Query your content

No SQL queries

Apstrata relieves you from having to create a relational database, defining tables or manipulating DDL or SQL. You are provided with a ready to use **st ore**, within which you create **documents.** Documents have a **key/value pairs** structure, where the key is a field name and the value can have one of the following types: string, numeric, text, date, file or geospatial.
To retrieve the data that you have persisted in your documents, you can either:

- Retrieve the whole document using the **document key** (unique identifier of a document), by invoking Query API
- Define sophisticated queries to search for document matching your criteria, using the Query API as well

Back to the map **Next station**: Apply validation

Example 1: load my saved game

So you are implementing a cool game that runs on mobile devices, such as smartphones or tablets. Since game data has to be accessible from any device owned by the user, you decided to persist it as an Apstrata document. What you need now is to load the game data back to the device, i.e. get the corresponding document using its document key (we assume our document has the following key 1EB1FFB956C9D0D7155C7422749F485A).

```
 var params = {

        "apdsb.store": "DefaultStore", // "DefaultStore" is the name by default. If you did not rename
your store, this line is optional
        "apsdb.query": "apsdb.documentKey=\"1EB1FFB956C9D0D7155C7422749F485A\"", // Notice that we use
the "apsdb.query" field to define our query
        "apsdb.fields": "player, score, level, lives" // We use "apsdb.fields" to specify the document
fields to return
}

// We invoke the "Query" API through the native "apsdb" object.
// Since we are not uploading any file, the third parameter is null
return apsdb.callApi("Query", params, null);
```

Try it!

Example 2: get top 10 highest scores, in descending order

Imagine you need to add a "hall of fame" section in your application, where the player can see the 10 best scores, the corresponding level and the name of the player. All you have to do is to create a query which returns no more than 10 results and such as "score > 0". You also should specify the sort order (descending). Let us see how easy this is to implement with the Apstrata Query API:

```
curl "https://varick.apstrata.com/apsdb/rest/O763A7F690/Query?apsws.time=1417704302580&apsws.
authSig=09e9c225c42a6893fed9da9f5168f585
&apsws.responseType=json&apsws.authMode=simple&apsdb.store=cabstore
&apsdb.query=score"%"3Cnumeric"%"3E"%"20"%"3E"%"200
&apsdb.queryFields=score"%"2C"%"20level"%"2C"%"20player
&apsdb.sort=score"%"3Cnumeric"%"3ADESC"%"3E
&apsdb.resultsPerPage=10"
```

Try it!

Example 3: get highest scores grouped by level

Now let us say that you would like to know what are the highest scores per game level. What you need is to group the results of your query by the "level" field. This is easily done using the "apsdb.aggregateGroupBy" parameter of the Query API.

```
curl "https://varick.apstrata.com/apsdb/rest/O763A7F690/Query?apsws.time=1417707644199&apsws.
authSig=bfe7bf3814e8e6908e9369967bcf48d0
&apsws.responseType=json&apsws.authMode=simple
&apsdb.store=DefautStore
&apsdb.query=score"%"3Cnumeric"%"3E"%"20"%"3E"%"200
&apsdb.queryFields=score"%"2C"%"20level"%"2C"%"20player
&apsdb.sort=score"%"3Cnumeric"%"3ADESC"%"3E
&apsdb.aggregateExpression=Max("%"24score)
&apsdb.aggregateGlobal=true
&apsdb.aggregateGroupBy=level"%"3Cnumeric"%"3E"
```

Try it!

Dig deeper

- Dynamic Queries API
- Query

Related tutorials

Apply validation	Create Query components	Persist your content	Query your content