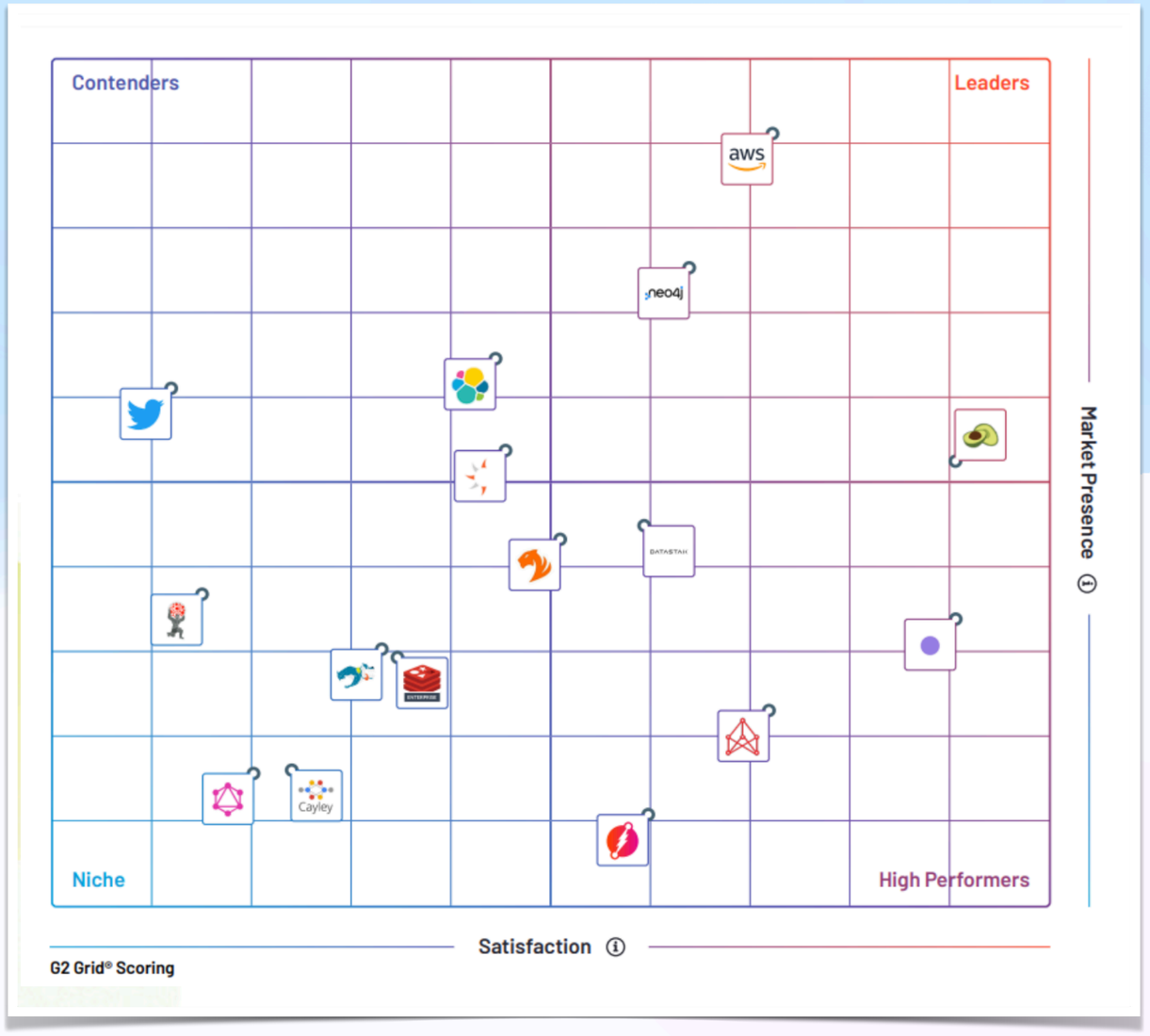
