

This is a tutorial on how to use Confluence as a query / reporting engine, querying SQL data sources like the Jira database. For our example we query JIRA's database to build a Monthly Worklogs Report, showing hours worked per day for every user in a given month. We use the free [Play SQL Base](#) plugin.

Of course, [Tempo Timesheets](#) is the de-facto plugin for this sort of thing, and already has a report like what we're building:

Tempo's report is prettier and more powerful, allowing hours to grouped by any field (e.g. project, or tempo Account), even hierarchically. Tempo's one deficiency here, which motivated this reimplementation, is that **it cannot show users which have not logged any work**. Tempo's also honors Tempo's [evil 'View All Worklogs' permission](#), which I consider an anti-feature.

Implementation

Choosing a Confluence SQL plugin

For this tutorial we are using the free [Play SQL Base](#) plugin. You could alternatively use [PocketQuery](#) or [SQL for Confluence](#), which are in fact better plugins overall - in particular, they let you restrict who can run SQL queries, whereas Play SQL can't.

This tutorial uses Play SQL Base because it's what I had available. We will restrict SQL queries at the Postgres layer, which is a good thing to do anyway.

Configure Play SQL Base

In Confluence, type 'gg', 'Find new apps' and install the free [Play SQL Base](#) plugin.

In Confluence spaces you will now see a new 'Tables' menu item. Here is the page from a live Confluence instance, with various queries already defined (there's one from the [Automatically deactivating inactive Jira users](#) report):

The screenshot shows the Confluence interface with the 'Queries' page selected. The left sidebar includes a user profile for Jeff Turner and a 'Tables' menu item. The main content area is divided into three sections: 'Queries', 'Charts', and 'Database'. The 'Queries' section lists several queries with expand/collapse icons and delete buttons. The 'Charts' section lists three charts. The 'Database' section provides links to manage connections, monitor queries, and view database info. The footer indicates the page is powered by Atlassian Confluence 7.1.0.

Queries (Create new...)	
▼ Inactive Users	✕
▼ Inactive Users	✕
▼ csat_stats_breakdown_parametrized	✕
▼ worklog_monthly	✕
▼ redradish_issuecount	✕
▼ active_jira_users	✕
▼ lostcomments	✕
▼ redradish_worksummary	✕

Charts (Create new...)	
▲ redradish_issuecount	✕
▲ redradish_hours_worked	✕
▲ redradish_hours_credit	✕

Database

[Manage Connections and Permissions](#)
[Monitor Running Queries](#)
[Database info](#)

Powered by Atlassian Confluence 7.1.0 · [Report a bug](#) · [Atlassian News](#)

ATLASSIAN

Click 'Manage Connections and Permissions' and set up the space's database connection. Here we just use the global datasource:

Confluence

Spaces

People

Create



Jeff Turner

★

Profile

Pages

Blog

Tables

SPACE SHORTCUTS

Here you can add shortcut links to the most important content for your team or project. [Configure sidebar.](#)

Space Tools

OverviewPermissionsContent ToolsLook and FeelIntegrationsPlay SQL

DatasourcesPermissionsAudit Trail

<< General Admin configuration

Datasource

Type

☒ Use global (See General Admin)

☐ Use a JNDI datasource (recommended)

☐ Use connection details

☐ Disable for this space

Check

SaveCancel

Clicking 'General Admin' shows the global config:

Confluence

Spaces

People

Create

Search

Confluence administration

CONFIGURATION

General Configuration

Further Configuration

Backup Administration

Source Editor

Languages

Shortcut Links

External Gadgets

Global Templates and Blueprints

Recommended Updates Email

Mail Servers

User Macros

In-app Notifications

Hipchat Integration

Spam Prevention

PDF Export Language Support

Configure Code Macro

ATLASSIAN MARKETPLACE

Find new apps

Manage apps

Play SQL Add-on

Play SQL Setup

OverviewSettingsDatabase

Here is a summary of your datasources. For help, [see the documentation](#). Press Next when you're done.

