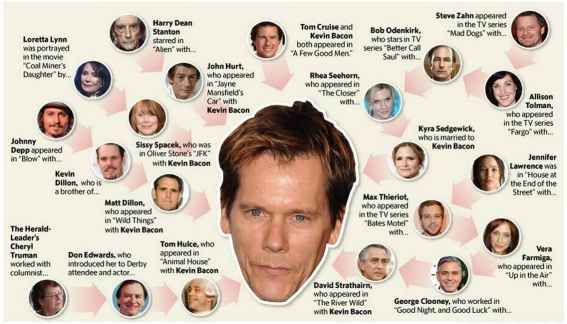
ArangoDB



Source: Chris Ware, Lexington Herald-Leader

ArangoDB Summit 2022

6 Degrees (in any K_PATHS) of Kevin Bacon:

Testing ArangoDB performance and usability with 100 Million IMDB records

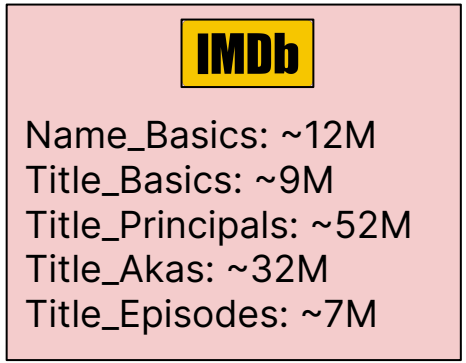
Jon Goldberg

Goldberg Corporate Finance

Problem

- A client (a movie producer) asked me to look at alternatives to a SAAS system used to research information about movies, actors, directors, etc.

- As part of my research, I found that IMDB provides significant part of their data as downloadable data sets (>100M records).
  - These can be downloaded at: https://datasets.imdbws.com



- This seemed like a good excuse to explore the potential of a graph database on a relatively large, but easily understood, set of data.

Exploration Phase

- I gave a quick look at several graph database, providers including:
  - Neo4J, TigerGraph, and ArangoDB.

- Why I chose ArangoDB?
  - While all the systems have their pros and cons, ArangoDB just worked.
  - I followed instructions for both the Desktop installation and the Docker version and it worked as described (for the most part).
    - The only caveat is that (as of 3.9.2) ArangoDB requires AVX extensions. This means you need to install a special version to use on machines with very old processors (pre-2011 Intel or AMD, i.e. my old dual-Xeon workstation), or new Apple M1 machines.
  - After giving ArangoDB a quick spin, I found the GUI, CLI and AQL to be very intuitive.
  - Overall the speed of the system was impressive.
    - e.g. Using MySQL, just to import a TSV (without commercial ETL tools) you have to go through a series of steps and even then you are not guaranteed the whole (massive) file will import.

End Goal - Simple Interactive Test Solution

- Inspired by 6-degrees of Kevin Bacon
  - Six Degrees of Kevin Bacon is a game where players challenge each other to choose two actors and then connect them via films that both have appeared in together. It is a reference to the concept "six degrees of separation", which assumes any two people on Earth are six or fewer acquaintance links apart.
    - See: https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon
- App Features
  - Quickly search data set for partial name of person.
  - Select any 2 people (actors, directors, producers) and find the relationship between them. e.g. 6 Degrees of Kevin Bacon.
  - Create network graph from resulting datasets and traverse this dataset interactively.
- Expected Problems
  - Figure out the best way to utilize/optimize AQL, Arango Indexes and ArangoSearch.
  - Choose and implement graphing tool.
  - Create user interface.

Project Structure

- Data Load From Workstation
  - Workstation - Windows 10, Arango 3.9.2 Client Tools.
  - Batch Script Using *arangoimport.*
- ArangoDB Server
  - ArangoDB on dedicated “Server”
    - Server is a laptop (c. 2012) – i72630QM, 16GB Memory.
    - Windows 10, Arango 3.9.2 Client Tools.
    - Arango (3.9.2) running under Docker.
- ArangoDB Web Interface and Display Layer
  - Queries and Structure
    - Python 3.10.6
      - (Non-Standard) Libraries: python-arango, pandas.
    - AQL Files (Imported by Python).
  - Display
    - Python 3.10.6
      - (Non-Standard) Libraries: dash, dash-cytoscape, dash-bootstrap-components, beautifulsoup4, pynput.
    - A small amount of Javascript for Context Menu.

Batch Script and Arango Import

- I wanted to create a script to recreate everything from scratch if needed.
  The Batch Script:
  - Recreates the ArangoDB Docker instance with the latest version
  - Creates the new Database (Movies), user, all the required collections with a Python Script
    - This could probably all be done in arangosh with a native Node.js script. For a few of the large queries, this probably would be better, but I decided to try everything in Python.
  - Imports the Newest IMDB files from the IMDB website:
    - Downloads (e.g.)

      rm name.basics.tsv.gz
      wget https://datasets.imdbws.com/name.basics.tsv.gz
      gzip -d name.basics.tsv.gz
    - Imports with arangoimport (e.g.)

      arangoimport --file name.basics.tsv --server.endpoint tcp://MYSERVERIP:8529 --server.password MYROOTPASSWORD --server.database Movies --collection=name_basics --create-collection=true --type=tsv --translate "nconst=_key"

Create Analyzers, Views, Graphs, Indexes, Nodes and Edges

- This is also done with a Python Script.
  - As many of these operations can take a while, if using Python (or another language), be sure to use an Asynchronous function or “in_background” property when possible, as the Python driver (and assumedly other drivers) will time out otherwise.
- The order to Create are:
  - Custom Nodes and Edges
    - Create all Vertex Nodes in 1 collection.
      - Can be either a Title or a Person - Total of ~21M Nodes.
      - Had to do some special processing to extract some additional Vertex Nodes since IMDB did not create a record for every Person in the Title_Basics collection.
    - Create all Edges in collection.
      - Uni-direction - _from Title, _to Person - Total of ~57M Edges.
  - Indexes
  - Custom Analyzers - No time to setup.
  - Views - Uses Custom (or standard) Analyzers.

Screenshot

Find Person from full or partial name

○Like Query®Levenshtein Distance

Levenshtein Queries are quicker but sometimes will not bring back all the desired results

Kevin Bacon | Find Person

The input value was "Kevin Bacon" and the button has been clicked 0 times

Selected Person

nconst	primaryName	KnownForTitles	Simularity
nm3636162	Kevin Bacon	|"See Girl Run","Sunset Park","Long Nights Short Mornings","Hello I Must Be Going"|	1
nm12606581	Kevin Bacon	|"The Devil You Know"|	1
nm0000102	Kevin Bacon	|"The Woodsman","Mystic River","Hollow Man","Footloose"|	1
nm9323132	Kevin Bacon	|"The Comsat Angels Live 12-12-1981","Super Dark Times","The Old Grey Whistle Test"|	1
nm11500328	Kevin Bacon	|"Nova","Queen of the Oil Patch"|	1
nm13843189	Kevin Bacon	|"Secrets, Lies and DNA Ties"|	1
nm10210267	Kevin Bacon	|"On the Case with Paula Zahn"|	1
nm4025714	Kevin Bacon	|"The Editor","The Suit","The Return","The Return of Swamp Thing"|	1
nm9792572	Kevin Bacon	|"Chowboys: An American Folktale","Sel et Diesel","Sous la coupole","Hors Québec"|	1
nm9449507	Kevin Basconi	|"Sid Roth's It's Supernatural"|	0.7777777910232544

Node 1:

Kevin Bacon
nm0000102

Node 2:

Sissy Spacek
nm0000651

Show Relationship

The button has been clicked 0 times. And the Relationship between nm0000102 and nm0000651 is:

Kevin Bacon-nm3636162

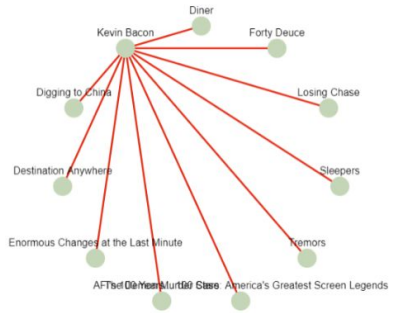
Kevin Bacon *actor* **in/of** The Gift *composer* **in/of** George Aliceson Tipton *composer* **in/of** Badlands *actress* **in/of** Sissy Spacek
Kevin Bacon *actor* **in/of** The Demon Murder Case *composer* **in/of** George Aliceson Tipton *composer* **in/of** Badlands *actress* **in/of** Sissy Spacek
Kevin Bacon *actor* **in/of** Enormous Changes at the Last Minute *actor* **in/of** David Strathairn *actor* **in/of** Beyond the Call *actress* **in/of** Sissy Spacek
Kevin Bacon *actor* **in/of** Footloose *actor* **in/of** John Lithgow *self* **in/of** Episode #13.29 *self* **in/of** Sissy Spacek

Selected Node:

{"id": "movie_nodes/nm0000102", "label": "Kevin Bacon"}

Add as Node 1 | Add as Node 2 | Total Connected Nodes:597 | Nodes on Page: 10 | Current Page Number: 1

First 10 Related items for Selected Person or Title:



Goldberg Corporate Finance

ArangoDB

Name Search Query

- Initial Search
  - Provide 2 options to test the response time and results.
  - View (name_basics_primaryName_view) uses the **text_en** and **identity** analyzer on the primaryName.
    - Levenshtein Match

```
/* Uses The LEVENSHTEIN Match.  Gives pretty good quick results. */
FOR name IN name_basics_primaryName_view
        SEARCH LEVENSHTEIN_MATCH(name.primaryName, @myname, 2, false)
        LET mytitles = {nconst: name._key, primaryName: name.primaryName, kftitle: name.knownForTitles}
        LIMIT 5000
        FOR all_kftitle in SPLIT(mytitles.kftitle,',')
                FOR title in title_basics FILTER title._key == all_kftitle
                        COLLECT nconst = name._key, myprimaryName = name.primaryName,mysim=NGRAM_SIMILARITY(name.primaryName, @myname, 3) INTO
groupedTitles SORT mysim DESC LIMIT 10
        RETURN {nconst: nconst, primaryName: myprimaryName, KnownForTitles: SUBSTITUTE( groupedTitles[*].title.primaryTitle, ["[", "]"], "|" ), Simularity: mysim}
```

    - LIKE Match

```
/* Uses LIKE Match.  More like traditional searches but slower */
FOR name IN name_basics_primaryName_view
        SEARCH ANALYZER(LIKE(name.primaryName, CONCAT('%',@myname,'%')), "identity")
        LET mytitles = {nconst: name._key, primaryName: name.primaryName, kftitle: name.knownForTitles}
        LIMIT 10
        FOR all_kftitle in SPLIT(mytitles.kftitle,',')
                FOR title in title_basics FILTER title._key == all_kftitle
                        COLLECT nconst = name._key, myprimaryName = name.primaryName,mysim=NGRAM_SIMILARITY(name.primaryName, @myname, 3) INTO
groupedTitles SORT mysim DESC LIMIT 10
        RETURN {nconst: nconst, primaryName: myprimaryName, KnownForTitles: SUBSTITUTE( groupedTitles[*].title.primaryTitle, ["[", "]"], "|" ), Simularity: mysim}
```

Name Search Results

- Uses the Python Dash Table Component to generate the results:

Find Person from full or partial name

○ Like Query ◉ Levenshtein Distance

Levenshtein Queries are quicker but sometimes will not bring back all the desired results

Kevin Bacon | Find Person

The input value was "Kevin Bacon" and the button has been clicked 0 times

Selected Person

nconst	primaryName	KnownForTitles	Simularity
nm3636162	Kevin Bacon	|"See Girl Run","Sunset Park","Long Nights Short Mornings","Hello I Must Be Going"|	1
nm12606581	Kevin Bacon	|"The Devil You Know"|	1
nm0000102	Kevin Bacon	|"The Woodsman","Mystic River","Hollow Man","Footloose"|	1
nm9323132	Kevin Bacon	|"The Comsat Angels Live 12-12-1981","Super Dark Times","The Old Grey Whistle Test"|	1
nm11500328	Kevin Bacon	|"Nova","Queen of the Oil Patch"|	1
nm13843189	Kevin Bacon	|"Secrets, Lies and DNA Ties"|	1
nm10210267	Kevin Bacon	|"On the Case with Paula Zahn"|	1
nm4025714	Kevin Bacon	|"The Editor","The Suit","The Return","The Return of Swamp Thing"|	1
nm9792572	Kevin Bacon	|"Chowboys: An American Folktale","Sel et Diesel","Sous la coupole","Hors Québec"|	1
nm9449507	Kevin Basconi	|"Sid Roth's It's Supernatural"|	0.7777777910232544

Bacon Number - Count Degrees of Separation

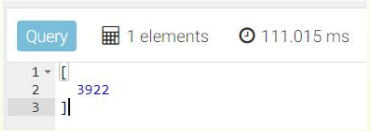
The Bacon number of an actor is the number of degrees of separation he or she has from Bacon, as defined by the game. This is an application of the Erdős number concept to the Hollywood movie industry. The higher the Bacon number, the greater the separation from Kevin Bacon the actor is.

- The computation of a Bacon number for actor X is a "shortest path" algorithm, applied to the co-stardom network:
- Kevin Bacon himself has a Bacon number of 0.
- Those actors who have worked directly with Kevin Bacon have a Bacon number of 1.
- If the lowest Bacon number of any actor with whom X has appeared in any movie is N, X's Bacon number is N+1.

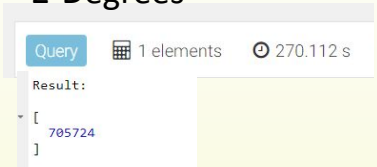
```
RETURN COUNT(FOR v IN @2xbacon_num1..@2xbacon_num2 ANY CONCAT("movie_nodes/",@startperson)
        GRAPH "Movie_Graph"  OPTIONS { uniqueVertices: "global", bfs: true }
RETURN v._key)
```

NOTE(s): 1. The bind parameters are 2x the Bacon Number (e.g., 2 to 2 for Bacon Number 1) since there are Movie Nodes in between each related person.
2. This solution is a little different than the original intention of the “Bacon Number”. It includes many places where a person is just mentioned or used in historical footage (e.g. job=“self”).

1-Degree



2-Degrees



3-Degrees

Query took about 5 hours to Run

Results: 4.640.071 Nodes

Goldberg Corporate Finance

ArangoDB

Key Queries (2)

- Relationship Between Persons

```
FOR p IN 1..6 ANY K_PATHS CONCAT('movie_nodes/',@nconst1) TO CONCAT('movie_nodes/',@nconst2) GRAPH 'Movie_Graph' LIMIT 4 LET mynodes =
p.vertices[*].NameOrTitle
   LET MYPATHS = INTERLEAVE(["<br>"],mynodes,p.edges[* RETURN CONCAT(' <i>',CURRENT.job,'</i> ', '<b> in/of </b> ')])
   FOR DOC IN MYPATHS
      RETURN DOC
```

- This creates a “verbal” description of the relationship between 2 persons.

Kevin Bacon *actor* **in/of** The Gift *composer* **in/of** George Aliceson Tipton *composer* **in/of** Badlands *actress* **in/of** Sissy Spacek
Kevin Bacon *actor* **in/of** The Demon Murder Case *composer* **in/of** George Aliceson Tipton *composer* **in/of** Badlands *actress* **in/of** Sissy Spacek
Kevin Bacon *actor* **in/of** Enormous Changes at the Last Minute *actor* **in/of** David Strathairn *actor* **in/of** Beyond the Call *actress* **in/of** Sissy Spacek
Kevin Bacon *actor* **in/of** Footloose *actor* **in/of** John Lithgow *self* **in/of** Episode #13.29 *self* **in/of** Sissy Spacek

- NOTE: That the above query limits the search to a Bacon Number of 3 which provides a connection for almost (but not all) persons in the collection.

Cytoscape

- Originally wrote the visualization in JS with vis.js but moved to Cytoscape when rewritten in Python.
- Need to Generate a dataset (NODES+EDGES) from Arango for Cytoscape:
  - Broken into NODES

```
FOR vertex, edges IN 1..1 ANY CONCAT('movie_nodes/', @nconst) movie_edges
   LIMIT @myoffset, @mycount
         RETURN  {'data': {id: CONCAT('movie_nodes/', vertex._key), label: vertex.NameOrTitle}}
```

  - And EDGES

```
FOR vertex, edges IN 1..1 ANY CONCAT('movie_nodes/', @nconst) movie_edges
   LIMIT @myoffset, @mycount
         RETURN  {'data': {source: edges._from, target: edges._to, job: edges.job}}
```

  - Additionally as this only gets the end points (e.g. the films for a person or the persons in a film), you need to add on the Center Node.
  - NOTE: the @myoffset and @mycount parameters allow the user to “scroll” through nodes.

DEMO

Final Notes

- Be careful with generating edges and relationships.
  It can get **very big very fast**.
  $$C(n,r) = \binom{n}{r} = \frac{n!}{(r!\,(n-r)!)}$$
  - e.g. there would be 1225 unique relations for 50 people (n=50, r=2):
  - I tried creating a separate edge database of just unique person-to-person relationships to speed up the (bigger) k_paths query. — I gave up after a day of processing and creating 150M edge records.
    - If anyone has access to a cluster / supercomputer that would like to see if this speeds up the search, it would be an interesting experiment.
- For simple user interface use-cases, ArangoDB users might want to check out the Open Source, Low-Code platform Budibase (https://budibase.com). It comes with a built in ArangoDB connector and for basic data retrieval and integration works pretty well.
- As can be seen, the time to calculate goes up exponentially with large datasets. Make sure to optimize your queries on smaller sets:
  - Read the Documentation on optimization: https://www.arangodb.com/learn/development/aql-query-optimization-with-profiler
- Finally, remember that ArangoDB is **NOT** a RDBMS. Some of the traditional ways of thinking about data organization and normalization transfer over, and some do not.

Thank You

THANK YOU

Jon Goldberg

jon_d_goldberg@hotmail.com
jon.goldberg@goldbergcf.com
Mobile (AT): +43 681 8170 5887
Mobile (US): +1 303 847 5384
GitHub: https://github.com/jdgamsterdam
LinkedIn: https://www.linkedin.com/in/jondgoldberg

Goldberg Corporate Finance

ArangoDB