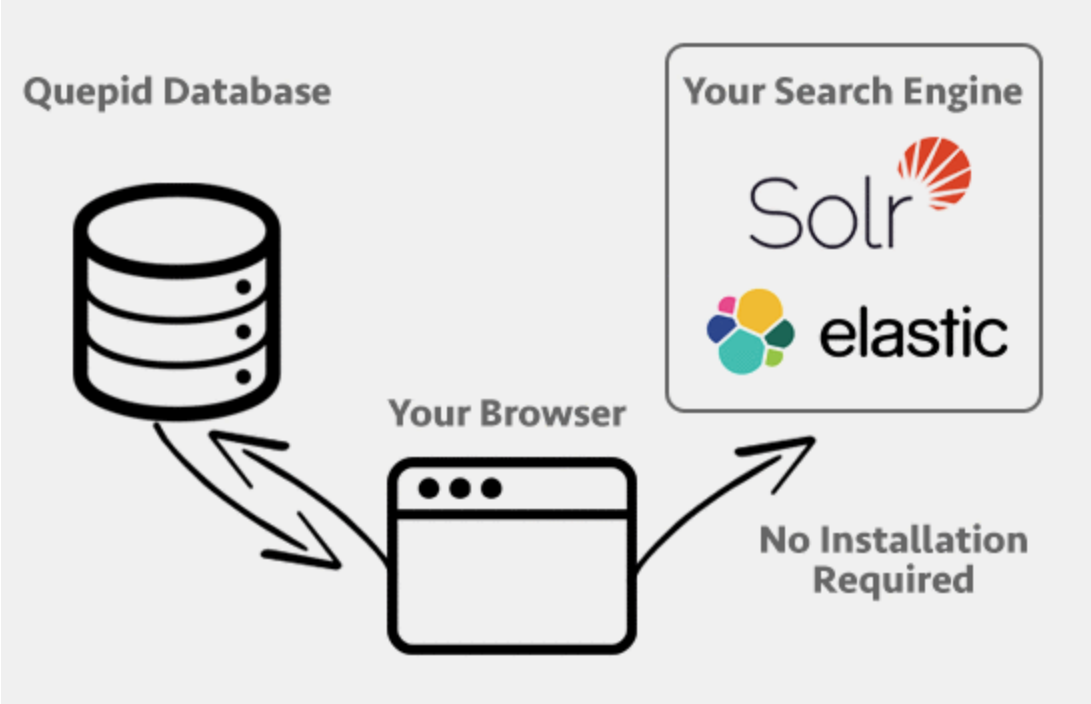
Quepid Data Storage Briefing

This document details the data that Quepid stores from your Solr or Elasticsearch instance. It's intended to describe in fine detail what is stored to allow customers with sensitive data to make informed decisions.

Quepid Data Storage Architecture

Quepid interacts with your search engine via a Javascript application running in your web browser. This allows Quepid to interact with your Solr or Elasticsearch even if it's only accessible in your local network (i.e behind a firewall). The following figure depicts how Quepid interacts with its database and your search engine



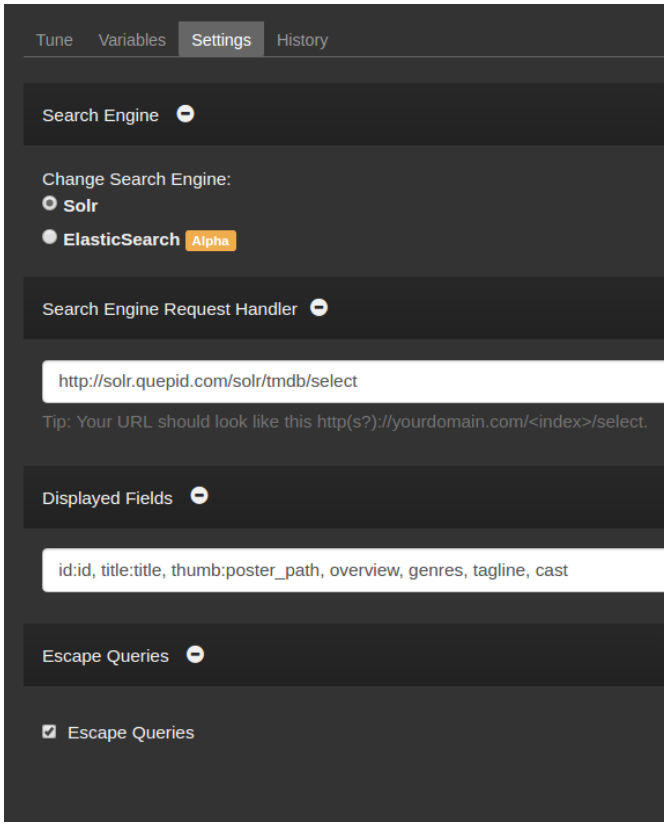
What data does Quepid Store?