Context	Type	Details	
Default Values	DIRECT	Embedded database-in-a-file. Details...	
Global Connection	JNDI	Dialect com.playsql.jdbc.dialect.PostgreSQLDDLialect Mode READ_ONLY JNDI java:comp/env/jdbc/QueriesDS Schema queries Test select 1	Edit

Next

For information - Available dialects on this instance:

	Read-Only	Read-Write	Autocomplete	Monitoring
PostgreSQL	✓	✓	✓	✓
HSQLDB	✓	✓	✓	✗
MySQL Unsupported	✓ *	✗	✗	✗
Generic	✓ *	✗	✗	✗
Oracle Unsupported	✓ *	✗	✗	✗

* This dialect has not been verified as "complete" by Play SQL.

Creating a Postgres read-only account

At this point we're about to tell Play SQL how to connect to our database. For the sake of security, we want to connect as a user with **read-only** permission s, and with **visibility restricted** to just data necessary for our report.

The **read-only** requirement can be achieved with Postgres permissions. The **restricted visibility** requirement can be achieved by only allowing queries of predefined views, in a custom queries schema. The main Jira tables in the public schema will be inaccessible.

First, create a 'queries' schema, with a sample view containing a small amount of data:

```
root@jturner-desktop:~# su - postgres
postgres@jturner-desktop:~$ psql redradish_jira
Null display is "".
Line style is unicode.
Border style is 2.
psql (12.2 (Ubuntu 12.2-4))
Type "help" for help.

redradish_jira=# CREATE SCHEMA IF NOT EXISTS queries;
CREATE SCHEMA
redradish_jira=# CREATE OR REPLACE VIEW queries.sample AS select project.pkey || '-' || jiraissue.issuenum AS
key, summary from public.project JOIN public.jiraissue ON project.id=jiraissue.project LIMIT 5;
CREATE VIEW
redradish_jira=# select * from queries.sample;

      key      | summary
-----+-----
SOC-3         | A second Response for good measure
ML-53         | Ongoing Atlassian Product Support, 2014
IC-34         | Invoice 93236 - 1/Jul/15 to 30/Sep/15
JTOD0-19      | Tax Payment Q2 Due
CLIC-2        | Move projects to OnDemand

(5 rows)
```

Next, create a `jira_queries_readonly` role that can only view the queries schema tables, and a `confluence_reports` user granted that role. These commands are cribbed shamelessly from <https://blog.redash.io/postgres-readonly/>, so read that to understand them properly. Run them when connected to the Jira database, *not* the default 'postgres' database.

```
CREATE ROLE jira_queries_readonly;
GRANT CONNECT ON DATABASE redradish_jira TO jira_queries_readonly;
GRANT USAGE ON SCHEMA queries TO jira_queries_readonly;
GRANT SELECT ON ALL TABLES IN SCHEMA queries TO jira_queries_readonly;
CREATE USER confluence_reports WITH PASSWORD 'confluence_reports';
GRANT jira_queries_readonly TO confluence_reports;
```

Verify that, when connecting as `confluence_reports` we can see our sample query but not generic Jira tables:

```
# PGUSER=confluence_reports PGPASSWORD=confluence_reports PGHOST=localhost PGDATABASE=redradish_jira psql -tAc
"select count(*) from queries.sample;"
5
# PGUSER=confluence_reports PGPASSWORD=confluence_reports PGHOST=localhost PGDATABASE=redradish_jira psql -tAc
"select count(*) from public.jiraissue;"
ERROR: permission denied for table jiraissue
```

Define a Datasource in Confluence

There are two ways to tell Play SQL (and other SQL plugins) how to connect to a database:

- A **direct** connection - the plugin will contact the database directly, given a hostname, port, username and password
- A **JNDI/Datasource** connection - the plugin will ask Confluence's middleware (the Tomcat application server) for a preconfigured database connection

