Tuning database connections

Original page = https://confluence.atlassian.com/adminjiraserver070/tuning-database-connections-749382655.html

JIRA uses a database connection pool, based on Apache Commons DBCP (DataBase Connection Pool), to manage JIRA's access to its underlying database.

In earlier JIRA versions, the database connection pool was handled purely through the Apache Tomcat application server running JIRA. However, from JIRA version 4.4, JIRA's `dbconfig.xml`file provides a set of database connection pool settings to Tomcat, which in turn are used by Tomcat to manage JIRA's database connection pool. From JIRA version 5.1, the number database connection pool settings defined in JIRA's `dbconfig.xml` file substantially increased.

The information on this page can help you tweak JIRA's database connection pool settings. You can do this by using the JIRA configuration tool or by directly editing JIRA's `dbconfig.xml` file, as described below.

The **Advanced** tab of the JIRA Configuration Tool makes it easier to both configure and control JIRA's database connection pool. The Database monitoring page (accessible to JIRA system administrators) provides a visual tool for monitoring JIRA's database connection usage.

Connection pool architecture

Whenever JIRA needs to access (i.e. read from or write to) its database, a database connection is required.

A database connection is a large and complex object that handles all communication between JIRA and its database. As such, database connections are time consuming to establish and consume a significant amount of memory on both the client (the JIRA application) and database server.

To avoid the impact of creating a new database connection for each database access request made by JIRA, a pool of pre-established database connections is maintained. Each new database access request made by JIRA uses a connection from this pool of pre-established connections, as required. Hence:

1. When JIRA starts up, a minimum number of database connections are established in the pool between JIRA and its database.
2. When JIRA needs to access its database, JIRA:
   a. requests a database connection from the pool
   b. uses this database connection to read from and/or write to its database
   c. returns the database connection to the pool when finished.

If the frequency of JIRA's database access requests begin to exceed the number of available database connections in the pool, extra connections are automatically created to handle the load.

Conversely, if the frequency of JIRA's database access requests begin to drop below the number of available database connections in the pool, connections can be automatically closed to release resources back to the system.

Modern databases can handle large numbers of connections relatively easily and with sufficient memory, many hundred. On the client side, however, these connections can consume a significant amount memory. Hence, it is generally best to limit the number of connections to a much smaller number while having a sufficient number for the application to rarely need to wait for a connection when it needs one.

Tuning JIRA's database connections

1. Shut down your JIRA installation.
2. Do either of the following:
   - Use the JIRA configuration tool to tune JIRA's database connections.
     a. Start the JIRA configuration tool:
        - **Windows**: Open a command prompt and run `config.bat` in the `bin`sub-directory of the JIRA installation directory.
        - **Linux/Unix**: Open a console and execute `config.sh` in the `bin` sub-directory of the JIRA installation directory.
          This may fail with the error as described in our Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB article. Please refer to it for the workaround.

     Please Note: You may need to set the `JAVA_HOME` environment variable to run the JIRA configuration tool. See Installing Java for details.
     b. Once the JIRA configuration tool is running, click its **Advanced** tab.
     blocked URL
     c. Refer to Connection pool settings below for more information about the options on this tab. To specify a value for one of these options, ensure its leftmost checkbox has been selected first.
     Some options above are simple checkboxes (i.e. in the centre of the JIRA configuration tool). Selecting these checkboxes sets the values of their associated options to 'true'. Conversely, clearing these checkboxes sets the values of their associated options to 'false'.
     d. Click the **Save** button to save your changes, which will be stored as elements in your `dbconfig.xml` file.
   - Alternatively, edit the `dbconfig.xml` file at the root of your JIRA home directory.
     a. Refer to Connection pool settings below for more information about the elements you can add to your `dbconfig.xml` file to fine tune JIRA's database connection.

b. Save your edited `dbconfig.xml` file.
3. Restart your JIRA installation.

