system for connection labeling for use with

connection pools, in accordance with an embodiment.

[0027] In accordance with an embodiment, each software application can utilize a cost function callback to provide configuration information 136 that defines, for that application, costs associated with repurposing connections, and additional configuration information such as high-cost connections and threshold values.

[0028] For example, a particular application may consider the cost of repurposing a (Blue) connection to a (Red) connection to have a value of 50; and the cost of repurposing a (Green) connection to a (Red) connection to have a value of 80; that a high-cost connection has a value of 70; and that a reasonable threshold is 10. The meanings of these values are similarly defined by the application, and the above are provided for purposes of illustration. In accordance with various embodiments, different numeric or non-numeric values can be used to distinguish between different connection costs.

[0029] In accordance with an embodiment, the connection pool logic iterates over each connection available in the pool. For each connection, it calls a cost method. The result of the cost method is an integer which represents an estimate of the cost required to reconfigure the connection to the required state. The larger the value, the costlier it is to reconfigure the connection.

[0030] In accordance with an embodiment, the configuration information 140 provided by the application 138 can be used by the connection pool logic in determining 141 whether to create or repurpose connections, particularly high-cost connections. For example, in accordance with an embodiment, the system can perform a process similar to that illustrated by the pseudocode below.

```
High-Cost: 70
Threshold: 10
getConnection(Red)
IF perfect match (Red)
      THEN return it
      ELSE find cheapest connection
IF cheapest connection's cost < High-Cost
      THEN repurpose this connection
      ELSE IF sum connections < Threshold
            THEN create new connection and apply label
            ELSE sum conn ≥ Threshold THEN repurpose cheapest
connection
```

[0030] Using the above illustration, in accordance with an embodiment, a particular application may define High-Cost to be 70, and Threshold to be 10.

[0031] When the system receives a request for a particular connection type (e.g., Red), the connection pool logic first determines if there is a perfect/existing match (i.e., an idle Red connection), and if so returns that connection for use by the application. Else, the connection pool logic finds the cheapest connection that can be repurposed (to be a Red connection). If the cost of the cheapest connection is less than High-Cost (70), then that connection is repurposed. Else, if the total number of connections is less than Threshold (10), then a new (Red) connection is created, labeled accordingly, and provided to the application. Else, if the total number of connections is greater than or equal to Threshold, then the cheapest connection is repurposed as a (Red) connection 142.

[0032] **Figure 4** further illustrates a system for connection labeling for use with connection pools, in accordance with an embodiment.

[0033] In accordance with an embodiment, when the total number of active and idle connections is low, a request to use a high-cost connection of a particular type may result in a new high-cost connection Y 144 (Red) being created, rather than an existing (potentially also high-cost) connection begin repurposed. The new type connection can then be used for subsequent requests of that type. Although the proposed approach may result in a high-cost connection being created, rather than an existing (potentially also high-cost) connection being repurposed, the approach can provide considerable performance improvements, particularly in complex, e.g., multi-tenant cloud environments, which generally utilize high-cost connections.

[0034] **Figure 5** is a flowchart that illustrates a process of connection labeling for use with connection pools, in accordance with an embodiment. As shown in Figure 5, in accordance with an embodiment, at step 152, the system receives a request for a connection to the database (e.g., getConnection (red)).

[0035] At step 154, the system determines whether there a perfect/existing connection match (red). If there is an existing matching connection, then at step 156, the existing (red) connection is returned. Otherwise, at step 158, the cheapest existing non-matching connection (e.g., blue, green) is found.

[0036] At step 160, the system determines whether the cheapest non-matching connection cost is less than a high-cost. If it is, then at step 162, the cheapest non-matching connection is repurposed as a (red) connection.

[0037] At step 164, the system determines whether the sum of all connections is less than a threshold. If it is, then at step 166, a new connection is created, and the appropriate connection label (red) applied to the new connection. Otherwise, at step 168, if the sum of all connections is greater than or equal to the threshold, then the cheapest connection is repurposed as a (red) connection.

[0038] The above describes one approach to determining whether to create or

repurpose connections, particularly high-cost connections. In accordance with other embodiments and implementations, other approaches can be used. Also, as described above, the labels/colors are provided for purposes of illustration; in accordance with other embodiments different types of labels can be used to distinguish between different connection types.

Connection Labeling with Multi-Tenant Environments

[0039] In accordance with an embodiment, a system and method for connection labeling for use with connection pools, can include support for cloud-based multi-tenant environments using connection labeling. In accordance with an embodiment, this type of environment can be considered an "Application as a Service" (AaaS) environment.

[0040] **Figure 6** illustrates a system for connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling, in accordance with an embodiment. As shown in Figure 6, a multi-tenant cloud environment can include an application server / database environment 100, such as a Fusion Applications environment, that includes or provides access to a database 102, for use by multiple tenants or tenant applications 172, 174, 176, in a cloud-based environment 170. As further shown in Figure 6, the system also includes a connection pool logic 104, which controls the creation of objects in a connection pool 106.

[0041] Software applications, which are accessed by tenants via the cloud 171, may initialize connections 178 retrieved from a connection pool before using the connection. As described above, examples of initialization include simple state re-initializations that require method calls within the application code, or more complex initializations including database operations that require round trips over a network, and the cost of these latter types of initialization may be significant. As also described above, labeling connections allows an application to attach arbitrary name/value pairs to a connection, and the application can then request a connection with a desired label from the connection pool. By associating particular labels with particular connection states, an application can retrieve an already initialized connection from the pool and avoid the time and cost of re-initialization. Again, connection labeling does not impose any meaning on user-defined keys or values; the meaning of any user-defined keys and values is defined solely by the application.

[0042] For example, as shown in Figure 6, the connection pool can include a plurality of connections that are currently in use 108, here indicated as connections A 112 and B 114; and can also include a plurality of connections that are idle 110, here indicated as connections C 116, D 118, E 120, F 122, G 124 and N 126. Each of the connections can be similarly labeled, in this illustration as (Blue) or (Green), and again these labels/colors are provided for purposes of illustration; in accordance with various embodiments different types

of labels can be used to distinguish between different connection types.

[0043] **Figure 7** further illustrates a system for connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling, in accordance with an embodiment.

[0044] In accordance with an embodiment, if a software application 130 wishes to make a request on the database, using a particular type of connection, for example a (Red) connection, it can make a getConnection(Red) request 180. In response, the connection pool logic will either create a new (Red) connection, or repurpose an existing idle connection from (Blue or Green) to (Red).

[0045] In accordance with an embodiment, the connection pool includes support for the application to use a configure ( ) callback to specify a "SET CONTAINER" or to set a container, to repurpose a particular connection from one tenant to another, which has the effect of switching the tenant on a particular database connection.

[0046] In accordance with an embodiment, each software application can utilize a cost function callback to provide configuration information 136 that defines, for that application, costs associated with repurposing connections, and additional configuration information such as high-cost connections and threshold values.

[0047] For example, a particular application may consider the cost of repurposing a (Blue) connection to a (Red) connection to have a value of 50; and the cost of repurposing a (Green) connection to a (Red) connection to have a value of 80; that a high-cost connection has a value of 70; and that a reasonable threshold is 10. The meanings of these values are similarly defined by the application, and the above are provided for purposes of illustration. In accordance with various embodiments different numeric or non-numeric values can be used to distinguish between different connection costs.

