

Overview

Setup

- Quick start
- Data Fields
- Lookup relations
- Parent-Child relations
- Connect to your online database
- Database limits

Coding

- Database REST API

REST API examples and queries

- Query language
- Simple queries
- Logic operators
- Conditional operators
- Date operator
- Dynamic date variables
- Sort operator
- URL Parameters overview
- Query examples
- SQL to NoSQL mapping

- Data manipulation with REST
- Swagger
- Code examples for REST API
- Database Driven Websites with Pages
- Plug-in API
- JavaScript API
- Realtime Messaging with EventSource
- NoSQL Database Aggregation
- Mail API
- Custom REST routes
- Codehooks
- Node.js modules
- Webhooks
- Rest API Inspector
- Command line tools
- Examples

Utils

- Media Archive
- Web Form Generator

REST API examples and queries

Querying your database is an essential part of any application. restdb.io uses plain URLs with simple parameters and JSON documents to query your database. In this section we will give you plenty of REST API examples using MongoDB like query syntax.

🚀 If you love the ease of making APIs and mongodb-like queries of restdb.io, please also check out our new backend service: codehooks.io. It integrates Javascript serverless functions with a document database (with MongoDB-like queries of course), a key/value database, a message queue and a CRON-like job system. A powerful CLI lets you instantly deploy new code.

Query language

Database queries are created as valid JSON documents. A query object consists of fields and operators that make up a complete query. *Please note that query fields are case sensitive.*

A simple query example:

```
https://mydb-fafc.restdb.io/rest/customers?q={"status": "GOLD"}
```

Which is the same as asking the database: *fetch all records with status GOLD*

Please note that maximum 1000 records are returns from any query unless the \$max parameter is specified.

Simple queries

Match a single field (find Joe):

```
https://mydb-fafc.restdb.io/rest/people?q={"name": "Joe"}
```

Match multiple fields (find Joe who's 17):

```
https://mydb-fafc.restdb.io/rest/people?q={"name": "Joe", "age": 17}
```

In addition to query simple fields, you can use special operators to perform more advanced queries.

Logic operators

\$not operator

Return documents not matching the query.

```
https://mydb-fafc.restdb.io/rest/people?q={"name" : {"$not" : "Joe"}}
```

\$in operator

Return documents matching any values.

```
https://mydb-fafc.restdb.io/rest/people?q={"name" : {"$in" : ["Joe", "Jane", "Donald"]}}
```

\$nin operator

Return documents not matching any of the values.

```
https://mydb-fafc.restdb.io/rest/people?q={"name" : {"$nin" : ["Joe", "Jane", "Donald"]}}
```

\$regex operator

Match a regular expression against field.

Random Data Generator

Controlling API output with intercepts

Resource Cache

Security

API keys and CORS Ajax calls

Authentication of external users with Auth0.com

Users and Roles

Admin

Import and Export of Excel, CSV, JSON

Snapps - clone your database

External MongoDB

Using custom domains with restdb.io and Let's Encrypt

```
https://mydb-fafc.restdb.io/rest/people?q={"name" : {"$regex" : ".Joe*"}}
```

\$or operator

Return documents that matches one or the other field.

```
https://mydb-fafc.restdb.io/rest/people?q={"$or": [{"name": "Jane"}, {"name": "Donald"}]}
```

\$and operator

Return documents both fields.

```
https://mydb-fafc.restdb.io/rest/people?q={"$and": [{"name": "Jane"}, {"last-name": "Cassid"}]}
```

Logical operators overview

Operator	Description	Example
\$not	Negation logical operator	<code>{"field" : {"\$not" : val}}</code>
\$in	Match any value in array	<code>{"field" : {"\$in" : [value1, value2, ...]}}</code>
\$nin	Not match any value in array	<code>{"field" : {"\$nin" : [value1, value2, ...]}}</code>
\$regex	Match field	<code>{"field" : {"\$regex" : ".*"}}</code>
\$or	Logical operator	<code>{"\$or": [{"status": "GOLD"}, {"status": "SILVER"}]}</code>
\$and	Logical operator	<code>{"\$and": [{"status": "GOLD"}, {"sales": 1000}]}</code>

Conditional operators

\$gt operator

Return documents that matches each field value greater than numeric value.

```
https://mydb-fafc.restdb.io/rest/people?q={"salary": {"$gt": 10000}}
```

\$gte operator

Return documents that matches each field value greater than or equal to numeric value.

```
https://mydb-fafc.restdb.io/rest/people?q={"salary": {"$gte": 10000}}
```

\$lt operator

Return documents that matches each field value less than numeric value.

```
https://mydb-fafc.restdb.io/rest/people?q={"salary": {"$lt": 10000}}
```

\$lte operator

Return documents that matches each field value less than or equal to numeric value.

```
https://mydb-fafc.restdb.io/rest/people?q={"salary": {"$lte": 10000}}
```

\$bt operator

Return documents that matches each field value between two numeric values.

```
https://mydb-fafc.restdb.io/rest/people?q={"salary": {"$bt": [5000, 7500]}}
```

\$exists operator

Return documents that matches each field with a value.

```
https://mydb-fafc.restdb.io/rest/people?q={"field": {"$exists": true}}
```

\$exists (sub array) operator

Return documents that matches each sub field with any value.

```
https://mydb-fafc.restdb.io/rest/people?q={"field.0": {"$exists": true}}
```

\$elemMatch operator

Return documents that matches at least one of the elements in an array field.

```
https://mydb-fafc.restdb.io/rest/people?q={"contact":{"$elemMatch":{"name":"Anderson", age:
```

\$distinct operator

Return array of unique values from matching fields.

```
https://mydb-fafc.restdb.io/rest/people?q={"$distinct": "age"}&sort=age
```

Example output:

```
[
  "18",
  "19",
  "20",
  "21",
  "22",
  "23",
  "24",
  "25",
  "26"
]
```

Conditional operators overview

Operator	Description	Example
\$gt	>	<code>{"salary": {"\$gt": 10000}}</code>
\$gte	>=	<code>{"salary": {"\$gte": 10000}}</code>
\$lt	<	<code>{"salary": {"\$lt": 10000}}</code>
\$lte	<=	<code>{"salary": {"\$lte": 10000}}</code>
\$bt	>= value <=	<code>{"salary": {"\$bt": [5000, 7500]}}</code>
\$exists	Check if field exists	<code>{"field": {"\$exists": true false}}</code>
\$exists (array)	Check if array field exists or is empty	<code>{"field.0": {"\$exists": true false}}</code>
\$elemMatch	Array element matching	<code>{"contact":{"\$elemMatch":{"name":"Anderson", age:35}}}</code>
\$distinct	Array with unique element values	<code>{"\$distinct": "name"}</code>

Date operator

Querying based on dates are done using the `$date` operator combined with ISO date strings. For example:

```
// between two dates
{"_changed":{"$gt":{"$date":"2016-08-01"},"$lt":{"$date":"2016-08-05"}}
```

Dynamic date variables

To enable more dynamic queries using the `$date` operator, like "get all records this month", we have added a number of predefined date variables.

```
// older than the time just now
{"startat":{"$lt":{"$date":"$now"}}
```

Name	Description
\$now	just now
\$currentMinute	start of last minute
\$currentHour	start of last hour
\$currentDate	start of current day
\$today	start of current day
\$tomorrow	start of next day
\$yesterday	start of day before
\$currentWeek	start of this week (sunday)
\$currentMonth	start of this month
\$currentYear	start of this year

\$nextWeek	start of next week (sunday)
\$nextMonth	start of next month
\$nextYear	start of next year

We also support [MomentJS](#). Check out their docs and compose the date variable like this:

```
{
  "startat": {"$gt": {"$date": "$moment.endOf.hour"}}
},
{
  "startat": {"$gt": {"$date": "$moment.subtract.3.days"}}
},
{
  "startat": {"$gt": {"$date": "$moment.subtract.3.hours"}},
  "$lt": {"$date": "$moment.add.3.hours"}
},
{
  "startat": {"$gt": {"$date": "$moment.startOf.year"}},
  "$lt": {"$date": "$moment.endOf.year"}
}
```

Sort operator

The `$orderby` operator lets you sort the result from a query. You can sort on single or multiple fields within the same query. We use the URL parameter hint `h={...}` to specify sorting.

E.g. query all records from `people` and sort the result set by `name` in ascending order, and then by `age` in descending order.

```
https://mydb-fafc.restdb.io/rest/people?q={}&h={"$orderby": {"name": 1, "age": -1}}
```

URL Parameters overview

As part of the REST API, restdb.io has a number of parameters that we can use. They are added to the REST URLs as query parameters shown below:

```
https://<db-name>.restdb.io/rest/<collection>[.<format>]?q={}&h={}&...
```

The following table shows all valid URL parameters to a database REST endpoint:

Parameter	Description
q	Database query Example: GET "Karen Johnston, age 39": <code>https://mydb-fafc.restdb.io/rest/people?q={"firstname":"Karen", "lastname":"Johnson", "age": 39}</code>
h	Query hints to specify: fields, max, skip and orderby. Aggregation parameters can also be specified as hints, read more here Example: Return title, only 10 records, forward to the 5.th item and sort result set by the body field: <code>https://mydb-fafc.restdb.io/rest/people?q={}&h={"\$fields": {"title":1}, "\$max":10, "\$skip":5, "\$orderby":{"body":1}}</code>
filter	Performs a text search and retrieves all matching documents Example: <code>https://mydb-fafc.restdb.io/rest/people?q={}&filter=johnson</code>
sort	Specifies which field(s) should be used to sort the result. You can add multiple fields by simply adding another sort parameter. Default sort= <code>_id</code> Example: <code>https://mydb-fafc.restdb.io/rest/people?q={}&sort=lastname</code>
dir	Sort direction. Allowed values are 1 (ascending) and -1 (descending). Used together with sort. Multiple dir parameters can be used in conjunction with sort. Example: <code>https://mydb-fafc.restdb.io/rest/people?q={}&dir=-1</code>
skip	Where to start in the result set. Used for paging. Example: <code>https://mydb-fafc.restdb.io/rest/people?skip=100</code>
max	Maximum number of records retrieved. Default is 1000. Example: <code>https://mydb-fafc.restdb.io/rest/people?max=20</code>
totals	<code>&totals=true</code> returns an object with both data and total count. Totals equals to max parameter or default 1000 Example: output from query -> <code>{data: [...], totals: { total: 100, count: 40, skip: 0, max: 1000}}</code>
totals and count	<code>&totals=true&count=true</code> returns an object with no data and just the total count Example: output from query -> <code>{data: [], totals: { count: 42}}</code>
metafields	Displays internal restdb.io fields (<code>_keywords</code> , <code>_tags</code> , <code>_version</code> , <code>_created</code> , <code>_createdby</code> , <code>_mock</code>) Example: <code>https://mydb-fafc.restdb.io/rest/people?metafields=true</code>
groupby	Group output from query by a field Example: <code>https://mydb-fafc.restdb.io/rest/people?groupby=gender</code>
aggregate	Perform aggregation function on data groups. Useful together with groupby parameter. Availiable functions; SUM, AVG, MIN, MAX and COUNT Docs Example: <code>https://mydb-fafc.restdb.io/rest/people?groupby=gender&aggregate=SUM:weight&aggregate=AVG:age</code>

format	Output format from request. .html returns a standard markup for all fields. .js (javascript) returns a script that can be included in a web page for search engine indexing. .json standard JSON format
	Example: <code>https://mydb-fafc.restdb.io/rest/people.html</code>
apikey	A valid apikey, full access key or a CORS enabled key
	Example: <code>https://mydb-fafc.restdb.io/rest/people?apikey=4321fd234f0876...</code> Not recommended for production. Use header field <code>x-apikey</code> instead
idtolink	Inserts canonical URLs for image references and object references
	Example: <code>https://mydb-fafc.restdb.io/rest/people?idtolink=true</code>
flatten	Used together with idtolink. Extract links as properties on root object
	Example: <code>https://mydb-fafc.restdb.io/rest/people?flatten=true</code>
referencedby	Return all items that refers to a record. Requires a record_id in the query or path
	Example: <code>https://mydb-fafc.restdb.io/rest/people/56011150e1321c7300000001?referencedby=true</code>
fetchmediadata	Replace media ID's with a full record from the media archive
	Example: <code>https://mydb-fafc.restdb.io/rest/people?fetchmediadata=true</code>
fetchchildren	Insert records from parent-child relation on parent record
	Example: <code>https://mydb-fafc.restdb.io/rest/company?fetchchildren=true</code>

Query examples

The following table shows some examples on how to query restdb.io:

Query	Syntax
Get items that have status = "pro" and title="yay"	<code>q={"status":"pro","title":"yay"}</code>
Get items that have status = "pro" or "basic"	<code>q={"\$or":[{"status":"pro"}, {"status":"basic"}]}</code>
Get items that have status like "basic" (regular expression)	<code>q={"status":{"\$regex": "basic"}}</code>
Get all items updated within date interval (use ISOString or dates that can be converted to ISOString)	<code>q={"_changed":{"\$gt":{"\$date":"2016-08-01"},"\$lt":{"\$date":"2016-08-05"}}}</code>
Get items created this month, i.e. newer than 1.st 00:00 AM this month	<code>q={"_created":{"\$gt":{"\$date":"\$currentMonth"}}}</code>
Get all items with a score > 900, and group them by category	<code>h={"\$groupby":["category"]}&q={"score":{"\$gt": 900}}</code>
Get all metadata for a database	<code>https://customerdb-fe0f.restdb.io/rest/_meta</code>
Get an image thumbnail from the media archive	<code></code>

Comprehensive [code examples for various languages](#)

SQL to NoSQL mapping

The following table shows how traditional SQL statements are expressed as RestDB Queries.

SQL	RestDB
SELECT * FROM users	<code>https://<db-name>.restdb.io/rest/users?q={}</code>
SELECT id, user_id, status FROM users	<code>/rest/users?q={}&h={"\$fields": {"user_id": 1, "status": 1 } }</code>
SELECT * FROM users WHERE status = "A"	<code>/rest/users?q={ "status": "A" }</code>
SELECT * FROM users WHERE status != "A"	<code>/rest/users?q={"status":{"\$not":"A"}}</code>
SELECT * FROM users WHERE status = "A" AND age = 50	<code>/rest/users?q={ "status": "A", "age": 50 }</code>
SELECT * FROM users WHERE status = "A" OR age = 50	<code>/rest/users?q={ "\$or": [{ "status": "A" }, { "age": 50 }] }</code>
SELECT * FROM users WHERE age > 25	<code>/rest/users?q={ "age": { "\$gt": 25 } }</code>
SELECT * FROM users WHERE age < 25	<code>/rest/users?q={ "age": { "\$lt": 25 } }</code>
SELECT * FROM users WHERE age > 25 AND age <= 50	<code>/rest/users?q={ "age": { "\$gt": 25, "\$lte": 50 } }</code>
SELECT * FROM users WHERE user_id like "%bc%"	<code>/rest/users?q={ "user_id": {"\$regex": "bc"}}</code>
SELECT * FROM users WHERE user_id like "bc%"	<code>/rest/users?q={ "user_id": {"\$regex": "^bc"}}</code>
SELECT * FROM users WHERE status = "A" ORDER BY user_id ASC	<code>/rest/users?q={ "status": "A" }&sort=user_id&dir=1</code>
SELECT * FROM users WHERE status = "A" ORDER BY user_id DESC	<code>/rest/users?q={ "status": "A" }&sort=user_id&dir=-1</code>
SELECT COUNT(*) FROM users	<code>/rest/users?q={}&h={"\$aggregate":["COUNT"]}</code>
SELECT COUNT(*) FROM users WHERE age > 30	<code>/rest/users?q={"age":{"\$gt": 30}}&h={"\$aggregate":["COUNT"]}</code>
SELECT * FROM users LIMIT 1	<code>/rest/users?q={}&max=1</code>

SELECT * FROM users LIMIT 5 SKIP 10

/rest/users?q={}&max=5&skip=10

Site Links

Home Page
About Us
Contact
Blog
Templates Catalog

Cloud Service

Features
Pricing
Terms & Conditions
Privacy Policy
Sign Up »

Documentation

Overview
Getting Started
Coding against the API
Utils
Security and Admin



© 2023 restdb.io