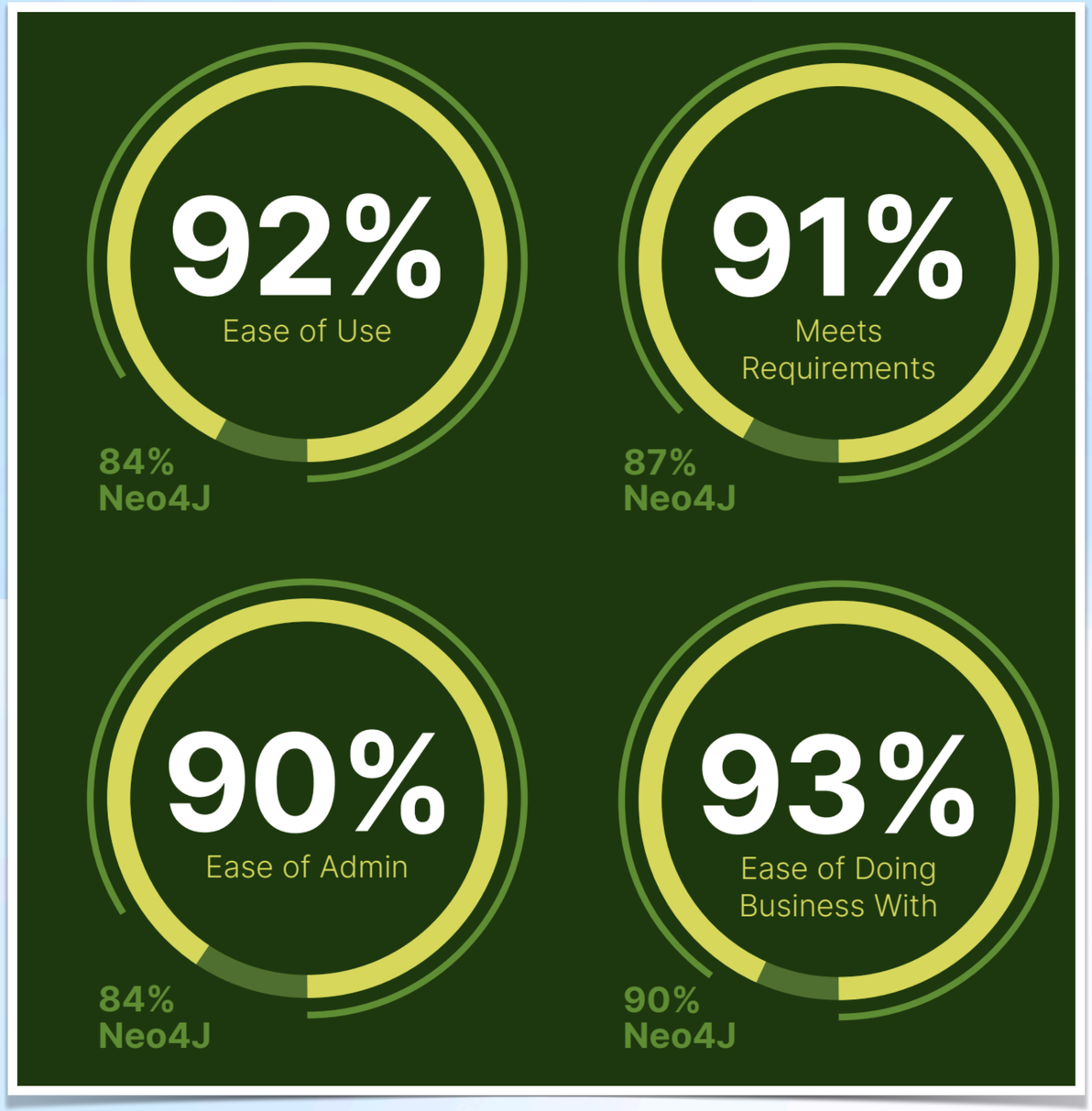
ArangoDB