[0048] In accordance with an embodiment, the connection pool logic iterates over each connection available in the pool. For each connection, it calls a cost method. The result of the cost method is an integer which represents an estimate of the cost required to reconfigure the connection to the required state. The larger the value, the costlier it is to reconfigure the connection.

[0049] In accordance with an embodiment, the configuration information provided by the application can be used by the connection pool logic in determining whether to create or repurpose connections, particularly high-cost connections. For example, in accordance with an embodiment, the system can perform a process similar to that illustrated above.

[0050] Using the above illustration, in accordance with an embodiment, a particular application may define High-Cost to be 70, and Threshold to be 10. When the system receives a request for a particular connection type (e.g., Red), the connection pool logic first determines if there is a perfect/existing match (i.e., an idle Red connection), and if so returns

that connection for use by the application. Else, the connection pool logic finds the cheapest connection that can be repurposed (to be a Red connection). If the cost of the cheapest connection is less than High-Cost (70), then that connection is repurposed. Else, if the total number of connections is less than Threshold (10), then a new (Red) connection is created, here indicated as Z 184 (Red), labeled accordingly, and provided to the application. Else, if the total number of connections is greater than or equal to Threshold, then the cheapest connection is repurposed as a (Red) connection, here indicated as E 182 (Red).

[0051] In accordance with an embodiment, when the total number of active and idle connections is low, a request to use a high-cost connection of a particular type may result in a new high-cost connection being created, rather than an existing (potentially also high-cost) connection begin repurposed. The new type connection can then be used for subsequent requests of that type.

[0052] Although the proposed approach may result in a high-cost connection being created, rather than an existing (potentially also high-cost) connection being repurposed, the approach can provide considerable performance improvements, particularly in complex, e.g., multi-tenant cloud environments, which generally utilize high-cost connections.

[0053] For example, as shown in Figure 7, the system can be used by multiple tenants or tenant applications in a cloud-based environment. In such a multi-tenant environment, connection types may be very complex, due to the need to accommodate multiple tenants, and, e.g., to maintain security between each different tenant's access to the database. Using the approach described herein, the performance of applications operating within a cloud environment can be improved.

[0054] **Figure 8** is a flowchart that illustrates a process of connection labeling for use with connection pools, including support for cloud-based multi-tenant environments using connection labeling, in accordance with an embodiment. As shown in Figure 8, in accordance with an embodiment, at step 192, a multi-tenant cloud environment is provided, that includes or provides access to a database, for use by multiple tenants or tenant applications.

[0055] At step 194, a connection pool is provided, which provides connections that software application can use to make requests to access the database, wherein connections can be labeled according to the configuration of particular applications.

[0056] At step 196, a software application is configured to support connection requests from the multiple tenants or tenant applications, together with connection labeling and connection cost information.

[0057] At step 198, connections labeled as high-cost connections are identified, and the system thereafter controls the creation or repurposing of high-cost connections to serve requests from the multiple tenants or tenant applications

Example Implementations

[0058] Provided below are illustrative examples of how connection labeling can be used with connection pools in an Oracle UCP environment, in accordance with various embodiments. In accordance with other embodiments, functionality can be provided, e.g., for use with WebLogic server connection pools, or other types of connection pools.

[0059] In Oracle UCP, Connection Labeling (CL) provides a mechanism to identify High-Cost Connections. In accordance with an embodiment, CL must support at least a discrete value; and may support a range of values. CL can provide a Reuse High-Cost Connection Threshold configuration parameter (similar to minpoolsize, maxpoolsize). When the lowest-cost available connection is a High-Cost Connection, the system can test the current pool size against Reuse High-Cost Connection Threshold, and against minimum pool size. If current < minimum or current < threshold, the system returns a new connection. Else if (current >= threshold), the system returns the lowest-cost High-Cost Connection. When there are no available connections, the current behavior holds (i.e., return a new connection, subject to maximum pool size, and if >= maximum pool size, wait for a connection to be available, subject to the timeout, etc.).

[0060] In accordance with an embodiment, the UCP Connection Labeling behavior UCP's Connection Labeling feature supports the cost( ) method in a Connection Labeling callback implementation, for any application to determine the cost of initializing and reinitializing a connection in the pool. The pool supports flexible cost value range that can be fully customized to application requirements. The pool uses the cost( ) value returned from the callback to determine the best candidate connection to serve each connection request. It always picks the connection with the lowest cost value. The lowest cost value being 0 indicates no reinitialization, while Integer.MAX_VALUE forces the pool to use a brand new physical connection to serve the request. The pool distinguishes between connections with applied labels and without labels. When Connection Labeling is activated, the pool always checks connections in the pool with applied labels first, and only when it cannot find any available labeled connection to serve the request, it attempts to find an available one from the connections without labels. When this fails, it attempts to create a new physical connection if the pool still has room to grow.

[0061] In accordance with various embodiments, variations on the above can include:

[0062] Add a new UCP pool property ConnectionLabelingHighCost (also available on UCP data sources PoolxxxDataSource). When the set value is greater than 0, connections with a cost value equal to or greater than the property value are considered "high-cost" connections. The default value is Integer.MAX_VALUE. For example, if the property value is set to 5, any connection whose calculated cost value from the labeling callback is equal to

or greater than 5 is considered a High-Cost connection.

[0063] Add a new UCP pool property HighCostConnectionReuseThreshold (also available on UCP data sources PoolxxxDataSource). When the set value is greater than 0, this specifies a threshold of the number of total connections in the pool beyond which Connection Labeling is allowed to reuse High-Cost connections in the pool to serve a request. Below this threshold, Connection Labeling either uses an available low-cost connection, or creates a brand-new physical connection to serve a request. For example, if the property value is set to 20, Connection Labeling reuses High-Cost connections when there are no low-cost connections available and the total connections reach 20. The default value for HighCostConnectionReuseThreshold is 0. A Connection Labeling callback must be registered at the same time for this property to take effect. Valid Connection Labeling callback registration continues to activate Connection Labeling. The pooling logic checks for the new threshold after the cost-selection iteration, when the lowest cost result is equal to or greater than ConnectionLabelingHighCost. The number of total connections at the moment when this new threshold is checked should account for the number of active connection creation requests (the pool already has code to extract this information). The checking must account for both MinPoolSize and MaxPoolSize. Note that any labeled connection with cost value Integer.MAX_VALUE will not be reused, even after the new threshold is reached. This is consistent with existing connection labeling behavior when the new threshold and ConnectionLabelingHighCost are not set.

[0064] In accordance with an embodiment, there is no requirement not to reuse connections without labels (stateless) in the pool to serve connection requests with labels (i.e., labeled requests). Once the HighCostConnectionReuseThreshold is reached and Connection Labeling is activated, the pool still favors connections without labels (stateless) over creating new physical connections.

[0065] In accordance with an embodiment, to support a special Connection Labeling callback implementation, for any connection the application considers High-Cost, the cost( ) method in such callback (1) simply returns Integer.MAX_VALUE for such connection, before the pool size reaches the HighCostConnectionReuseThreshold, and (2) switches to return an actual High-Cost value after the threshold is reached. This effectively prohibits existing UCP code from reusing a High-Cost connection to serve a request, below the threshold. The callback implementation can dynamically check the pool size against the threshold.

[0066] Embodiments of the present invention may be conveniently implemented using one or more conventional general purpose or specialized digital computer, computing device, machine, or microprocessor, including one or more processors, memory and/or computer readable storage media programmed according to the teachings of the present

disclosure. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.