Either way will work. I used a datasource, defined as the `jdbc/QueriesDS` section in my `/opt/atlassian/confluence/conf/server.xml` file:

```

    <Engine name="Standalone" defaultHost="localhost" debug="0">
      <Host name="localhost" debug="0" appBase="webapps" unpackWARs="true" autoDeploy="false"
startStopThreads="4">
        <Context path="" docBase="../../confluence" debug="0" reloadable="false" useHttpOnly="true">
          <Resource name="jdbc/ConfluenceDS" auth="Container" type="javax.sql.DataSource"
            username="confluence"
            password="<REDACTED>"
            driverClassName="org.postgresql.Driver"
            url="jdbc:postgresql://localhost:5432/confluence"
            maxTotal="20"
            validationQuery="select 1"/>
          <Resource name="jdbc/QueriesDS" auth="Container" type="javax.sql.DataSource"
            username="confluence_reports"
            password="confluence_reports"
            driverClassName="org.postgresql.Driver"
            url="jdbc:postgresql://localhost:5432/jira?currentSchema=queries"
            maxTotal="20"
            validationQuery="select 1"/>

          <!-- Logging configuration for Confluence is specified in confluence/WEB-INF/classes/log4j.
properties -->
            <!-- Uncomment this to DISABLE session serialization.
            <Manager pathname=""/>
            -->
            <Valve className="org.apache.catalina.valves.StuckThreadDetectionValve" threshold="60"/>
          </Context>

          <Context path="`${confluence.context.path}/synchrony-proxy" docBase="../../synchrony-proxy" debug="
0"
            reloadable="false" useHttpOnly="true">
            <Valve className="org.apache.catalina.valves.StuckThreadDetectionValve" threshold="60"/>
          </Context>
        </Host>
      </Engine>

```

You will need to restart Confluence to pick up this change.

- It's more secure - database credentials aren't stored as plaintext in the database or in innumerable backups.
- it lets you configure the 'QueriesDS' differently in production vs. sandbox. The database hostname for Jira might be different on the sandbox server. Rather than reconfigure PlaySQL every time you sync sandbox data, you configure 'QueriesDS' once correctly in the sandbox conf /server.xml.
- the app server can provide stats about database connection use via JMX or [JavaMelody](#).
- It's just conceptually nicer (the inversion of control principle).

Configure PlaySQL with the Datasource

To recap, we've just been on a detour to create a read-only Postgres account, and edited Confluence's conf/server.xml file to define our QueriesDS datasource.

Now configure Play SQL to use the Datasource. Here I've configured QueriesDS as our default 'global connection':

Play SQL Setup

Overview Settings Database

Here is a summary of your datasources. For help, [see the documentation](#). Press Next when you're done.

Context	Type	Details
Default Values	DIRECT	Embedded database-in-a-file. Details...

Global Connection

Use JNDI connection

Dialect

PostgreSQL

JNDI Name*

java:comp/env/jdbc/QueriesDS

Example: (jndi) java:comp/env/jdbc/userdatasource

Mode

Read-Only (Query mode)

Initialization SQL

Executed each time a connection is opened

Schema

queries

Executed after Init SQL. Creates a schema for read-write connections (for example: SPACE_[PERSONAL_]\$spaceKey|CONTEXT_\$name).

Close SQL

Executed before closing the connection

Test SQL

select 1

Example: VALUES(1); [Test now.](#)

Create a test Play SQL Table

Now return to the 'Tables' tab in a space:

Confluence

Spaces

People

Create

...

Search

?

⚙

🔊

Jeff Turner

★

Profile

Pages

Blog

Tables

SPACE SHORTCUTS

Here you can add shortcut links to the most important content for your team or project. [Configure sidebar.](#)

Queries

Queries (Create new...)

Inactive Users

InactiveUsers

csat_stats_breakdown_parametrized

worklog_monthly

redradish_issuecount

active_jira_users

lostcomments

redradish_worksummary

Charts

Charts (Create new...)

redradish_issuecount

redradish_hours_worked

redradish_hours_credit

Database

Manage Connections and Permissions

Monitor Running Queries

Database info

Powered by Atlassian Confluence 7.1.0 · [Report a bug](#) · [Atlassian News](#)

ATLASSIAN

Under 'Queries' click 'Create new...'.

Now query your sample view and click 'Preview' to verify it works:

Jeff Turner

Profile

Pages

Blog

Calendars

Tables

SPACE SHORTCUTS

How-to articles

sample

1

select * from queries.sample

internetexpenses

- ID
- POSITION
- workratio
- date
- info
- invoicenumber
- billtotal
- phone
- fixedcost

Options >>

Wizards

?