Connection pool settings

JIRA configuration tool'Advanced' tab option	Element in dbconfig.xml	Explanation	Recommendations / Notes	Default value*
Maximum Size	`pool-max-size`	The maximum number of database connections that can be opened at any time.	This value should be sufficiently large enough that JIRA rarely needs to wait for a database connection to become available when JIRA requires one. See Monitoring below for suggestions on how to set this parameter.	20
Maximum Idle	`pool-max-idle`	The maximum number of database connections that are allowed to remain idle in the pool.	Specifying a negative number sets no limit on the number of database connections that can remain idle. If the value of **Minimum Idle/Size**(below) is the same as that of **Maximum Size**(above), which is the case by default, then this setting has no effect.	Value of **Maximum Size**
Minimum Idle /Size	`pool-min-size` (`min-idle`)	The minimum number of idle database connections that are kept open at any time.	Having this value set to that of **Maximum Size** (above), which is the case by default, means the pool will always have a fixed number of connections and idle connections will never be closed. On very large JIRA installations, there may be some benefit in specifying a lower value for this setting than that of **Maximum Size**, to conserve resources.	Value of **Maximum Size**
Initial Size	`pool-initial-size`	The initial number of database connections opened in the pool.	This setting is not usually configured (other than the default value of 0), since a number of database connections are quickly created when JIRA starts up.	0 (when not specified in `dbconfig.xml`)
Maximum Wait Time	`pool-max-wait`	The length of time (in milliseconds) that JIRA is allowed to wait for a database connection to become available (while there are no free ones available in the pool), before returning an error.	Specifying a value of '-1' means that Tomcat will wait indefinitely. You should specify a time here which is long enough to allow for any contention spikes, but short enough that users will receive a meaningful error rather than just getting no response or a browser time out.	30000
Advanced settings Generally, changing the settings below are not usually required. Refer to the Apache DBCP documentation if required.				
Pool Statements	`pool-prepared-statements`	Enable the pooling of prepared statements for the database connection pool.	Do **not** amend the default value of false, as it will cause exceptions. For more information see blocked URLJRASERVER-44908 - DBPC configuration pool-prepared-statements leads to Statement Leak CLOSED	false (when not specified in `dbconfig.xml`)
Maximum Open Statements	`max-open-prepared-statements`	The maximum number of open statements that can be allocated from the statement pool at the same time.	Do **not** amend the default value, as it will cause exceptions.	0 (when not specified in `dbconfig.xml`)
Validation Query	`validation-query`	The SQL query that will be used to validate connections from this pool. If specified, this query MUST be an SQL SELECT statement that returns at least one row.	See Surviving connection closuresfor more information.	select 1 (for MySQL) (otherwise, not specified in `dbconfig.xml`)
Validation Query Timeout	`validation-query-timeout`	The length of time (in seconds) that the system should wait for a validation query to succeed before it considers the database connection broken.	The length of time should be quite short as the validation query should be designed to do a minimum amount of work. If you specify a **Validation Query** above, then you should specify a value for the **Validation Query Timeout**too. If not, a value of '-1' is assumed, which results in the system waiting indefinitely until a validation query succeeds against a broken database connection, which it never will. ⚠ This should only be done for MySQL. Using a **Validation Query Timeout** on any database other than MySQL will cause significant problems with the JIRA instance.	3 (for MySQL) (otherwise, not specified in `dbconfig.xml`)
Test On Borrow	`pool-test-on-borrow`	Tests if the database connection is valid when it is borrowed from the database connection pool by JIRA. If the database connection is broken, it is removed from the pool.	This value should always be 'false' as JIRA borrows a connection for each database operation. ✓ If you continue to have problems with database connections closing, try setting this option to 'true'. However, this should only be used as a last resort and only in the event that decreasing the value of **Time Between Eviction Runs**has not reduced or prevented problems with database connections closing.	True (when not specified in `dbconfig.xml`), however this does not take effect unless a Validation Query has been explicitly specified, except for MySQL, which has a default Validation Query, and it will therefore have an effect.
Test On Return	`pool-test-on-return`	Tests if the database connection is valid when it is returned to the database connection pool by JIRA. If the database connection is broken, it is removed from the pool.	This value should always be 'false' as JIRA returns borrowed connections for each database operation.	false (when not specified in `dbconfig.xml`)

Test While Idle	`pool-test-while-idle`	Periodically tests if the database connection is valid when it is idle. If the database connection is broken, it is removed from the pool.	This should be set to 'true' for MySQL. By default, MySQL database servers close database connections if they are not used for an extended period of time. This causes problems with JIRA installations (which use MySQL databases) that are largely inactive for long periods, e.g. overnight. Setting this to 'true' will work around this behavior. **Test While Idle**only needs to be specified if you have specified a **Validation Query** above.	true (for MySQL) false (when not specified in `dbconfig.xml`)
Time Between Eviction Runs	`time-between-eviction-runs-millis`	The number of milliseconds to sleep between runs of the idle object eviction thread. When non-positive, no idle object eviction thread will be run. The eviction thread will remove idle database connections when the number of idle connections exceeds **Minimum Idle/Size**(above).	This should be set to a positive but largish value for MySQL so the evictor runs and tests connections. A reasonable value would be 300000 (5 minutes). If you continue to have problems with database connections closing, try setting this option to a lower value.	300000 (for MySQL) 5000 (for HSQLDB) (otherwise, not specified in `dbconfig.xml`)
Minimum Evictable Idle Time	`min-evictable-idle-time-millis`	The minimum amount of time an object may sit idle in the database connection pool before it is eligible for eviction by the idle object eviction (if any).		60000 (for MySQL) 4000 (for HSQLDB) (otherwise, not specified in `dbconfig.xml`)
Remove Abandoned	`pool-remove-abandoned`	Flag to remove abandoned database connections if they exceed the **Removed Abandoned Timeout**(below). If an internal failure occurs, it is possible that JIRA may borrow a connection and never return it. If this happens too often, then the pool may run short of database connections, causing JIRA's performance to degrade or JIRA to fail altogether.	This value should be set to 'true'. This will allow the pool to recover any abandoned connections and prevent this affecting system performance.	true
Remove Abandoned Timeout	`pool-remove-abandoned-timeout`	The length of time (in seconds) that a database connection can be idle before it is considered abandoned.		300

* **Please note:**

- JIRA writes elements with their default values (in the right-hand column of the table above) to the `dbconfig.xml` file after:
  - You have run through the JIRA setup wizard or
  - You use the **Advanced** tab of the JIRA configuration tool to configure/tune your database connection — even when the leftmost checkboxes of options associated with these elements have not been selected.
- The exception to this are elements whose values have '(when not specified in `dbconfig.xml`)' indicated below them. These elements are:
  - Not written to the `dbconfig.xml` file after running through the JIRA setup wizard.
  - Only written to the `dbconfig.xml` file by:
    - Manually writing them into this file.
    - Using the Advanced tab of the JIRA configuration tool, selecting the leftmost checkboxes of the options associated with these elements and specifying values for these options.
- When '(when not specified in `dbconfig.xml`)' is indicated below a default value in the right-hand column of the table above, then this default value is assumed, even when it is not present in the `dbconfig.xml` file.

Monitoring the connection pool

JIRA provides a view of its database connection usage via the 'Database Monitoring' page. See Monitoring database connection usage for more information.