When interacting with Solr or Elasticsearch Quepid endeavours to only store data necessary for relevance tuning and debugging. The following sections break down what is stored in different categories.

Search Engine Settings

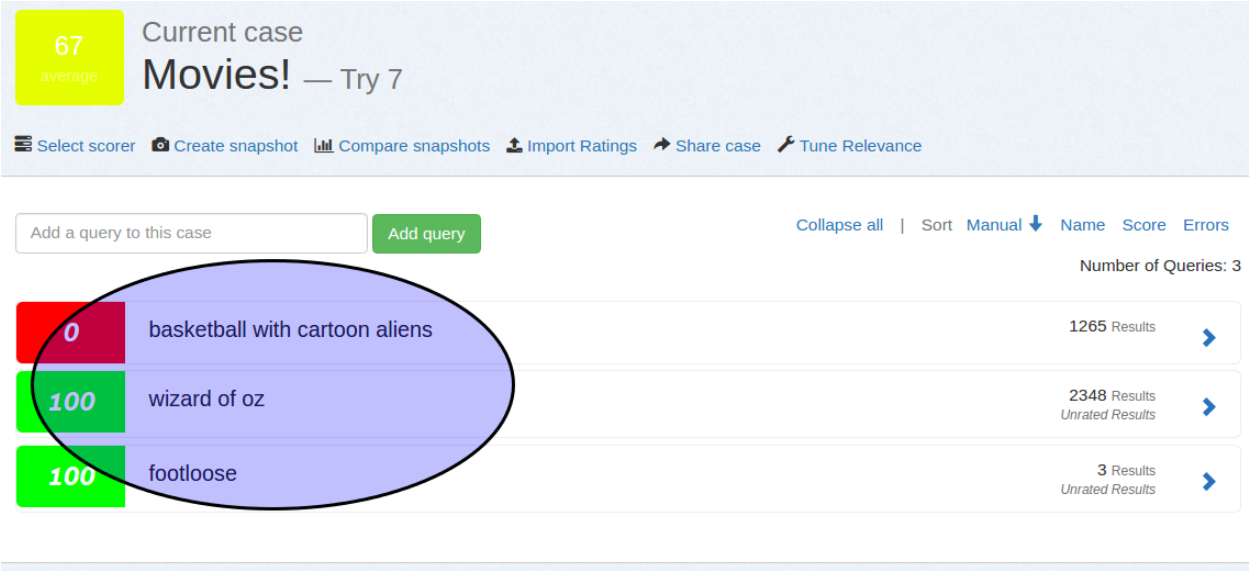
We store the URL to your Solr or Elasticsearch, the fields you'd like to display, and whether or not you'd like to apply Solr query escaping to the user's queries.

This data can be inspected by clicking **Tune Relevance** and selecting **Settings**. An example is shown in the screenshot below.



Search Keywords

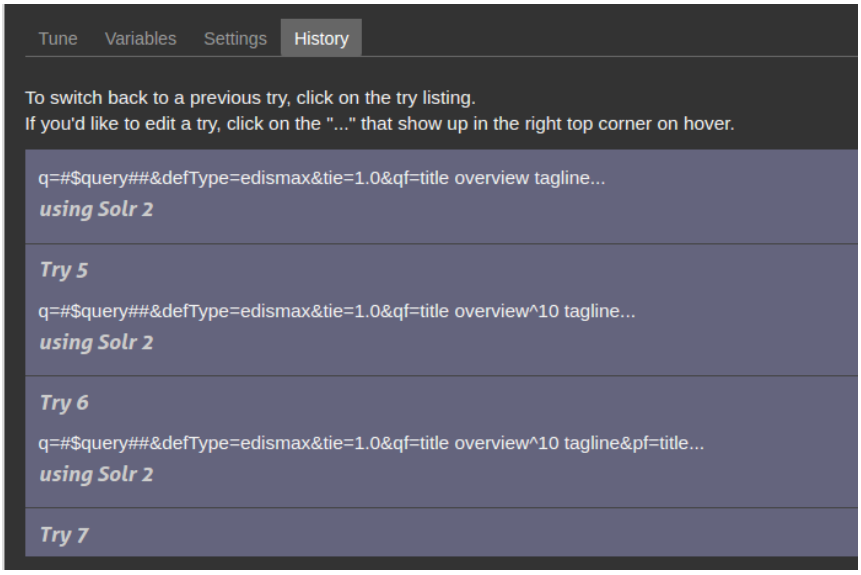
Quepid stores a collections of queries to test as a "case." Search developers or content managers add queries to be benchmarked in Quepid's user interface. To see this data, simply view the main screen after logging into Quepid. Here's a sample of user queries for our movie search engine:



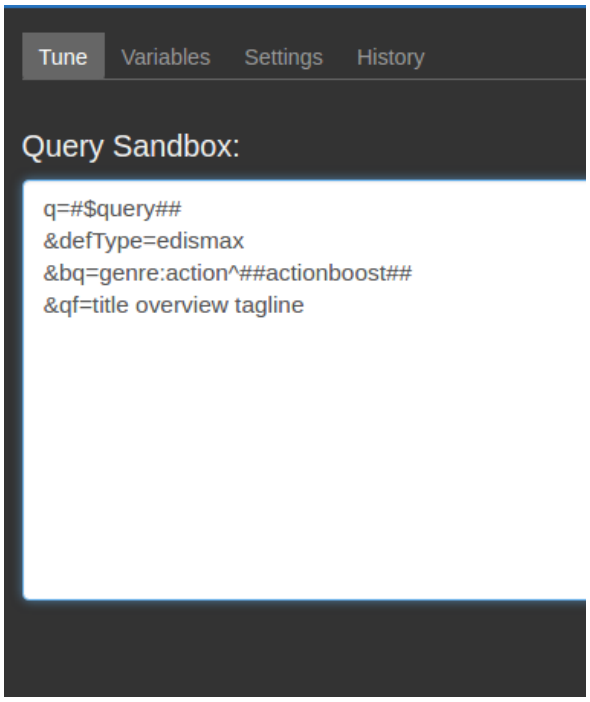
"Tries" -- History of relevance tuning parameters

For the convenience of Quepid users, we store a history of Solr and Elasticsearch query tuning parameters. This includes various filters, boosts, field names, and whatever additional Solr/Elasticsearch query language to modify how results are ranked.

Below is a sample of relevance tuning experimentation for our movies collection.
This data can be inspected by clicking **Tune Relevance** and selecting **History**



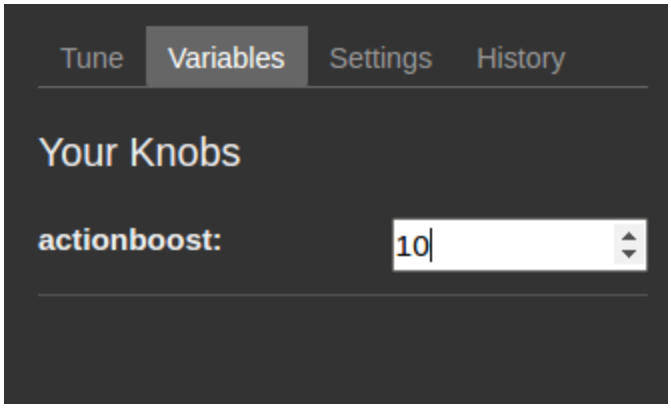
To drill down further, you can select a try and inspect what query parameters might be stored. If you select the "**Tune**" tab, you see the in depth details of what we're storing.



"Variables" -- various boosts and tunable parameters

In the "Query sandbox" above, you'll notice the **##actionboost##** parameter. This is Quepid's way of setting a parameter that a user can modify via the variables tab. In addition to the settings shown above, we also store the values of these tunable boosts in our database.

These values can be view in **Tune Relevance** -> **Variables**



Snapshots

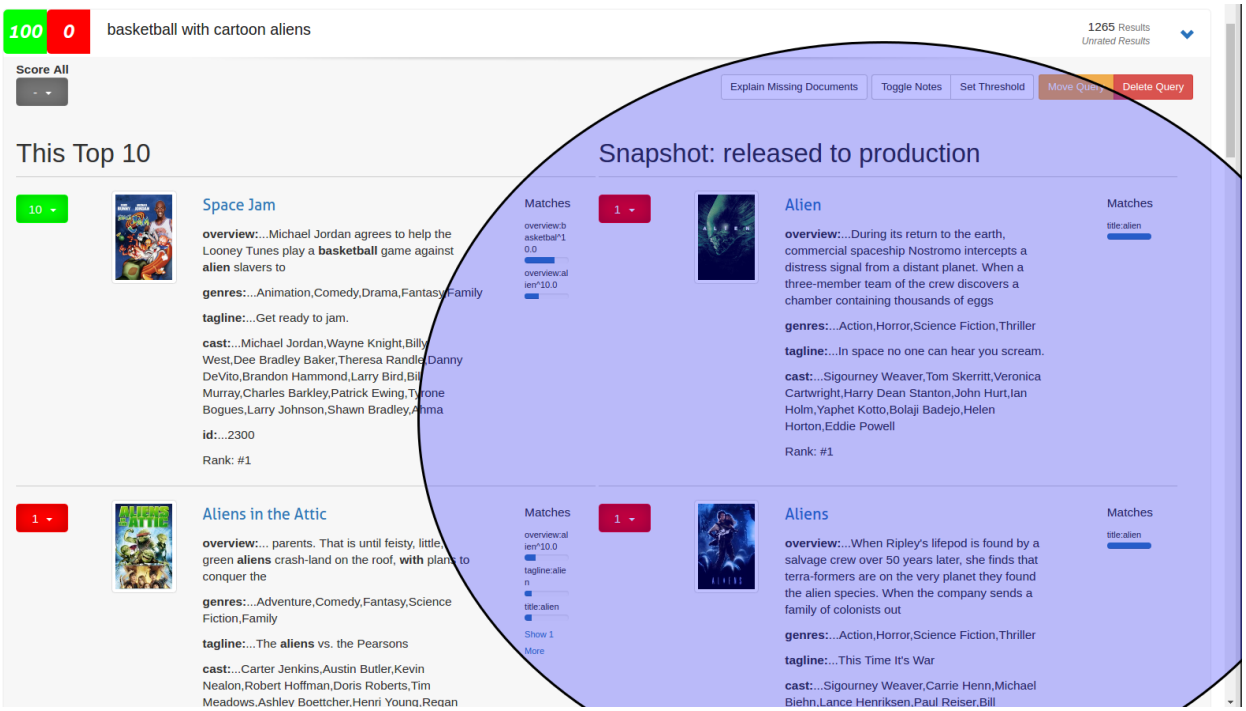
Snapshots let you "freeze" a set of search results along with their debug information. When taking a snapshot, we endeavour to minimize the data stored. We store the following

- A snapshot name (ie "released to production")
- A snapshot timestamp ("8/5/2016, 16:15")
- For each current search keyword, an ordered list of **ids** for each document (["1", "125"...])
- For each document, the **debug info** detailing the relevance of this document at this point in time (information on what exactly matched, such as *title:rambo*)

It's important to note we intentionally avoid storing data other than what's necessary. When Quepid displays the snapshot, it reconstructs the document fields in the display by querying the search engine at that point in time, using the stored id. If the document has drastically changed from the time the snapshot was created till when the comparison is made, then that can be reflected at that point. The team has spent a great deal of time ensuring this data stays only in the browser.

To take a snapshot, on the main Quepid screen select **Create snapshot**. Once a snapshot has been taken select **Compare Snapshots** to view what has been stored. Select the snapshot from the dropdown and select "Update diff settings."

Now within each query two sets of search results will be visible. On the left hand side is the current working set of search results. On the right side is the snapshot.



The debug information stored corresponds to the debug information returned from Solr's **debug.explain.structured** JSON, as documented here https://wiki.apache.org/solr/CommonQueryParameters#debug.explain.structured

For Elasticsearch the equivalent is the debug explain when **"explain": true.** Is added to a search query.
https://www.elastic.co/guide/en/elasticsearch/guide/current/relevance-intro.html#explain

This debug information lists

- What fields matched the search keywords (in the screenshot above, what matched keywords basketball, with, cartoon, and aliens
- What are the statistics behind relevance scoring? How often these words occur in the document and how often they occur over the whole corpus
- How are the relevance statistics combined into an overall score. Are the constituent term statistics summed together, multiplied, etc.

A sample of the debug info from the snapshot above can be viewed here

- https://gist.github.com/softwaredoug/578587f6c742386aa18c5d22da53bf3b

If any of this data is sensitive, we recommend not using the snapshot feature.

How is Quepid Hosted?

The Quepid app is built and deployed using Docker. The Docker image is deployed and hosted on the Google Computer Platform (aka Google Cloud) Kubernetes platform, hosted in a data center in the United States. The whole process is managed using an internal tool that we developed called Keel. The Rails app uses a MySQL database that is also hosted on Google Computer Platform via the Cloud SQL platform.