Cancel

Preview

Save

...	key	summary
1	SOC-3	A second Response for good measure
2	ML-53	Ongoing Atlassian Product Support, 2014
3	IC-34	Invoice 93236 - 1/Jul/15 to 30/Sep/15
4	JTODO-19	Tax Payment Q2 Due
5	CLIC-2	Move Clickability projects to OnDemand
Click here to add totals		



Did we mention Play SQL Base is free? It is free, but also buggy, and at this point the bugs are very evident:

- The list of queryable tables on the right may or may not be correct. In the screenshot above it reflects an unrelated 'playsql' schema, not 'queries'.
- SQL queries can't end with a semi-colon, or you'll get an error
- Clicking 'Save' on a newly defined query, as you will now want to do, results in an error:

Red Radish Wiki

Spaces

People

Calendars

Create

...

com.playsql.base.web.QueryEditorAction2.action.name

An error was encountered

The following error(s) occurred:

- The space doesn't exist

Press the 'Back' button of your browser to come back to the previous screen.

If this page doesn't provide enough information to solve your issue

Please submit a support request to playsql+support@gmail.com or search the following channels:

- View the [most recent Q&A](#) about Play SQL
- Ask on [Atlassian's Q&A](#) (with the Play SQL tag)

But don't worry, your query did save.

If you persevere, it does work in the end. Don't complain - the Play SQL author makes his money from [Play SQL Spreadsheets](#), not Play SQL Base - we're fortunate to have a free, roughly functional plugin at all.

Create the timesheets database view

So far we've successfully queried `queries.sample`. We now create a `queries.worklog_monthly` view containing our real timesheet data.