[0067] In some embodiments, the present invention includes a computer program product which is a non-transitory storage medium or computer readable medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. Examples of the storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[0068] The foregoing description of 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 the practitioner skilled in the art. 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.

Claim:

What is claimed is:

1. A system for connection labeling for use with connection pools, comprising:

a computer including a processor and a database;

a connection pool, including a plurality of connection objects which provide connections that software applications can use to make requests to access the database, wherein each of the connections can be labeled according to the configuration of particular applications; and

a connection pool logic that identifies connections labeled as high-cost connections, and controls one or more of the use, creation, or repurposing of high-cost connections to serve requests from the software applications.

2. The system of claim 1, wherein the system is adapted for use by multiple tenants or tenant applications in a multi-tenant cloud-based environment, and wherein each particular one of the multiple tenants or tenant applications can be associated with a labeled connection type, which connection type the particular applications use to connect to the database for that particular tenant.

3. The system of any of claims 1-2, wherein the system is adapted for use by multiple tenants or tenant applications in a multi-tenant cloud-based environment, and wherein the connection pool includes support for the particular applications to use a configure callback to specify or set a container, to repurpose a particular connection from one of the multiple tenants or tenant applications to another of the multiple tenants or tenant application, which has the effect of switching the tenant on a particular database connection.

4. The system of any of claims 1-3, wherein the connection pool logic performs a process of

determining, for a received request, if there is an existing matching connection; and

if there is an existing matching connection, returning the existing matching connection, otherwise

finding a cheapest non-matching connection and determining whether the cheapest non-matching connection cost is less than a high-cost value, and if so repurposing the cheapest non-matching connection for use with the request, otherwise

if the sum of all connections is less than a threshold value, creating a

new connection for use with request, and returning the new connection, and

if the sum of all connections is greater than or equal to the threshold value, repurposing the cheapest connection for use with the request.

5. The system of any of claims 1-4, wherein the connection pool logic identifies connections labeled as high-cost connections, and avoids using those high-cost connections to serve requests when the total number of connections is below a particular threshold value.

6. The system of any of claims 1-5, wherein the connection pool logic is configured such that, when the total number of active and idle connections is low, a request to use a high-cost connection of a particular type results in a new high-cost connection being created, rather than an existing connection begin repurposed.

7. A method for connection labeling for use with connection pools, comprising:

providing, at a computer including a processor and a database, a multi-tenant cloud environment that includes or provides access to the database, for use by multiple tenants or tenant applications in a cloud-based environment;

providing a connection pool, including a plurality of connection objects which provide connections that software applications can use to make requests to access the database, wherein each of the connections can be labeled according to the configuration of particular applications; and

using a connection pool logic that identifies connections labeled as high-cost connections, and controls the creation or repurposing of high-cost connections to serve requests from the multiple tenants or tenant applications.

8. The method of claim 7, wherein the method is adapted for use by multiple tenants or tenant applications in a multi-tenant cloud-based environment, and wherein each particular one of the multiple tenants or tenant applications can be associated with a labeled connection type, which connection type the particular applications use to connect to the database for that particular tenant.

9. The method of any of claims 7-8, wherein the method is adapted for use by multiple tenants or tenant applications in a multi-tenant cloud-based environment, and wherein the connection pool includes support for the particular applications to use a configure callback to specify or set a container, to repurpose a particular connection from one of the multiple tenants or tenant applications to another of the multiple tenants or tenant application, which has the effect of switching the tenant on a particular database connection.

10. The method of any of claims 7-9, wherein the connection pool logic performs a process of

determining, for a received request, if there is an existing matching connection; and

if there is an existing matching connection, returning the existing matching connection, otherwise

finding a cheapest non-matching connection and determining whether the cheapest non-matching connection cost is less than a high-cost value, and if so repurposing the cheapest non-matching connection for use with the request, otherwise

if the sum of all connections is less than a threshold value, creating a new connection for use with request, and returning the new connection, and

if the sum of all connections is greater than or equal to the threshold value, repurposing the cheapest connection for use with the request.

11. The method of any of claims 7-10, wherein the connection pool logic identifies connections labeled as high-cost connections, and avoids using those high-cost connections to serve requests when the total number of connections is below a particular threshold value.

12. The method of any of claims 7-11, wherein the connection pool logic is configured such that, when the total number of active and idle connections is low, a request to use a high-cost connection of a particular type results in a new high-cost connection being created, rather than an existing connection begin repurposed.

13. A non-transitory computer readable medium, including instructions stored thereon which when read and executed by one or more computers cause the one or more computers to perform the method comprising:

providing, at a computer including a processor and a database, a multi-tenant cloud environment that includes or provides access to the database, for use by multiple tenants or tenant applications in a cloud-based environment;

providing a connection pool, including a plurality of connection objects which provide connections that software applications can use to make requests to access the database, wherein each of the connections can be labeled according to the configuration of particular applications; and

using a connection pool logic that identifies connections labeled as high-cost connections, and controls the creation or repurposing of high-cost connections to serve requests from the multiple tenants or tenant applications.

14. The method of claim 13, wherein the method is adapted for use by multiple tenants or tenant applications in a multi-tenant cloud-based environment, and wherein each particular one of the multiple tenants or tenant applications can be associated with a labeled connection type, which connection type the particular applications use to connect to the database for that particular tenant.

15. The method of any of claims 13-14, wherein the method is adapted for use by multiple tenants or tenant applications in a multi-tenant cloud-based environment, and wherein the connection pool includes support for the particular applications to use a configure callback to specify or set a container, to repurpose a particular connection from one of the multiple tenants or tenant applications to another of the multiple tenants or tenant application, which has the effect of switching the tenant on a particular database connection.

16. The method of any of claims 13-15, wherein the connection pool logic performs a process of

determining, for a received request, if there is an existing matching connection; and

if there is an existing matching connection, returning the existing matching connection, otherwise

finding a cheapest non-matching connection and determining whether the cheapest non-matching connection cost is less than a high-cost value, and if so repurposing the cheapest non-matching connection for use with the request, otherwise

if the sum of all connections is less than a threshold value, creating a new connection for use with request, and returning the new connection, and

if the sum of all connections is greater than or equal to the threshold value, repurposing the cheapest connection for use with the request.

17. The method of any of claims 13-16, wherein the connection pool logic identifies connections labeled as high-cost connections, and avoids using those high-cost connections to serve requests when the total number of connections is below a particular threshold value.

18. The method of any of claims 13-17, wherein the connection pool logic is configured such that, when the total number of active and idle connections is low, a request to use a high-cost connection of a particular type results in a new high-cost connection being created, rather than an existing connection begin repurposed.