final milestone strugglings and victories

Mikołaj Wielgos, 24 Jan 2025

Rated the #1 Graph Database in the world for Winter 2025





"Beautiful query language, **best performance**"

G2 Reviewer

"**Great choice** for multi-model connected data"

G2 Reviewer

"ArangoDB is the **modern solution to the modern problem!**"

G2 Reviewer

"Great DB for **real-world data**"

G2 Reviewer

★★★★★

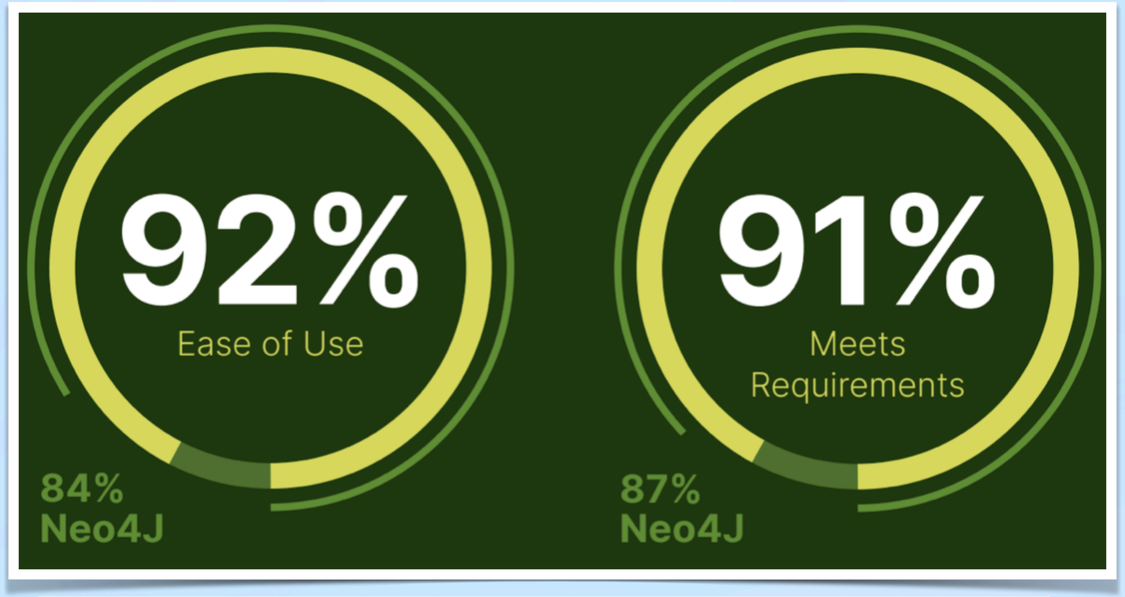
Verified User in Hospital & Health Care ⓘ

Enterprise (> 1000 emp.)

Validated Reviewer ✔ | Review source: Seller invite | Incentivized Review

★★★★★ Oct 27, 2023

"Great DB for real world data"



Meets Requirements	8.8 (99)	8.5 (72)
Ease of Use	8.9 (100)	8.2 (74)
Ease of Setup	8.7 (67)	8.8 (30)
Ease of Admin	8.7 (54)	8.1 (27)
Quality of Support	8.9 (87)	8.5 (63)

License

From **Apache 2.0** license to **BSL 1.1** to **Apache 2.0** again after 4 years

Current license allows full usage of the ArangoDB source code for any purpose except for providing a managed service of ArangoDB.

ArangoDB Community Edition - commercial use up to 100GB limit on dataset size within a single cluster

Implementation

Docker

```yaml
services:
    arangodb:
        image: arangodb
        container_name: arangodb
        environment:
            - ARANGO_ROOT_PASSWORD=test123
        ports:
            - "8529:8529"
        volumes:
            - arangodb_data:/var/lib/arangodb3
            - ./data:/data

volumes:
    arangodb_data:
        driver: local
```

