Queries

A query fetches data from one of your datasources. Its properties largely depend on the type of datasource being used.

Properties

All queries consist of the following properties:

- **Datasource:** The datasource on which the query is to be run (referencing a datasource entity). You can filter the list of queries by their datasources using the dropdown on the top left of the query list.
- **Name:** The name identifying the query. Names must start with a letter and only contain letters [A-Za-z_-], numbers, hyphens and underscores.
- **Template:** A template used to display the result of the query in the Confluence page containing the PocketQuery macro. References a template entity. If the *Default* template is selected, a default PocketQuery template is used.
- **Converter:** Converter used to transform the response of the datasource before it is handed on to the template. See the “Converters” section below.
- **Spaces:** A string specifying in what spaces and/or space categories this query should be available. This is a comma-separated list of spacekeys and space cagegories prefixed by "*category*" and a colon. Example: *spaceKey1, spaceKey2, category:mySpaceCategory*.
- **Cache for duration of:** A string specifying how long the results of this query will be cached. If this is left empty, nothing will be cached (results are always recent). The time is specified like “3d” (3 days), “12h” (12 hours), “50m” (50 minutes) - like in the work log of JIRA issues, for example. Caching can improve performance significantly, especially for more intense queries. Also see the section “Caching” for further information.
- **Add results to Confluence search:** Boolean value indicating if the results of query should be added to the Confluence index. This makes the results searchable by the Confluence search. Also see the section “Adding Results to the Search Index” for further information.

Depending on the type of the query, there are some more properties. Please refer to the sections below.

SQL Queries on JDBC datasources

Additional Properties

For JDBC datasources the following additional properties are available:

- **Statement:** The query statement. It consists of arbitrary SQL code, that forms exactly one single SQL statement. (If you need multiple statements, please use multiple queries or create a stored procedure at your database.) The code can optionally be enhanced by parameters prefixed by a colon in the form *:parameterName*. These parameter placeholders (“named parameters”) are filled during runtime of the PocketQuery Macro (see the example statements below for further explanation). It is possible to use wildcards within theses parameters (see section "Using Wildcards for Parameters" below for more information)
- **Parameter Types:** By default they are all strings. Currently also the types Integer, ListOfStrings, ListOfIntegers, Boolean and Constant are supported, which have to be specified explicitly. For each parameter, add its name on the left and from the select box on the right choose the correct type. Examples where this is required is the LIMIT clause (e.g. “*LIMIT :number*“) or the IN clause (e.g. “*:name IN (:myListOfStrings)*“). See the “Parameter Types” section below.

Parameter Types

By default, all parameters in a query will become strings. For example:

```
SELECT Name, Population
FROM Country
WHERE Continent = :continent
```

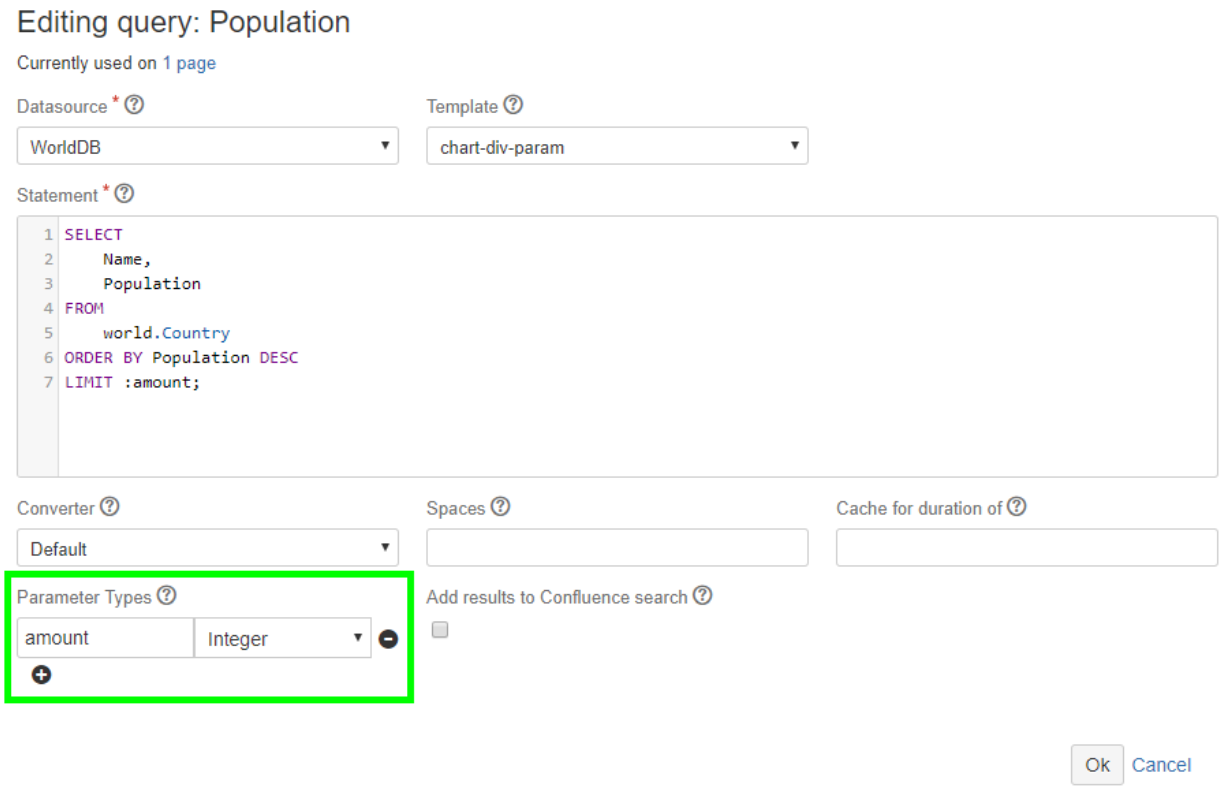
This will become something like the following after the replacement took place:

```
SELECT Name, Population
FROM Country
WHERE Continent = 'Europe'
```