We're not going to dwell too much on the specifics of our query. Here it is:

```
-- A giant table of worklog hours per day, for each day of the month, selectable by user, year and month
-- See https://www.redradishtech.com/display/KB/Creating+interactive+Jira+reports+in+Confluence+using+free+tools
create schema if not exists queries;
create or replace view queries.worklog_monthly AS
select * from (
  select user_name, email_address, year, month
    , round(sum(sum),2) AS month_total
    ,case sum("1") when 0 then 0 else round(sum("1"),2) end AS "1"
    ,case sum("2") when 0 then 0 else round(sum("2"),2) end AS "2"
    ,case sum("3") when 0 then 0 else round(sum("3"),2) end AS "3"
    ,case sum("4") when 0 then 0 else round(sum("4"),2) end AS "4"
    ,case sum("5") when 0 then 0 else round(sum("5"),2) end AS "5"
    ,case sum("6") when 0 then 0 else round(sum("6"),2) end AS "6"
    ,case sum("7") when 0 then 0 else round(sum("7"),2) end AS "7"
    ,case sum("8") when 0 then 0 else round(sum("8"),2) end AS "8"
    ,case sum("9") when 0 then 0 else round(sum("9"),2) end AS "9"
    ,case sum("10") when 0 then 0 else round(sum("10"),2) end AS "10"
    ,case sum("11") when 0 then 0 else round(sum("11"),2) end AS "11"
    ,case sum("12") when 0 then 0 else round(sum("12"),2) end AS "12"
    ,case sum("13") when 0 then 0 else round(sum("13"),2) end AS "13"
    ,case sum("14") when 0 then 0 else round(sum("14"),2) end AS "14"
    ,case sum("15") when 0 then 0 else round(sum("15"),2) end AS "15"
    ,case sum("16") when 0 then 0 else round(sum("16"),2) end AS "16"
    ,case sum("17") when 0 then 0 else round(sum("17"),2) end AS "17"
    ,case sum("18") when 0 then 0 else round(sum("18"),2) end AS "18"
    ,case sum("19") when 0 then 0 else round(sum("19"),2) end AS "19"
    ,case sum("20") when 0 then 0 else round(sum("20"),2) end AS "20"
    ,case sum("21") when 0 then 0 else round(sum("21"),2) end AS "21"
    ,case sum("22") when 0 then 0 else round(sum("22"),2) end AS "22"
    ,case sum("23") when 0 then 0 else round(sum("23"),2) end AS "23"
    ,case sum("24") when 0 then 0 else round(sum("24"),2) end AS "24"
    ,case sum("25") when 0 then 0 else round(sum("25"),2) end AS "25"
    ,case sum("26") when 0 then 0 else round(sum("26"),2) end AS "26"
    ,case sum("27") when 0 then 0 else round(sum("27"),2) end AS "27"
    ,case sum("28") when 0 then 0 else round(sum("28"),2) end AS "28"
    ,case sum("29") when 0 then 0 else round(sum("29"),2) end AS "29"
    ,case sum("30") when 0 then 0 else round(sum("30"),2) end AS "30"
    ,case sum("31") when 0 then 0 else round(sum("31"),2) end AS "31"
  from (
    select user_name, email_address, year, month, day, sum
      , case day when 1 then sum else 0 end AS "1"
      , case day when 2 then sum else 0 end AS "2"
      , case day when 3 then sum else 0 end AS "3"
      , case day when 4 then sum else 0 end AS "4"
      , case day when 5 then sum else 0 end AS "5"
      , case day when 6 then sum else 0 end AS "6"
      , case day when 7 then sum else 0 end AS "7"
      , case day when 8 then sum else 0 end AS "8"
      , case day when 9 then sum else 0 end AS "9"
      , case day when 10 then sum else 0 end AS "10"
      , case day when 11 then sum else 0 end AS "11"
      , case day when 12 then sum else 0 end AS "12"
      , case day when 13 then sum else 0 end AS "13"
      , case day when 14 then sum else 0 end AS "14"
      , case day when 15 then sum else 0 end AS "15"
      , case day when 16 then sum else 0 end AS "16"
      , case day when 17 then sum else 0 end AS "17"
      , case day when 18 then sum else 0 end AS "18"
      , case day when 19 then sum else 0 end AS "19"
      , case day when 20 then sum else 0 end AS "20"
      , case day when 21 then sum else 0 end AS "21"
      , case day when 22 then sum else 0 end AS "22"
      , case day when 23 then sum else 0 end AS "23"
      , case day when 24 then sum else 0 end AS "24"
      , case day when 25 then sum else 0 end AS "25"
      , case day when 26 then sum else 0 end AS "26"
      , case day when 27 then sum else 0 end AS "27"
      , case day when 28 then sum else 0 end AS "28"
```

```

, case day when 29 then sum else 0 end AS "29"
, case day when 30 then sum else 0 end AS "30"
, case day when 31 then sum else 0 end AS "31"
from (
    select
        user_name
        , email_address
        , extract(year from dte) AS year
        , extract(month from dte) AS month
        , extract(day from dte) AS day
        , sum(coalesce(timeworked,0))/60.0/60 AS sum
    from
        (select generate_series::date AS dte from generate_series('2019-01-01'::date,
now()::date, '1 day')) alldays
    FULL OUTER JOIN cwd_user
    ON (true)
    INNER JOIN app_user
    USING (lower_user_name)
    LEFT JOIN
    public.worklog
    ON
        worklog.author = app_user.user_key AND
        to_char(dte, 'YYYY-MM-DD') = to_char(worklog.startdate, 'YYYY-MM-DD')
    WHERE cwd_user.active=1
    -- and email_address ~ '(redradishtech\.com)$' -- Optionally filter to just
workloggable users here.
    group by (user_name, email_address, year, month, day)
) y
) z group by rollup((user_name, email_address), year, month)
) q
order by month_total desc
;
grant select on queries.worklog_monthly to jira_queries_readonly;

```

I suggest creating a directory in your Confluence app dir for SQL queries like this:

```

/opt/atlassian/jira # mkdir SQL_QUERIES
/opt/atlassian/jira # cd SQL_QUERIES/
/opt/atlassian/jira/SQL_QUERIES #

```

Then you can fetch the SQL directly using curl and run it to create the view in your database:

```

/opt/atlassian/jira/SQL_QUERIES # curl -sLOJ 'https://github.com/redradishtech/jira-interesting-sql-queries/raw
/master/worklog_monthly.sql'
/opt/atlassian/jira/SQL_QUERIES # sudo -u postgres psql redradish_jira -tAxq < worklog_monthly.sql

```

Verify that our confluence_reports user can read our new queries.worklog_monthly table:

```

# PGUSER=confluence_reports PGPASSWORD=confluence_reports PGHOST=localhost PGDATABASE=redradish_jira psql -tAc
"select count(*) from queries.worklog_monthly;"
121

```

Create a worklog_monthly Play SQL Table

As we did earlier for queries.sample, now configure a Table in Play SQL for our queries.worklog_monthly view.