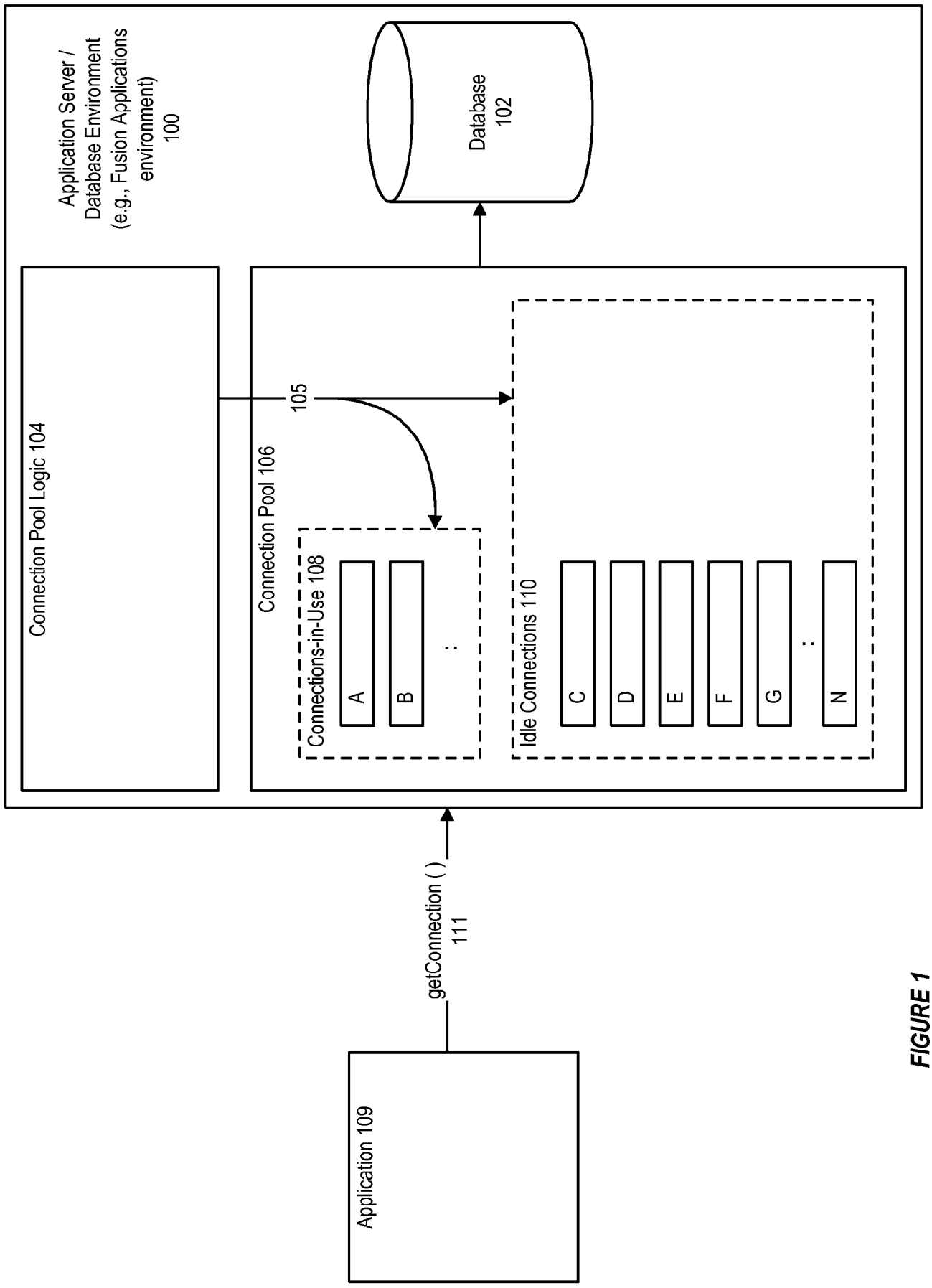
Application Server / Database Environment (e.g., Fusion Applications environment) 100
Database 102
Connection Pool Logic 104
Connection Pool 106
105
Connections-in-Use 108
A
B
Idle Connections 110
C
D
E
F
G
N
getConnection ( ) 111
Application 109

FIGURE 1

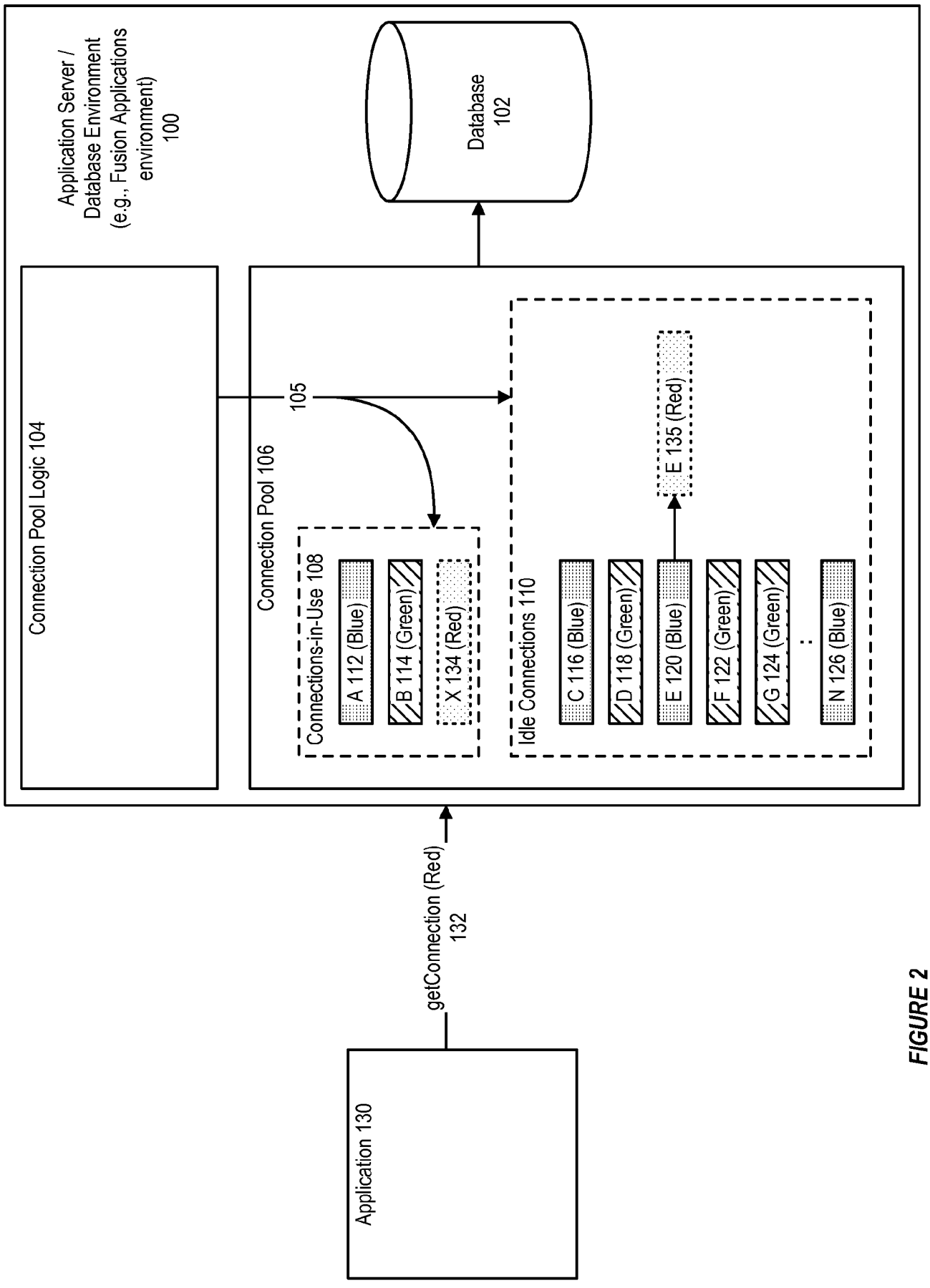
Application Server / Database Environment (e.g., Fusion Applications environment) 100
Database 102
Connection Pool Logic 104
Connection Pool 106
105
Connections-in-Use 108
A 112 (Blue)
B 114 (Green)
X 134 (Red)
Idle Connections 110
C 116 (Blue)
D 118 (Green)
E 120 (Blue)
E 135 (Red)
F 122 (Green)
G 124 (Green)
N 126 (Blue)
getConnection (Red) 132
Application 130

FIGURE 2

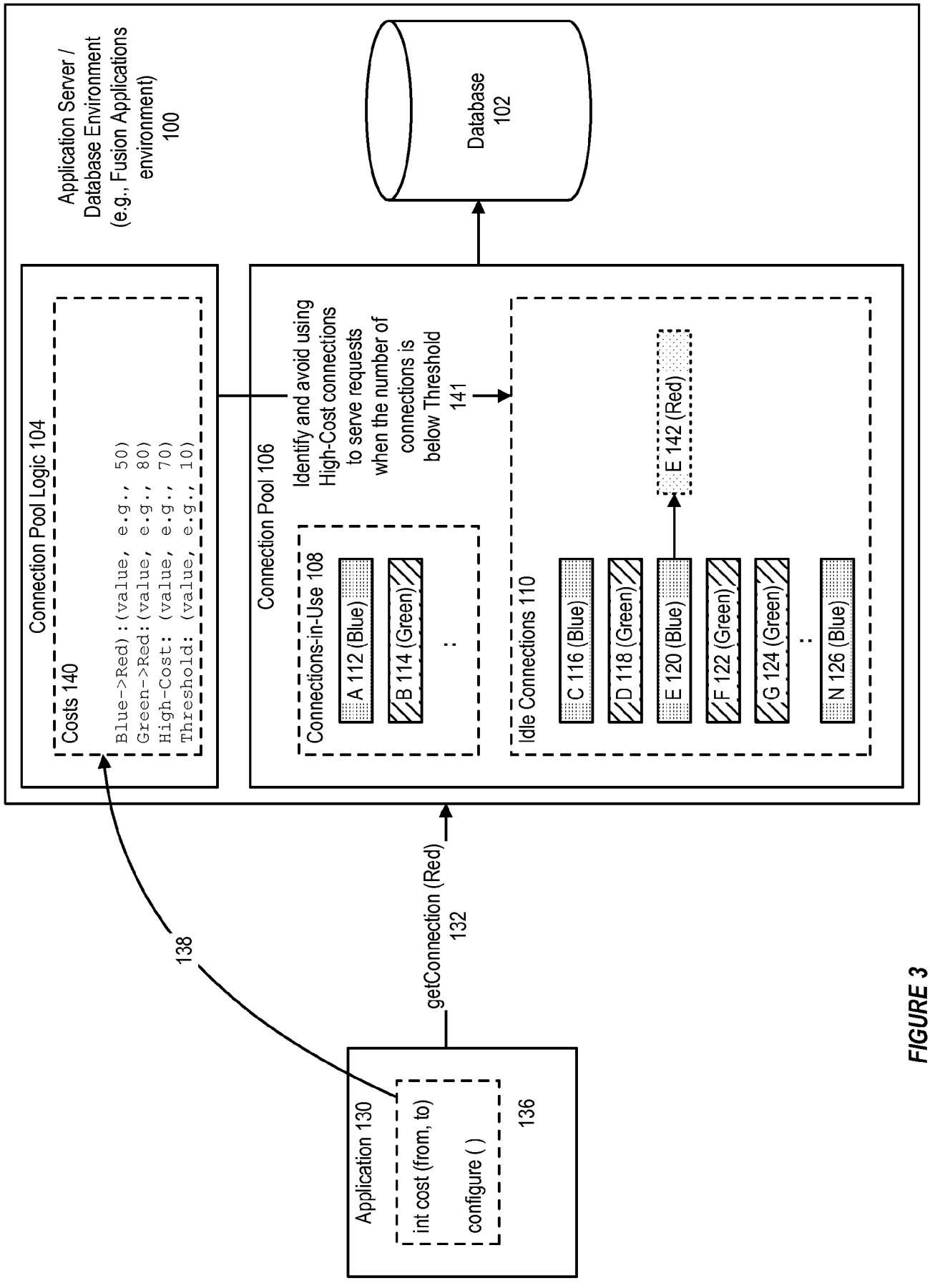
Application Server / Database Environment (e.g., Fusion Applications environment) 100
Database 102
Connection Pool Logic 104
Costs 140
Blue->Red):(value, e.g., 50)
Green->Red:(value, e.g., 80)
High-Cost: (value, e.g., 70)
Threshold: (value, e.g., 10)
Connection Pool 106
Identify and avoid using High-Cost connections to serve requests when the number of connections is below Threshold 141
Connections-in-Use 108
A 112 (Blue)
B 114 (Green)
Idle Connections 110
C 116 (Blue)
D 118 (Green)
E 120 (Blue)
E 142 (Red)
F 122 (Green)
G 124 (Green)
N 126 (Blue)
138
getConnection (Red) 132
Application 130
int cost (from, to)
configure ( )
136

FIGURE 3

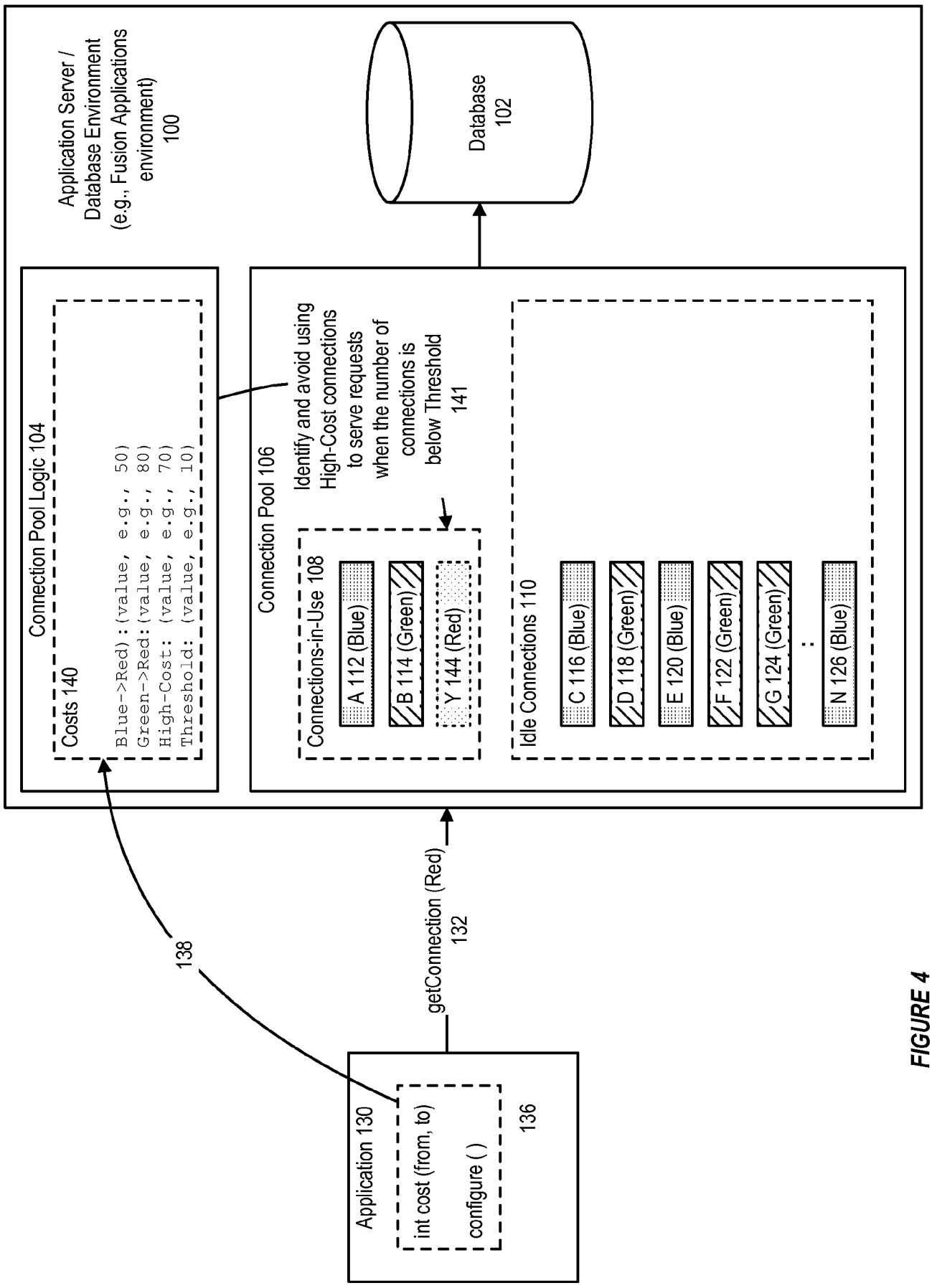
Application Server / Database Environment (e.g., Fusion Applications environment) 100
Database 102
Connection Pool Logic 104
Costs 140
Blue->Red):(value, e.g., 50)
Green->Red:(value, e.g., 80)
High-Cost: (value, e.g., 70)
Threshold: (value, e.g., 10)
Connection Pool 106
Identify and avoid using High-Cost connections to serve requests when the number of connections is below Threshold 141
Connections-in-Use 108
A 112 (Blue)
B 114 (Green)
Y 144 (Red)
Idle Connections 110
C 116 (Blue)
D 118 (Green)
E 120 (Blue)
F 122 (Green)
G 124 (Green)
N 126 (Blue)
138
getConnection (Red) 132
Application 130
int cost (from, to)
configure ( )
136

FIGURE 4

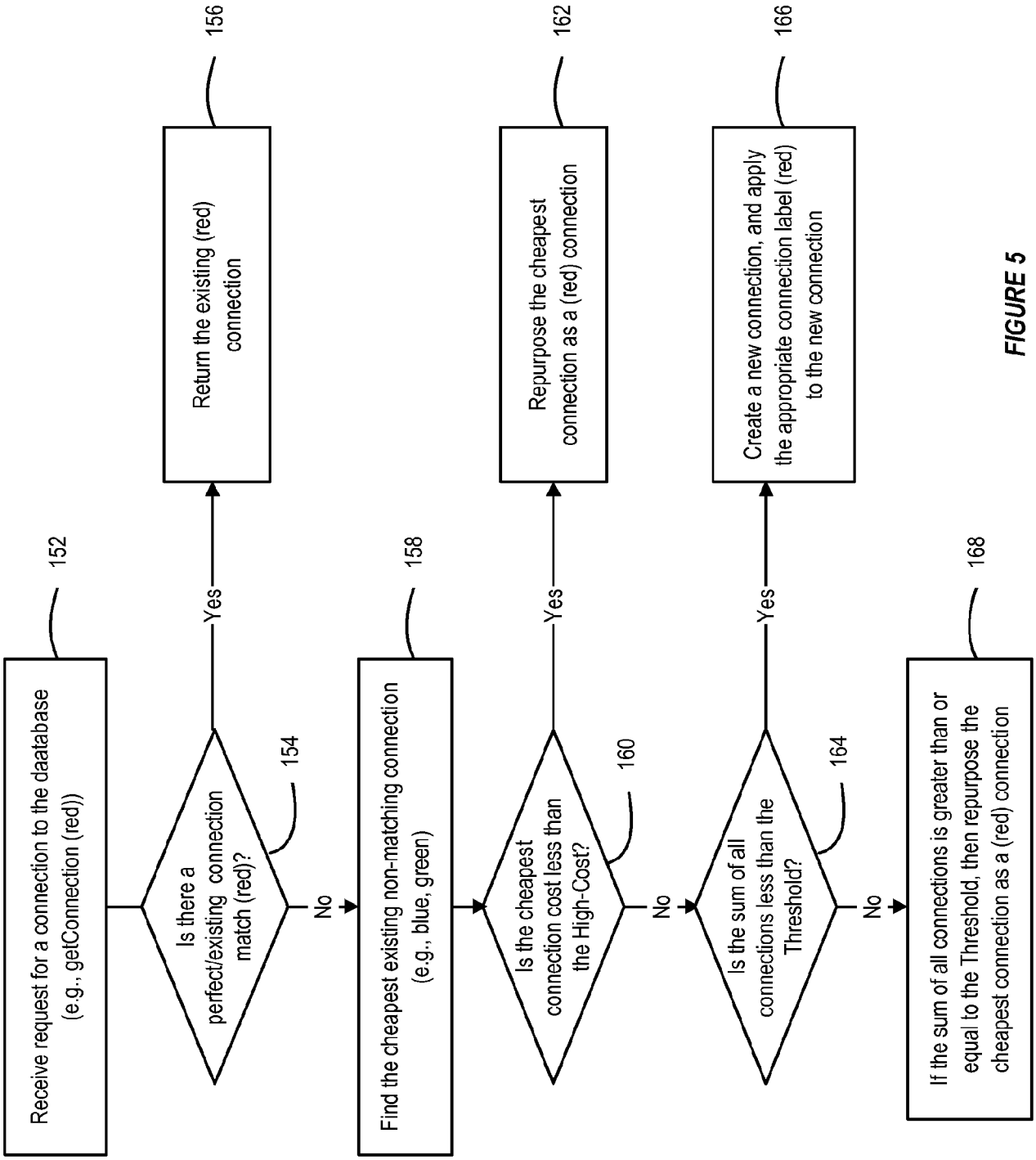
Receive request for a connection to the daatabase (e.g., getConnection (red))
152
Is there a perfect/existing connection match (red)?
154
Yes
Return the existing (red) connection
156
No
Find the cheapest existing non-matching connection (e.g., blue, green)
158
Is the cheapest connection cost less than the High-Cost?
160
Yes
Repurpose the cheapest connection as a (red) connection
162
No
Is the sum of all connections less than the Threshold?
164
Yes
Create a new connection, and apply the appropriate connection label (red) to the new connection
166
No
If the sum of all connections is greater than or equal to the Threshold, then repurpose the cheapest connection as a (red) connection
168

FIGURE 5

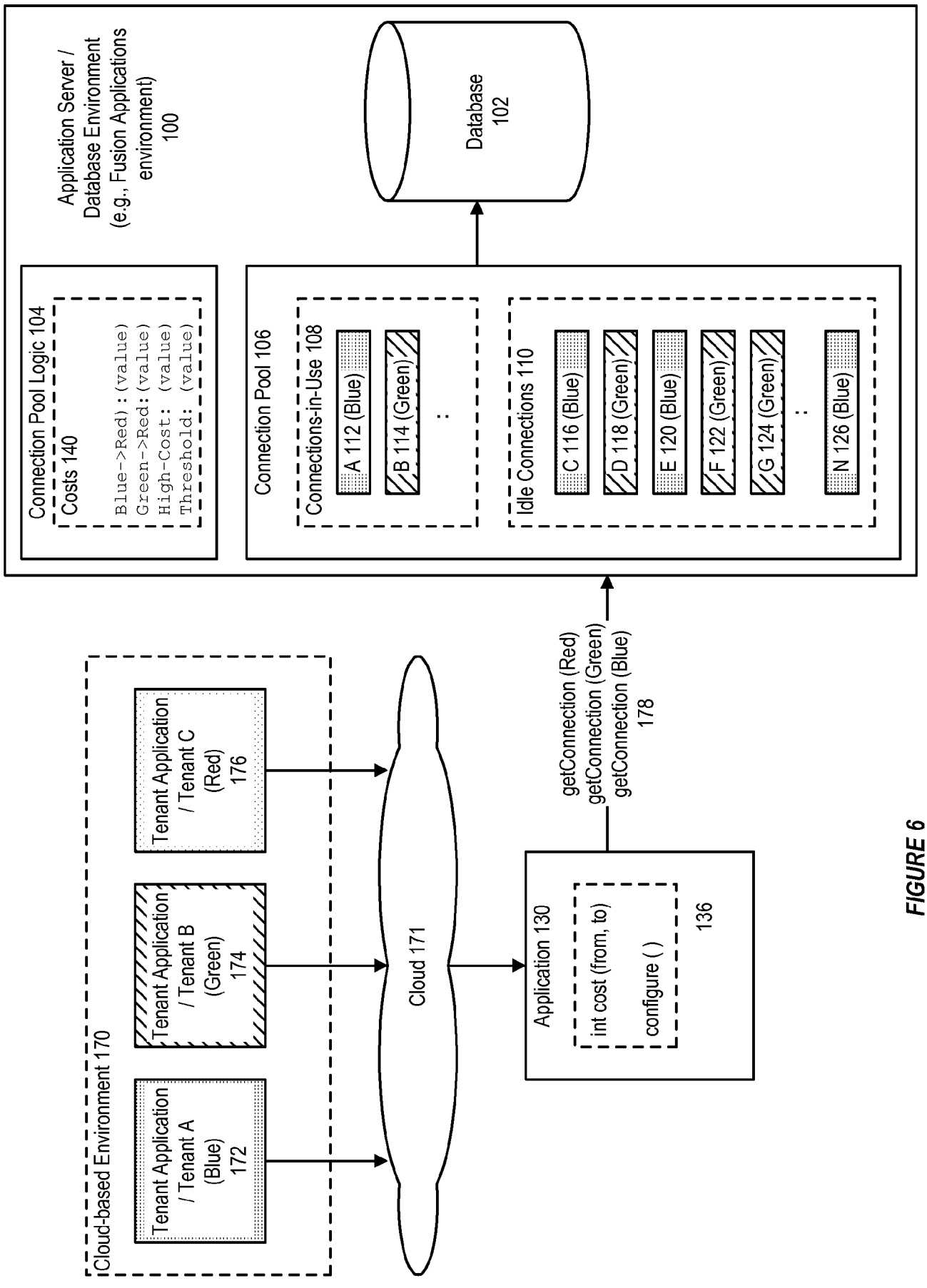
Application Server / Database Environment (e.g., Fusion Applications environment) 100
Database 102
Connection Pool Logic 104
Costs 140
Blue->Red):(value)
Green->Red:(value)
High-Cost: (value)
Threshold: (value)
Connection Pool 106
Connections-in-Use 108
A 112 (Blue)
B 114 (Green)
Idle Connections 110
C 116 (Blue)
D 118 (Green)
E 120 (Blue)
F 122 (Green)
G 124 (Green)
N 126 (Blue)
getConnection (Red)
getConnection (Green)
getConnection (Blue)
178
Cloud-based Environment 170
Tenant Application / Tenant A (Blue) 172
Tenant Application / Tenant B (Green) 174
Tenant Application / Tenant C (Red) 176
Cloud 171
Application 130
int cost (from, to)
configure ( )
136

FIGURE 6

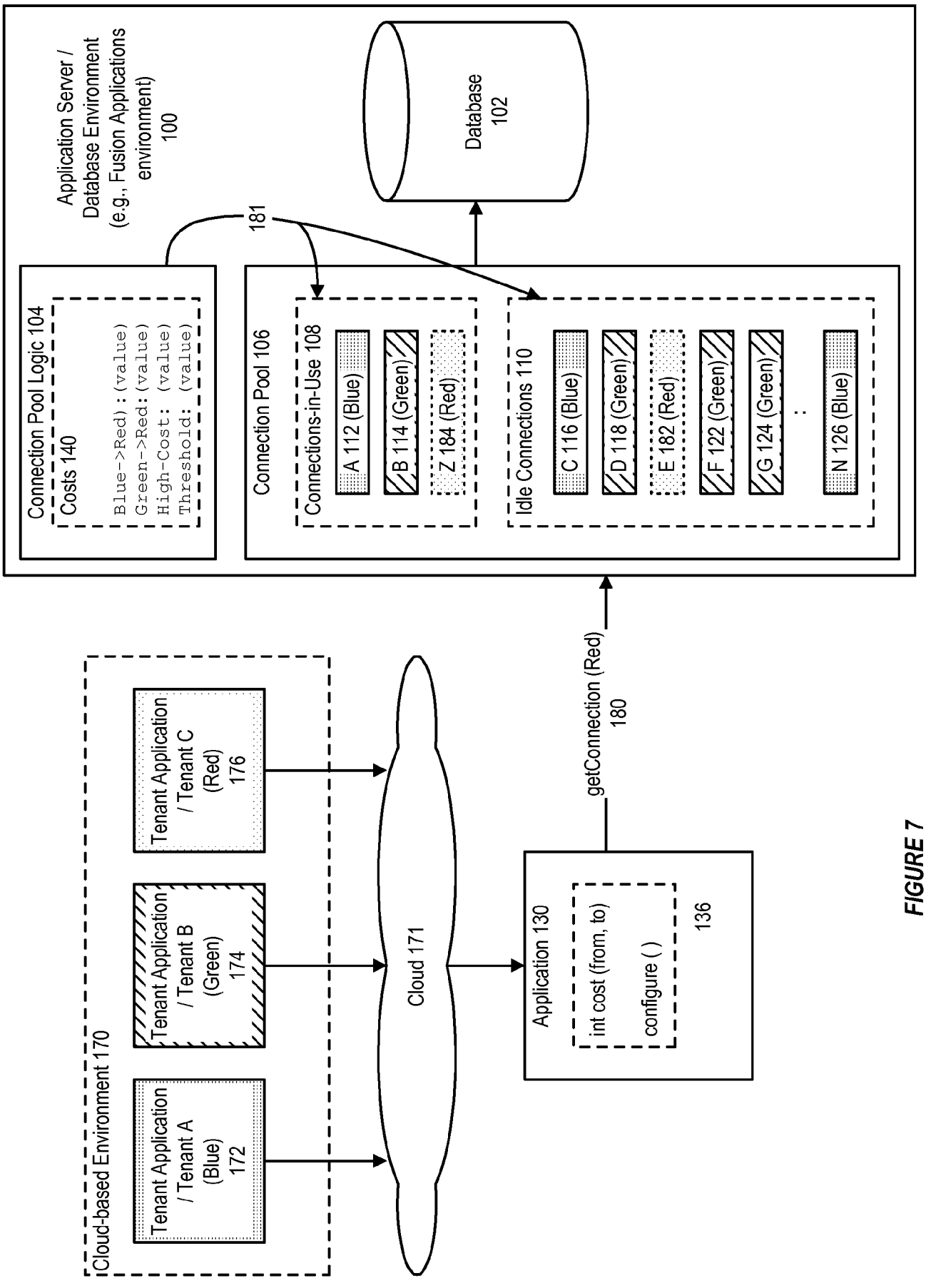
Application Server / Database Environment (e.g., Fusion Applications environment) 100
Database 102
181
Connection Pool Logic 104
Costs 140
Blue->Red):(value)
Green->Red:(value)
High-Cost: (value)
Threshold: (value)
Connection Pool 106
Connections-in-Use 108
A 112 (Blue)
B 114 (Green)
Z 184 (Red)
Idle Connections 110
C 116 (Blue)
D 118 (Green)
E 182 (Red)
F 122 (Green)
G 124 (Green)
N 126 (Blue)
getConnection (Red) 180
Cloud-based Environment 170
Tenant Application / Tenant A (Blue) 172
Tenant Application / Tenant B (Green) 174
Tenant Application / Tenant C (Red) 176
Cloud 171
Application 130
int cost (from, to)
configure ( )
136

FIGURE 7

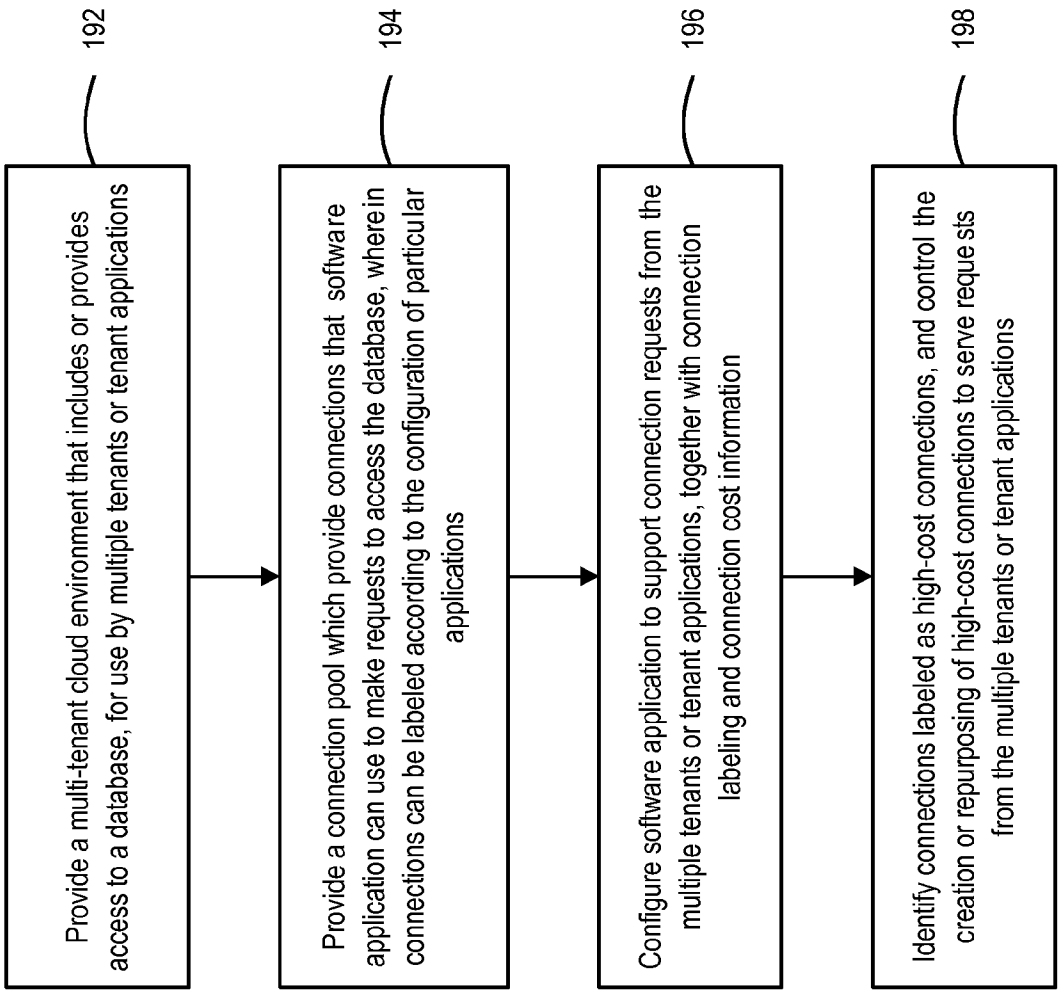
192
Provide a multi-tenant cloud environment that includes or provides access to a database, for use by multiple tenants or tenant applications
194
Provide a connection pool which provide connections that software application can use to make requests to access the database, wherein connections can be labeled according to the configuration of particular applications
196
Configure software application to support connection requests from the multiple tenants or tenant applications, together with connection labeling and connection cost information
198
Identify connections labeled as high-cost connections, and control the creation or repurposing of high-cost connections to serve requests from the multiple tenants or tenant applications

FIGURE 8

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/035187

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/50
ADD.

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)
G06F

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)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2009/064199 A1 (BIDELIS SIGITAS [US] ET AL BIGELIS SIGITAS [US] ET AL) 5 March 2009 (2009-03-05) the whole document	1-18
A	WO 2006/073865 A2 (BEA SYSTEMS INC [US]; SRIVASTAVA RAHUL [US]) 13 July 2006 (2006-07-13) the whole document	1-18
A	WO 2012/037163 A1 (ORACLE INT CORP [US]; SOMOGYI ALEX [US]; REVANURU NARESH [US]; IRUDAYA) 22 March 2012 (2012-03-22) the whole document	1-18

-/--

[X] Further documents are listed in the continuation of Box C.

[X] 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 but 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
29 July 2014	05/08/2014

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Beyer, Steffen

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2014/035187

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ANONYMOUS: "Database Connection Pool Management", RESEARCH DISCLOSURE, MASON PUBLICATIONS, HAMPSHIRE, GB, vol. 41, no. 416, 1 December 1998 (1998-12-01), XP002141842, ISSN: 0374-4353 the whole document -----	1-18

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2014/035187

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009064199 A1	05-03-2009	NONE	
WO 2006073865 A2	13-07-2006	AU 2005323039 A1	13-07-2006
		EP 1844395 A2	17-10-2007
		JP 2008525916 A	17-07-2008
		KR 20070110011 A	15-11-2007
		WO 2006073865 A2	13-07-2006
WO 2012037163 A1	22-03-2012	CN 103124967 A	29-05-2013
		EP 2616966 A1	24-07-2013
		JP 2013541764 A	14-11-2013
		US 2012066363 A1	15-03-2012
		WO 2012037163 A1	22-03-2012