Entering database

```
/ # arangosh --server.username root --server.password test123


                                       _
  __ _ _ __ __ _ _ __   __ _  ___  ___| |__
 / _` | '__/ _` | '_ \ / _` |/ _ \/ __| '_ \
| (_| | | | (_| | | | | (_| | (_) \__ \ | | |
 \__,_|_|  \__,_|_| |_|\__, |\___/|___/_| |_|
                       |___/

arangosh (ArangoDB 3.12.2 [linux] 64bit, using jemalloc, build
refs/tags/v3.12.2 154c4c5632c, VPack 0.2.1, RocksDB 7.2.0, ICU
64.2, V8 12.1.165, OpenSSL 3.3.1 4 Jun 2024, build-id:
0f674dfac5247637960930fec4f02acafd5016b8)
Copyright (c) ArangoDB GmbH
This executable uses the GNU C library (glibc), which is
licensed under the GNU Lesser General Public License (LGPL),
see https://www.gnu.org/copyleft/lesser.html and
https://www.gnu.org/licenses/gpl.html

Command-line history will be persisted when the shell is
exited. You can use `--console.history false` to turn this off
Connected to ArangoDB 'http+tcp://127.0.0.1:8529, version:
3.12.2 [SINGLE, server], database: '_system', username: 'root'

Type 'tutorial' for a tutorial or 'help' to see common examples
127.0.0.1:8529@_system>
```

Javascript API

```
127.0.0.1:8529@_system> db._databases();
[
  "_system",
  "wlgs"
]

127.0.0.1:8529@_system> db._useDatabase("wlgs");
true

127.0.0.1:8529@wlgs> db._collections();
[
  [ArangoCollection 41321052, "_analyzers" (type document, status loaded)],
  [ArangoCollection 41321057, "_appbundles" (type document, status loaded)],
  [ArangoCollection 41321056, "_apps" (type document, status loaded)],
  [ArangoCollection 41321053, "_aqlfunctions" (type document, status loaded)],
  [ArangoCollection 41321058, "_frontend" (type document, status loaded)],
  [ArangoCollection 41321051, "_graphs" (type document, status loaded)],
  [ArangoCollection 41321055, "_jobs" (type document, status loaded)],
  [ArangoCollection 41321054, "_queues" (type document, status loaded)],
  [ArangoCollection 41321092, "edges" (type edge, status loaded)],
  [ArangoCollection 41321087, "nodes" (type document, status loaded)]
]

127.0.0.1:8529@wlgs> ('b' + 'a' + + 'a' + 'a').toLowerCase()
banana
```

DB initialization

```
db._dropDatabase("wlgs");
db._createDatabase("wlgs");
db._useDatabase("wlgs");
db._createDocumentCollection("nodes");
db._createEdgeCollection("edges");
var graph_module = require("@arangodb/general-graph");
var graph = graph_module._create("graph");
graph;

graph._addVertexCollection("nodes");
var rel = graph_module._relation("edges", ["nodes"], ["nodes"]);
graph._extendEdgeDefinitions(rel);
```

```
#!bin/sh
arangosh --server.username root --server.password test123 --javascript.execute /data/init.js

arangoimport --collection nodes \
    --file "./data/nodes.csv" \
    --type csv \
    --server.database wlgs \
    --server.endpoint http://localhost:8529 \
    --server.username root \
    --server.password test123 \
    --on-duplicate ignore

arangoimport --collection edges \
  --file "./data/edges.csv" \
  --type csv \
  --server.database wlgs \
  --server.endpoint http://localhost:8529 \
  --server.username root \
  --server.password test123 \
  --on-duplicate ignore
```

Import time

```
> time python ./preprocess_csv.py
🔧 Fixing and encoding ./data/popularity_iw.csv... ✅ 1.12s.
🔧 Fixing and encoding ./data/taxonomy_iw.csv... ✅ 10.81s.
🔧 Processing ./data/popularity_iw.csv... ✅ 1.13s.
🔧 Processing ./data/taxonomy_iw.csv... ✅ 16.67s.
28.59s user 0.87s system 98% cpu 29.772 total
```

```
# time ./data/init.sh
.
.
.
created:          5483248
warnings/errors:  288361
updated/replaced: 0
ignored:          0
lines read:       5771611
real    7m 15.72s
user    0m 4.29s
sys     0m 2.41s
```