You should first enter the query:

```
select * from worklog_monthly
```

Preview it to make sure that works. If so, parametrize it:

```
select * from queries.worklog_monthly where year='$year'::integer and month='$month'::integer and email_address ~ '$email'
```

Click 'Options >>' and configure the parameters:

The screenshot shows the Confluence Query Editor interface. At the top, the Confluence header is visible with the user 'Jeff Turner' and navigation links for Spaces, People, and Create. The main area displays a SQL query: `select * from queries.worklog_monthly where year='$year'::integer and month='$month'::integer and email_address ~ '$email'`. Below the query editor, the 'Options' tab is selected, showing various configuration settings. The 'Row numbers' dropdown is set to 'Generate'. The 'Timeout (ms)' field is empty, with a note that the default is 30,000 milliseconds. The 'Custom empty message' is set to 'No data'. The 'Cache' checkbox is checked, with a note that users can refresh the cache by running the query. The 'Parameters' section has the 'Replace built-in parameters in the query' checkbox unchecked, and the 'Display a form above the table (user parameters and cache)' checkbox checked. Under 'User Parameters', three parameters are defined: 'Year' (String to replace: \$year, Prompt: Year, Default value: 2019), 'Month' (String to replace: \$month, Prompt: Month, Default value: 12), and 'Email Address' (String to replace: \$email, Prompt: Email Address Regex: .*, Default value: .*). Below the parameters, a form displays the current values: Year: 2019, Month: 12, and Email Address: .*. At the bottom of the form are 'Run' and 'Refresh the cache' buttons. The bottom of the interface shows a table with columns: user_name, email_address, year, month, month_total, and a grid of numbered cells from 1 to 14.

Confluence Spaces People Create Search

Jeff Turner

worklog_monthly

```
1 select * from queries.worklog_monthly where year='$year'::integer and month='$month'::integer and email_address ~ '$email'
```

<< Options Wizards ?

Row numbers Generate

Timeout (ms)

Time in milliseconds (default is 30,000)

Custom empty message No data

The message when there is no data in the table. Default is: "No data in this table."

Cache ☒ Cache the results the first time the query runs. Users can refresh the cache by running in the Query Editor.

Parameters ☐ Replace built-in parameters in the query

Available parameters: \$userName, \$userFullName, \$email, \$creator, \$lastModifier, \$seoType, \$seoId, \$pageId, \$pageTitle, \$parentPageId, \$parentPageTitle, \$spaceKey, \$spaceName

☒ Display a form above the table (user parameters and cache)

User Parameters Add a User Parameter

String to replace: \$year Prompt: Year Default value: 2019

String to replace: \$month Prompt: Month Default value: 12

String to replace: \$email Prompt: Email Address Regex: .* Default value: .*

Year 2019

Month 12

Email Address .*

Run Refresh the cache

Space tools << user_name email_address year month month_total 1 2 3 4 5 6 7 8 9 10 11 12 13 14

You may want to tick the 'Cache' checkbox if you have a lot of data to query.

Create a page containing the table

Our final step is to create a page in the Confluence space, containing a Play SQL Query macro:

Confluence Spaces People Create ...

Paragraph B I U A ...

People / Jeff Turner / Dashboard / Jeff Turner's Home / Monthly Worklogs Report UNPUBLISHED CHANGES

Monthly Worklogs Report

Number of hours worked, per day, in a given year/month. Per Jira | IT-13760

Play SQL Query | worklog_monthly | space:~je...

Configure the macro to use the worklog_monthly query:

Edit 'Play SQL Query' Macro

Inserts and formats the results of a SQL query. [Documentation](#)

Choose a query *

worklog_monthly

[Edit the query...](#)

Output

RICH

The table can be displayed as a dynamic widget, a normal Confluence table or simple text. PRINT is used when printing.

Year

2020

Month

1

Email Address Regex:

Preview

Year 2020

Month 1

Email Address Regex: .*

Run Refresh the cache

...	user_name	email_address	year	month	mon
1			2020.0	1.0	
2			2020.0	1.0	
3			2020.0	1.0	

Select macro Save Cancel

and there you have it: our final worklog report:

