On the use of SPARQL in my Project

After doing some additional research, I do not think SPARQL will be a suitable query language for this project. I was unable to find evidence of SPARQL support for either the MusicBrainz webservice or the MediaWiki API that IMSLP uses. Some MediaWiki installations include an extension that would allow SPARQL queries but IMSLP in particular does not. I do not plan on storing either database in its entirety locally, so it would not be useful that way either.

The question was not at all in vain because it led me to do more research on the MediaWiki API and the way MediaWiki organizes information. My understanding so far is that the data is stored in *pages* which can have *properties* and belong to *categories*. This sounds like a structure that would be easily translated to a graph. Furthermore, if I am to match search results from MusicBrainz and IMSLP, I would need to traverse this “in category” relationship to determine how to match a given page on IMSLP with an entity in the MusicBrainz database (e.g. is this page for a composer or a work?) The relationships between entities in the MusicBrainz database could also easily be converted to a graph (much like any other SQL database).

Thus, it seems that a graph database could provide a good intermediate representation of the data from both sources. It could also be useful for browsing results and allow for graph queries on the results (e.g. all composers two “category-hops” away from Alkan). With this in mind, my site would work like this:

* An initial search page allows a user to enter a free-form search query (like Google)
* The backend queries both databases and displays the results. These would be links and include either the composer’s name or title of a work
* Once clicked, the data about the composer or work would be fetched and loaded into a graph database local to my site and perhaps unique for each session (or not, there are advantages both ways I think)
* This data would be used to generate a page with links to entries on either site and either a list of works (if composer) or a list of scores (if work), plus any information from the MusicBrainz database about the composer/work. I could perhaps include links to adjacent nodes too
* This could also be used to generate recommendations based on category membership. If I store search queries (and perhaps make them part of the graph too somehow), I can also offer recommendations based on what other users did
* The graph visualization features of OrientDB could allow for a visual representation of the graph
* Only the parts of the graph immediately visible would need to be loaded, but a given graph could be stored and exported for caching or to allow users to save them for later

Since MediaWiki is a popular wiki engine, the part of my project that loads MediaWiki data into a graph could be used in other projects.