Python driver

```python
from arango import ArangoClient

client = ArangoClient(hosts="http://localhost:8529")
db = client.db("wlgs", username="root", password="test123"

aql_query = """
UPDATE @node_key WITH { popularity: @new_popularity } IN nodes
RETURN OLD
"""

cursor = db.aql.execute(
    aql_query,
    bind_vars={'node_key': node_key,
               'new_popularity': str(new_popularity)}
)
```

```python
aql_query = """
FOR v IN nodes
    LET incoming = (
        FOR vertex IN 1..1 INBOUND v GRAPH 'graph'
            RETURN vertex
    )
    FILTER LENGTH(incoming) == 0
    RETURN {
        key: v._key,
        title: v.title,
        popularity: v.popularity
    }
"""
cursor = db.aql.execute(aql_query)
results = [doc for doc in cursor]
```

Cool stuff

K-paths

Task 14 was a breeze

```
FOR path
  IN MIN..MAX OUTBOUND|INBOUND|ANY K_PATHS
  startVertex TO targetVertex
  GRAPH graphName
  [OPTIONS options]
```

https://docs.arangodb.com/3.13/aql/graphs/k-paths/

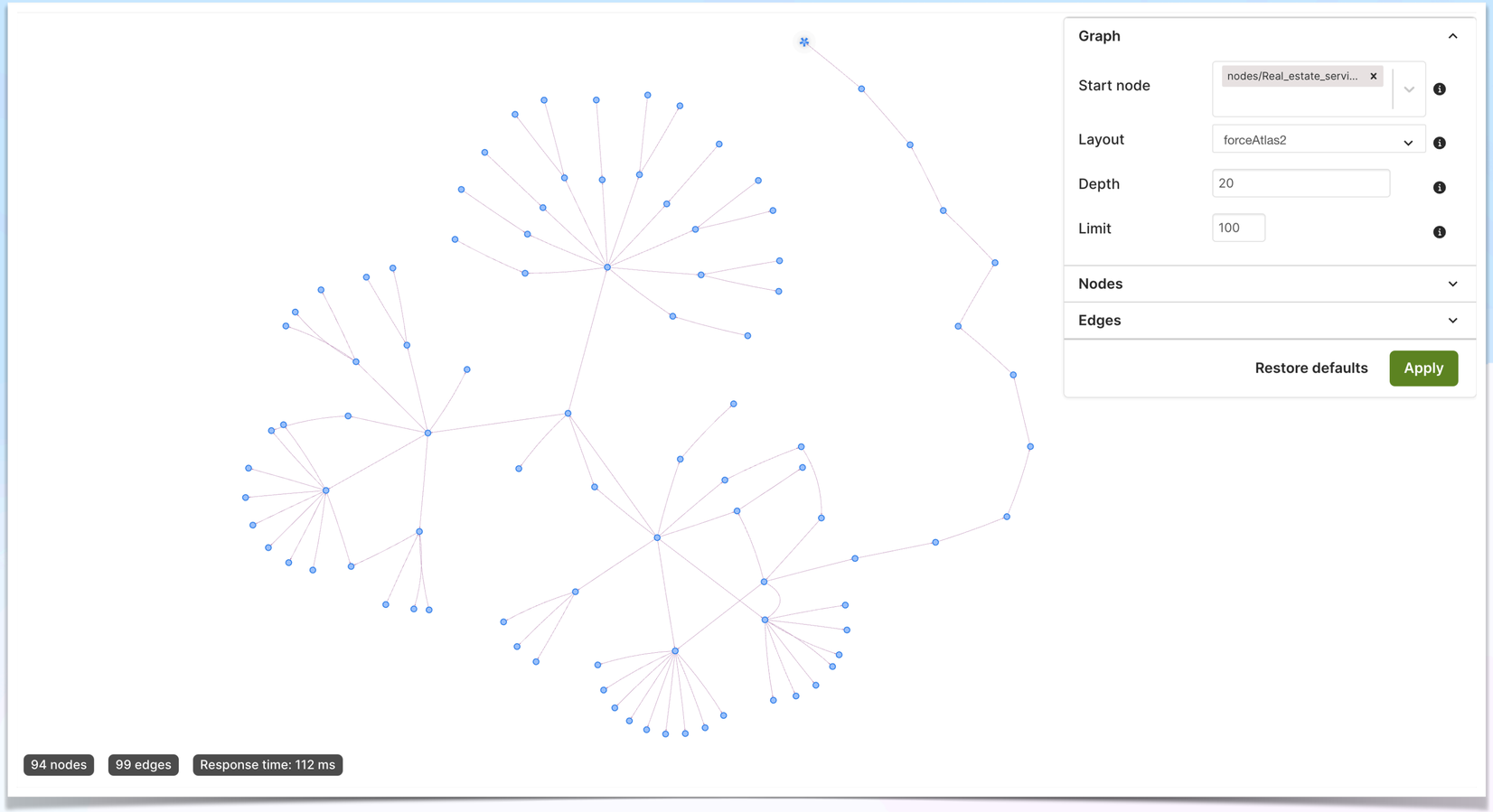
.getExtra()

```
127.0.0.1:8529@wlgs> db._query(`
...>   LET neighborhood = (
...>         FOR v, e, p IN 0..@radius ANY @start_vertex
...>         GRAPH 'graph'
...>         RETURN DISTINCT {
...>             key: v._key,
...>             popularity: TO_NUMBER(v.popularity)
...>         }
...>     )
...>     RETURN {
...>         total_popularity: SUM(neighborhood[*].popularity),
...>         node_count: LENGTH(neighborhood)
...>     }
...> `,
...> {radius: 5, start_vertex: 'nodes/Microeconomics' },
...> {},
...>  { fullCount: true }
...> ).getExtra();
```


```
{
  "warnings" : [ ],
  "stats" : {
    "writesExecuted" : 0,
    "writesIgnored" : 0,
    "documentLookups" : 0,
    "seeks" : 0,
    "scannedFull" : 0,
    "scannedIndex" : 5934909,
    "cursorsCreated" : 2,
    "cursorsRearmed" : 251016,
    "cacheHits" : 171576,
    "cacheMisses" : 79442,
    "filtered" : 125560,
    "httpRequests" : 0,
    "fullCount" : 1,
    "executionTime" : 13.531461615000808,
    "peakMemoryUsage" : 222593024,
    "intermediateCommits" : 0
  }
}
```


https://docs.arangodb.com/3.13/aql/execution-and-performance/query-statistics/

Visualization*



*ArangoDB Community Edition feature

Not so cool stuff

Write operation locks collection for read

Design limitations

The following design limitations are known for AQL queries:

- Up to v3.12.0, subqueries used inside expressions are pulled out of these expressions and executed beforehand. That means that subqueries do not participate in lazy evaluation of operands, for example in the **ternary operator** or when used as sub-expressions that are combined with logical `AND` or `OR`.

  From v3.12.1 onward, short-circuiting is applied.

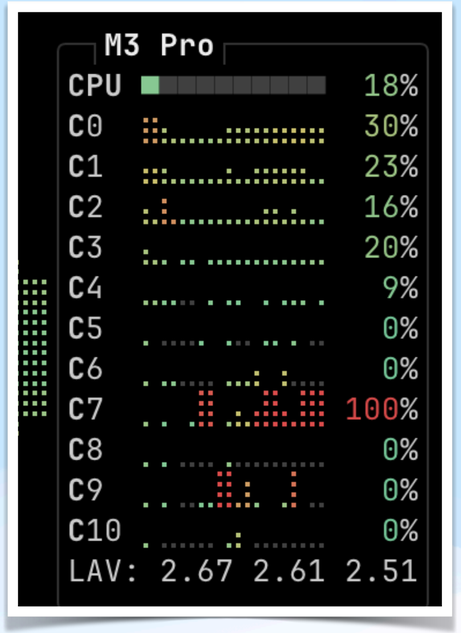
  Also see **evaluation of subqueries**.

- It is not possible to use a collection in a read operation after it was used for a write operation in the same AQL query.