Now in certain cases this behaviour is not intended. Examples are:

- Numbers
- Boolean values
- Lists for IN-clauses
- Constant values in the SELECT clause

That's why we implemented the *Parameter Types* configuration option for queries. You can configure types for all your query parameters in the section "Parameter Types" by inserting their names and selecting their respective types from the dropdown. If parameters are not specified the default value will be "String".



Currently supported parameter types:

- **String:** This is the default parameter type that does not need to be specified. Values provided for query parameters will be wrapped in single quotes.
- **Number:** Your numeric value will be inserted in the query with no quotes.
- **ListOfIntegers:** Your integer list will be inserted as comma-separated list of values. This is useful for IN-clauses with numbers, e.g. `... IN(1,2,3,4,5) ...`

- **ListOfStrings:** Your list of strings will be inserted as comma-separated list of values with single quotes. This is useful for IN-clauses with strings, e.g. `... IN('one','two','three') ...`
- **Boolean:** Your true/false value will be inserted without quotes.
- **Constant:** This is useful for parameters in the SELECT clause like `SELECT :mycolumn ...` where you simply want the value to be inserted as a constant.

Example query statements

```
SELECT Name, GovernmentForm, Region, HeadOfState
FROM Country
WHERE Continent = :continent;

SELECT Name
FROM Country
INNER JOIN CountryLanguage ON Country.Code = CountryLanguage.CountryCode
WHERE Language = :language;

SELECT *
FROM City
WHERE Population > :population;

SELECT Name, GovernmentForm, Region, HeadOfState
FROM Country
WHERE Population > :minPopulation AND Population < :maxPopulation;

SELECT Name
FROM Country
LIMIT :max;  -- "max" needs the parameter type "Integer"

SELECT Name, GovernmentForm
FROM Country
WHERE Name IN (:names);  -- "names" needs the parameter type
"ListOfStrings"
```

Queries on REST Datasources

Additional Properties

For REST datasources the following additional properties are available:

- **REST URL:** The URL path that is appended to the root URL of this datasource when calling it. It may contain parameters prefixed by a colon in the form *:parameterName*. These parameter placeholders (“named parameters”) are filled during runtime of the PocketQuery Macro. It is possible to use wildcards within theses parameters (see section "Using Wildcards for Parameters" below for more information). If special characters occur, they need to be url encoded (e.g. the percentage sign "%" is encoded as "%25", "ä" is encoded as "%C3%A4"). The structure of the REST URL depends entirely on the REST API used - please refer to its documentation.
- **JSON Path:** Path to the result within the JSON response. JSONPath expressions refer to a JSON structure in the same way as XPath is used in combination with an XML document. The $-sign stands for the virtual root, which is also the default value, if the field is left empty. Starting from there, field names, wildcards and indexes are used to navigate deeper. The child operator can be a dot (e.g. *$.store.book[0].title*) or square brackets (e.g. *$['store']['book'][0]['title']*). The most important operates are the following one, you can find a more detailed explanation and some examples in this article on JSONPath by Stefan Goessner.

JSONPath	Description

$	root element
@	current element
. or []	child operator
*	wildcard for all elements regardless their names
[<integer i>]	array operator for element at position i
?(<boolean expr>)	applies a filter expression

Using Parameters in Queries

As mentioned above, both SQL queries and REST calls may include PocketQuery parameters. These parameters can be set when adding the PocketQuery Macro to a page which makes the query much more versatile.

PocketQuery parameters are marked with a colon in front of their name (*:parameterName*). For SQL queries, it is possible to set the type of the parameter to influence how the final values will be included in the SQL query. See the section Parameter Types above for more details. For REST calls there are no different parameter types - all parameter values will be URL encoded and then included in the REST call URL.

Using Wildcards for Parameters

Instead of using fixed values, it is possible to use wildcards to set a PocketQuery parameter. These wildcards will be replaced by the actual content before the query is executed. The wildcards can be set by the user when adding the PocketQuery macro to a page, and since version 3.2.0 it is also possible to set wildcards directly within the query statement or REST URL itself. This way the macro user herself has no opportunity to adjust the values manually which can be important for security.

The following wildcards can be set as parameters values:

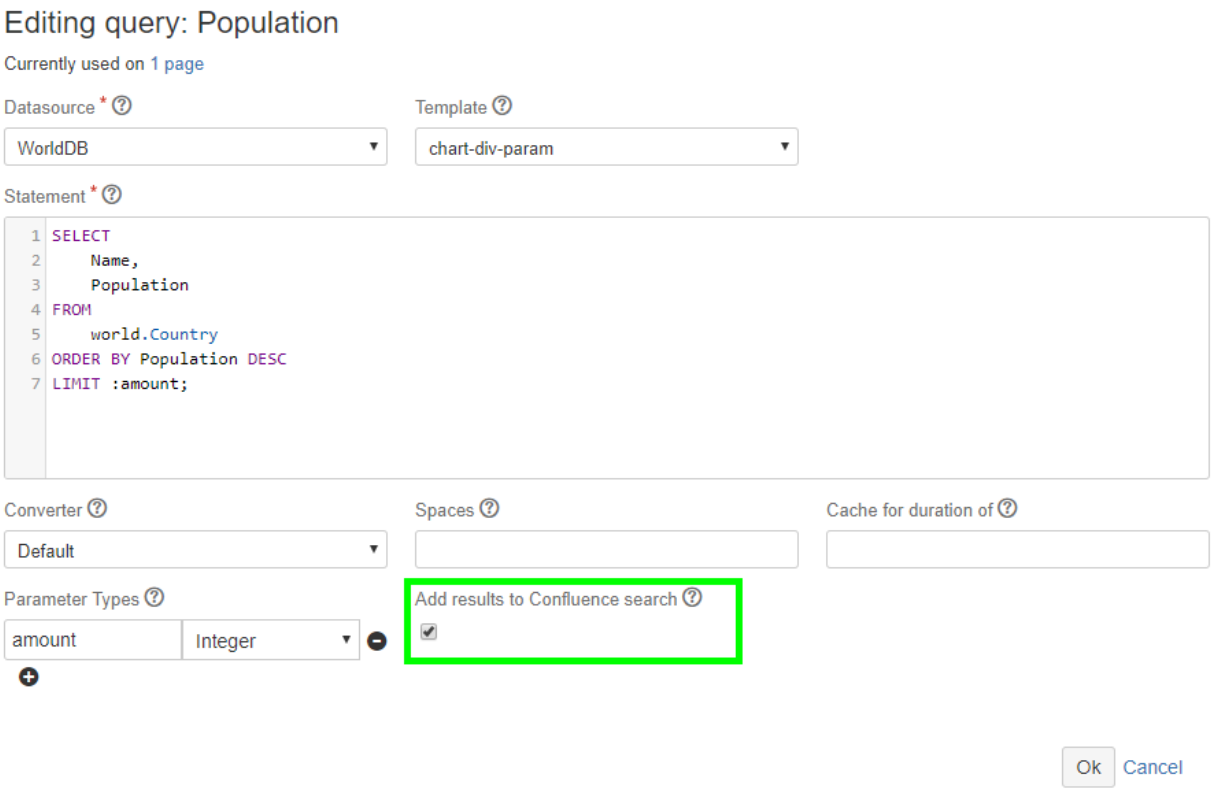
- **@username** - the username of the current Confluence user
- **@userfullname** - the full name of the current Confluence user
- **@usermail** - the email address of the current Confluence user
- **@page** - the title of the current Confluence page/blogpost
- **@pageid** - the ID of the current Confluence page/blogpost
- **@spacekey** - the space key of the current Confluence page/blogpost

For users that are not logged into Confluence **@username**, **@userfullname** and **@usermail** will return "anonymous".

> Note: When using a wildcard within the query statement or REST URL, it has to be prefixed with a colon to mark it as a parameter. So the resulting syntax is something like "SELECT * FROM mytable WHERE id LIKE :@pageid".

Adding Query Results to the Search Index

For each query you can decide whether its results should be added to the Confluence search index. They can then be found using the search bar on the top right, which makes content discovery much easier.



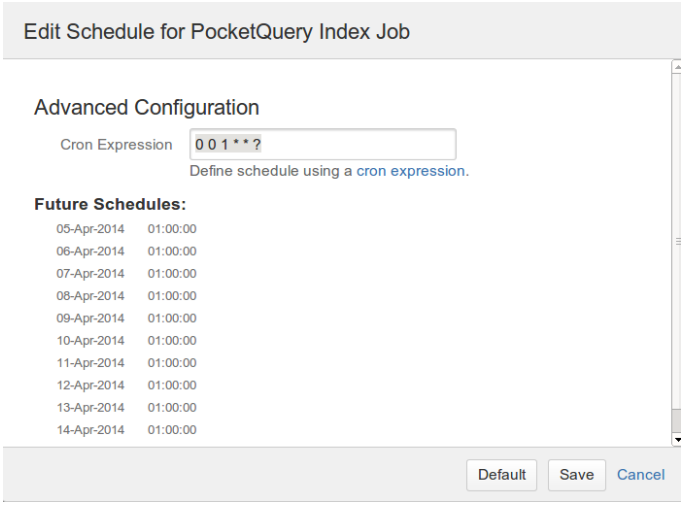
If the Index checkbox is checked, all results retrieved with this query will be added to the Confluence index when

- …a page (or other ContentEntityObject) with a PocketQuery macro using this query is edited or
- …the PocketQuery Index Job runs (see below)

PocketQuery Index Job

One challenge using the Confluence search index for PocketQuery results is that changes are not detected by Confluence like they are with the content of Confluence pages: Confluence won't recognize when something changes in your external database and the results of your macro change in consequence. In order to keep index data up-to-date, we implemented a Scheduled Job. Its task is to reindex all pages that contain a PocketQuery macro with a query that has indexing enabled. You can configure the job's schedule at *Confluence Admin > Scheduled Jobs*:

Flush Index Queue	Scheduled	04-Apr-2014 11:16:00	04-Apr-2014 11:16:05	180	History · Run · Edit
Flush Local Task Queue	Scheduled	04-Apr-2014 11:16:00	04-Apr-2014 11:17:00	2	History
Flush Mail Queue	Scheduled	04-Apr-2014 11:16:00	04-Apr-2014 11:17:00	18	History · Run · Edit · Disable
Flush Task Queue	Scheduled	04-Apr-2014 11:16:00	04-Apr-2014 11:17:00	1	History · Run · Disable
PocketQuery Index Job	Scheduled		05-Apr-2014 01:00:00	0	Run · Edit · Disable
scheduledjob.desc.purgeHistoryJob	Scheduled		05-Apr-2014	0	Edit



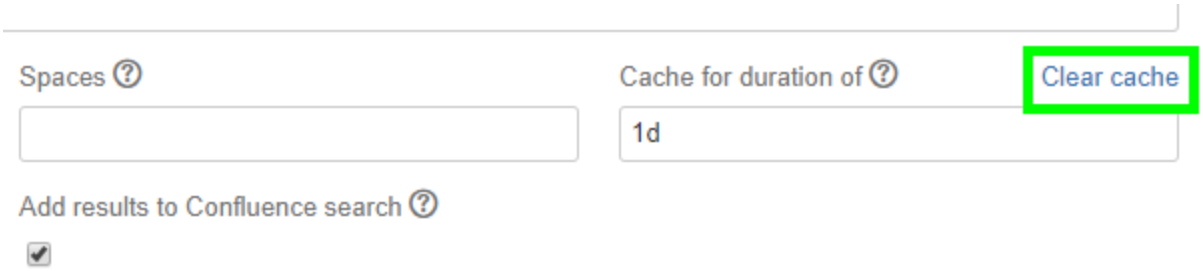
The default schedule is to trigger the job every night at 1 AM. You might want to change this regarding your specific environment. Note that the job might be quite performance-intensive since it may iterate over a large set of pages, depending on how many PocketQuery macros you use. Also, these queries are run against your external database to retrieve the latest results. (Feel free to disable the job if you don't want to make use of this feature at all.)

Caching Query Results

PocketQuery has a caching mechanism implemented that enables to cache the result of queries. This can reduce traffic with your datasource significantly. Without caching, every PocketQuery macro will trigger an interaction with the datasource when it is run (i.e. when a Confluence page with a macro is viewed).

For every query in the PocketQuery Admin, you can define how long the results of the query should be cached. When a PocketQuery macro with that query is run, it is checked if the result in the cache is older than the specified time, and only if so, the contents are refreshed by a new query. The cache duration can be specified in seconds (e.g. "45s"), minutes (e.g. "45m"), hours (e.g. "12h") or days (e.g. "3d").

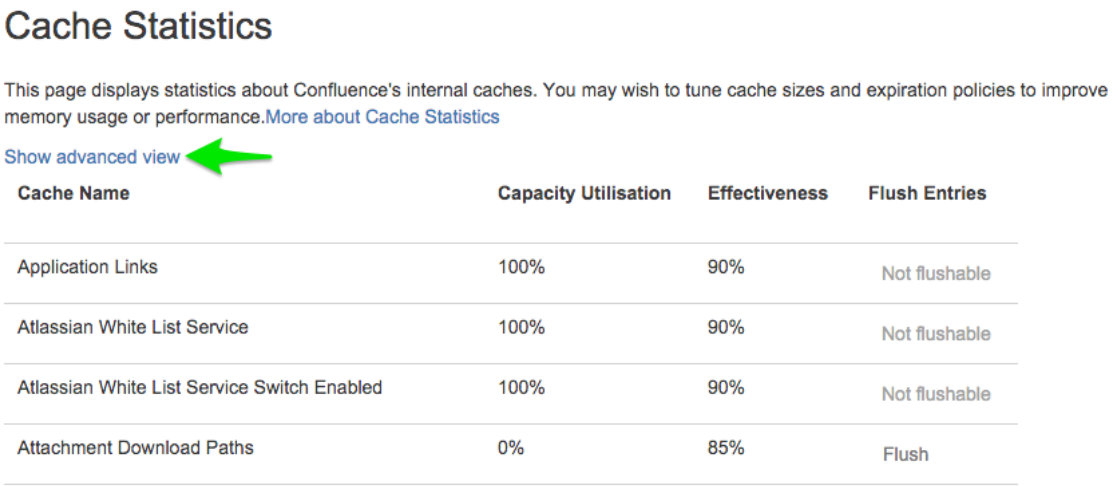
You can clear the cache separately for every query by clicking at the "Clear cache" label above the cache setting (within the PocketQuery Administration when editing a query):



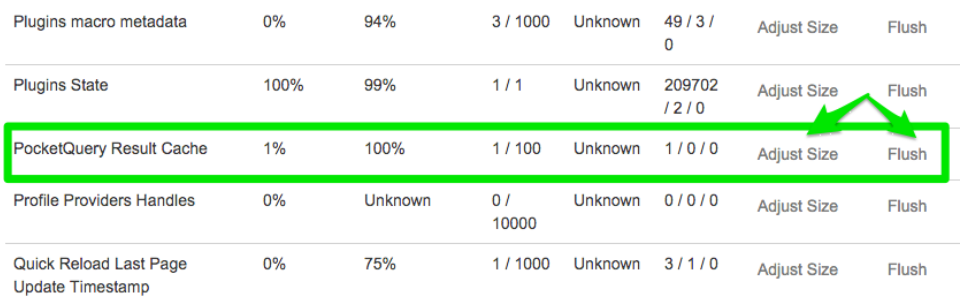
Cache Administration

Since results from the database can become arbitrarily big, it is important to limit the cache size to some amount. We set a default to 100 items which means that only a total of 100 query results can live in the cache. You can administer that limit at *Confluence Admin > Cache Management*.

Click on “Show advanced view” on top of the list:



Locate the PocketQuery cache row and click on “Adjust size”. From time to time it might also be useful to flush the whole PocketQuery cache and start from scratch again. You can do so by clicking “Flush”.



How it works

Results in the cache are always identified by three parts:

- query name
- query parameters
- query statement

More formally, each result gets a cache key by the following pattern:

```
queryName:::MD5(queryParameterString):::MD5(queryStatement)
```

This means, a new cache item is created when the PocketQuery macro is run if:

- there is no cached item yet for the current triple (queryName, queryParameters, queryStatement)
- the query parameters change
- the statement changes