Filters and sort benchmark

Stress tests various query filters (with optional sorting and counting).

Usage

```
NUM_NODES=1000 NUM_PAGES=1000 FILTER=eq SORT=1 TEXT=1 COUNT=1 yarn bench
```

Description

All queries have `limit=100` (although some of them may return just several items or 0).

Env vars:

- `NUM_NODES` : The number of nodes created (1000 by default)
- `NUM_PAGES` : The number of pages created (1000 by default, must be <= `NUM_NODES`)
- `FILTER` . Available values:
  - `eq` : captures 1/4 of all nodes (default)
  - `eq-id` : captures a single node by id
  - `eq-uniq` : captures a single node by unique value (e.g. `slug`)
  - `eq-two-fields` : applies `eq` filter to two fields and captures 1/4 of all nodes
  - `elemMatch-eq` : captures 1/2 of all nodes
  - `in` : captures 1/2 of all nodes
  - `gt` : the first query captures all nodes, the last one - 0 nodes
  - `lt` : the first query captures 0 nodes, the last - all nodes
  - `gt-lt` : any query captures 1000 items; last 1000 pages will capture from 999 to 0 (`gt: currentPage, lt: currentPage + 1000`)
  - `nin` : captures 1/2 of all nodes
  - `ne` : captures 3/4 of all nodes
  - `regex` : captures from 1/4 to 1/3 of all nodes (simple and fast regexp)
- `SORT` . Available values:
  - `0` : no sort (default)
  - `1` : sorts by random number
  - comma-separate list of fields (e.g. `SORT=fooBar,random` sorts by fields `[foo, bar]`)
- `TEXT` . Available values:
  - `0` : small nodes without big text content (default)
  - `1` : adds 4kb of random text to each node. Note: text is returned by graphql queries, so it affects `page-data.json` file size.
- `COUNT` . Available values:
  - `0` : query doesn't request total count of items (default)
  - `1` : adds `totalCount` to query request

Example

Let's figure out time complexity of `gt` filter. To make this happen - let's run the benchmark 3 times with the same number of pages but growing number of nodes:

run 1:

```
NUM_NODES=1000 FILTER=gt yarn bench
```

run 2:

```
NUM_NODES=10000 FILTER=gt yarn bench
```

run 3:

```
NUM_NODES=100000 FILTER=gt yarn bench
```