- In the cluster, all collections that are accessed **dynamically** by **traversals working with collection sets** (instead of named graphs) must be stated in the query's initial `WITH statement`. To make the `WITH` statement required in single server as well (e.g. for testing a migration to cluster), please start the server with the option `--query.require-with`.

https://docs.arangodb.com/3.11/aql/fundamentals/limitations/#design-limitations

Single threaded queries




```
 ✔ Container arangodb  Created                                                                                    0.0s
Attaching to arangodb
arangodb  | 2025-01-22T15:32:14.157980Z [1-2] INFO [e52b0] {general} ArangoDB 3.12.2 [linux] 64bit, using jemalloc, build refs/tags/v3.12.2
154c4c5632c, VPack 0.2.1, RocksDB 7.2.0, ICU 64.2, V8 12.1.165, OpenSSL 3.3.1 4 Jun 2024, build-id: 4a5d39cb634736e1e6dc8c4f25b92a395087b915
arangodb  | 2025-01-22T15:32:14.157989Z [1-2] INFO [11111] {general} This executable uses the GNU C library (glibc), which is licensed under the
GNU Lesser General Public License (LGPL), see https://www.gnu.org/copyleft/lesser.html and https://www.gnu.org/licenses/gpl.html
arangodb  | 2025-01-22T15:32:14.158244Z [1-2] INFO [a1c60] {syscall} file-descriptors (nofiles) hard limit is 1048576, soft limit is 1048576
arangodb  | 2025-01-22T15:32:14.158340Z [1-2] INFO [75ddc] {general} detected operating system: Linux version 6.11.5-orbstack-00280-g96d99c92a42b
(orbstack@builder) (ClangBuiltLinux clang version 18.1.8 (https://github.com/llvm/llvm-project.git 3b5b5c1ec4a3095ab096dd780e84d7ab81f3d7ff),
ClangBuiltLinux LLD 18.1.8) #51 SMP Sun Nov  3 08:07:37 UTC 2024
arangodb  | 2025-01-22T15:32:14.158370Z [1-2] INFO [25362] {memory} Available physical memory: 6275833856 bytes, available cores: 11
```

Key limitations

- It must consist of the letters `A` to `Z` (lower- and uppercase), the digits `0` to `9`, or any of the following punctuation characters: `_` `-` `.` `@` `(` `)` `+` `,` `=` `;` `$` `!` `*` `'` `%` `:`

  > Avoid using `:` in document keys as this can conflict with the requirements of **EnterpriseGraphs**, **SmartGraphs**, and **SmartJoins** which use the colon character as a separator in keys.

- Any other characters, especially multi-byte UTF-8 sequences, whitespace, or punctuation characters not listed above cannot be used inside key values.

```
SAFE_CHARACTERS = "_-.@()+=;$!*'%:"
key = urlencode("Ō∑ęąśə&żźć®†īń©ķ√ļ¨^óΔŻłńĶ", safe=SAFE_CHARACTERS)
key = key.replace('~', '%7E')
key = key.replace(':', '%3A')
```

https://docs.arangodb.com/3.11/concepts/data-structure/documents/#document-keys

Memory

```
/app # ./dbcli 14 Microeconomics Service_companies

Error finding paths: [HTTP 500][ERR 32] AQL: global memory limit
exceeded [node #11: SubqueryStartNode] [node #4: CalculationNode]
[node #5: EnumerateListNode] [node #6: CalculationNode] [node #12:
SubqueryEndNode] [node #9: CalculationNode] [node #9:
CalculationNode] [node #10: ReturnNode] (while executing)
```

```
FOR path
IN 1..999 OUTBOUND K_PATHS
@start_vertex TO @end_vertex
GRAPH 'graph'
OPTIONS {
    uniqueVertices: 'path',
}
RETURN {
    vertices: (
        FOR vertex IN path.vertices
            RETURN {
                key: vertex._key,
                title: vertex.title
            }
    ),
    edges: LENGTH(path.edges),
    weight: path.weight
}
```

Thank you