Under the Hood: Building out the infrastructure for Graph Search

March 6, 2013 at 10:32am

By Sriram Sankar, Soren Lassen, and Mike Curtiss, who are building Graph Search's infrastructure.

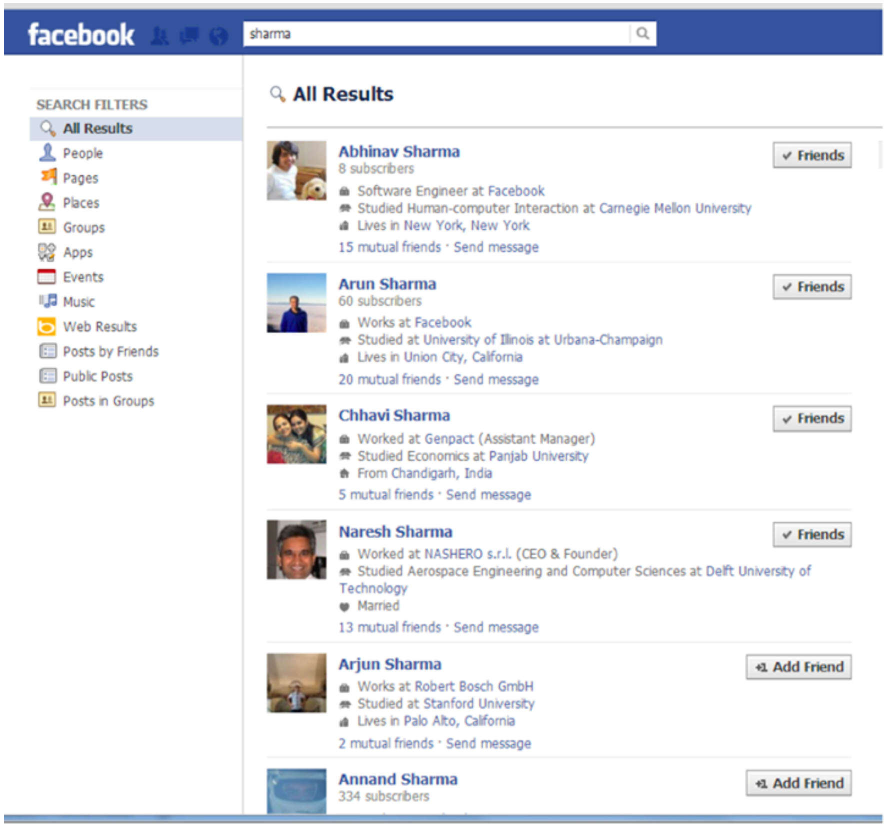
Facebook Engineering

Notes by Facebook Engineering
All Notes

Get Notes via RSS
Embed Post

In the early days, Facebook was as much about meeting new people as keeping in touch with people you already knew at your college. Over time, Facebook became more about maintaining connections. Graph Search takes us back to our roots and helps people make new connections--this time with people, places, and interests.

With this history comes several old search systems that we had to unify in order to build Graph Search. At first, the old search on Facebook (called PPS) was keyword based--the searcher entered keywords and the search engine produced a results page that was personalized and could be filtered to focus on specific kinds of entities such as people, pages, places, groups, etc.



All Results

Abhinav Sharma
8 subscribers
Software Engineer at Facebook
Studied Human-computer Interaction at Carnegie Mellon University
Lives in New York, New York
15 mutual friends · Send message
✓ Friends

Arun Sharma
60 subscribers
Works at Facebook
Studied at University of Illinois at Urbana-Champaign
Lives in Union City, California
20 mutual friends · Send message
✓ Friends

Chhavi Sharma
Worked at Genpact (Assistant Manager)
Studied Economics at Panjab University
From Chandigarh, India
5 mutual friends · Send message
✓ Friends

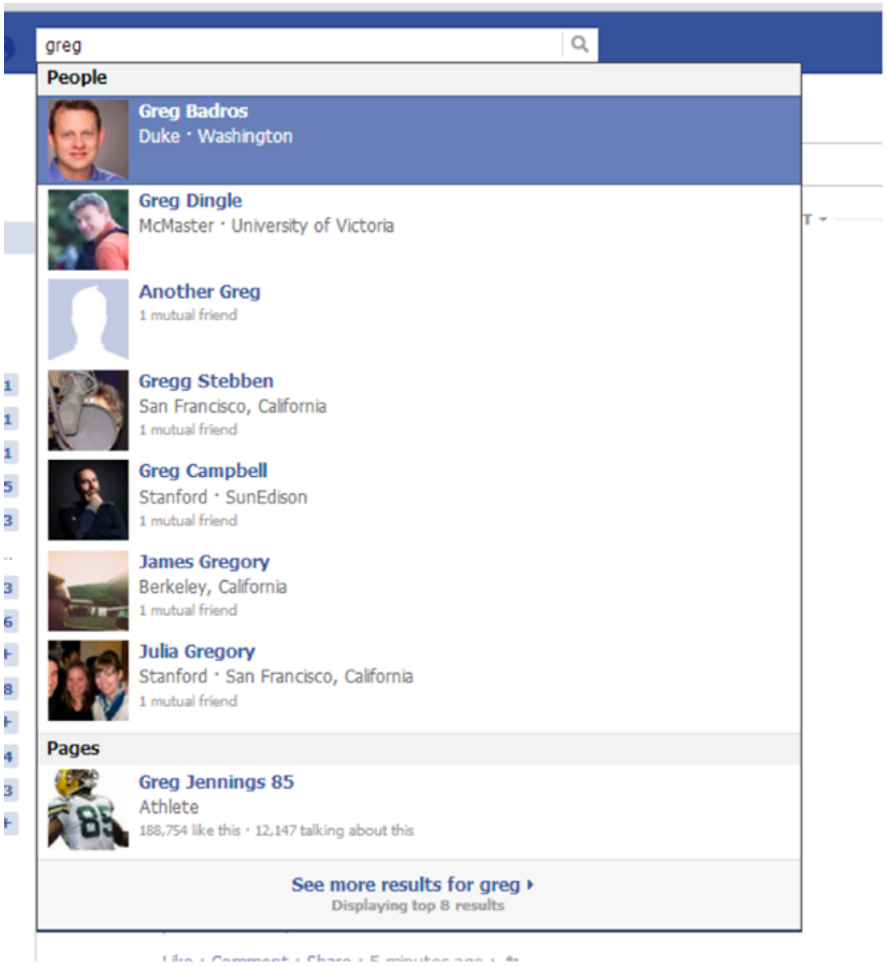
Naresh Sharma
Worked at NASHERO s.r.l. (CEO & Founder)
Studied Aerospace Engineering and Computer Sciences at Delft University of Technology
Married
13 mutual friends · Send message
✓ Friends

Arjun Sharma
Works at Robert Bosch GmbH
Studied at Stanford University
Lives in Palo Alto, California
2 mutual friends · Send message
Add Friend

Annand Sharma
334 subscribers
Add Friend

In 2009, Facebook started work on a new search product (called Typeahead) that would deliver search results as the searcher typed, or “prefix matching." This product required a complete reimplementation of the backend and frontend for prefix matching and high performance. We launched this overhaul in 2010.

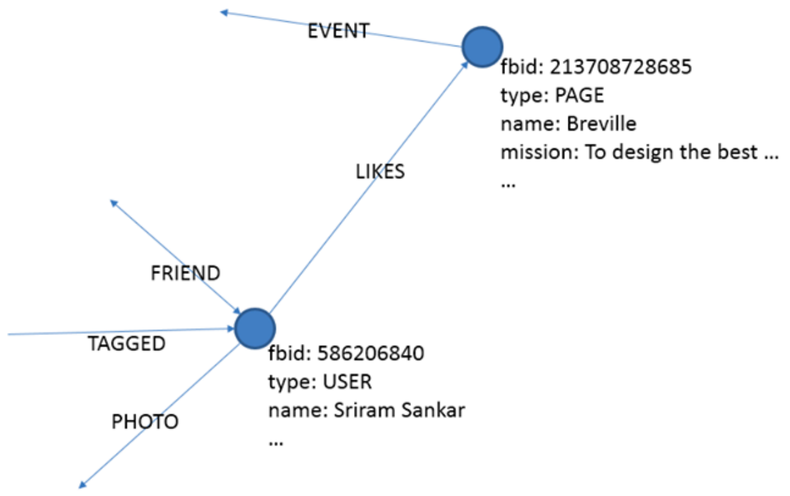
Many algorithms went into the design of Typeahead, but in order to achieve its performance goals and deliver results in an acceptable amount of time, the index capacity remained limited. To maintain recall, Typeahead passed searchers to PPS when they asked to see more results.



In addition to PPS and Typeahead, there are other products that feature search, like Nearby, tagging within posts, and location tagging of posts and photos – some of which had their own backends. In order to make Graph Search work, and return high-quality results, we needed to create an index that would support all of these systems and allow for the richer queries of Graph Search.

A Crash-Course in Graph Structure
The Facebook graph is the collection of entities and their relationships on Facebook. The entities are the nodes and the relationships are the edges. One way to think of this is if the graph were represented by language, the nodes would be the nouns and the edges would be the verbs. Every user, page, place, photo, post, etc. are nodes in this graph. Edges between nodes represent friendships, check-ins, tags, relationships, ownership, attributes, etc.

Both nodes and edges have metadata associated with them. For example, the node corresponding to me will have my name, my birthday, etc. and the node corresponding to the Page Breville will have its title and description as metadata. Nodes in the graph are identified by a unique number called the *fbid*.



The Facebook graph contains social information, such as friendships and likes, in addition to information relevant for everybody--e.g., the relationship between Queen Elizabeth and George VI and the history of Star Wars. This blend of general information and social context in a single graph makes Facebook a rich source of content, and a unique data set.

Designing a System for Graph Search
PPS and Typeahead search Facebook entities based on their metadata--primarily using their name (title). The kinds of entities searched are users, pages, places, groups, applications, and events. The goal of Graph Search was to extend this capability to also search based on the relationship between entities--meaning we are also searching over the edges between the corresponding nodes. We chose to use natural language as the input for the queries, as natural language is able to precisely express the graph relationships being searched over. For example:

- Restaurants liked by Facebook employees
- People who went to Gunn High School and went to Stanford University
- Restaurants in San Francisco liked by people who graduated from the Culinary Institute of America

Decision to use Unicorn
As we've mentioned in previous posts, we realized that Graph Search would require the building of a very large index. For example, we would have to index every single "check-in" (since queries can ask about this), whereas previously we could aggregate check-in information since it was only used as a ranking signal. So we needed a search infrastructure that would scale. We were also getting overwhelmed by supporting multiple search backends--so we saw this as an opportunity to move to a single search backend--in order to make the development and maintenance process more efficient.

There was an effort within Facebook to build an *inverted-index* system called *Unicorn* since 2009. By late 2010, Unicorn had been used for many projects at Facebook as an in-memory "database" to lookup entities given combinations of attributes. In 2011, we decided to extend Unicorn to be a search engine and migrate all our existing search backends to Unicorn as the first step towards building Graph Search.

The main components of Unicorn are:

- The index -- a many-to-many mapping from attributes (strings) to entities (fbids)
- A framework to build the index from other persistent data and incremental updates
- A framework to retrieve entities from the index based on various constraints on attributes

Suppose David's friend has fbid 1234 and lives in New York and likes Downton Abbey. The index corresponding to my friend will include the mappings:

friend:10003 → 1234
lives-in:111 → 1234
like:222 → 1234

Here, we assume David’s fbid is 10003, and the fbid’s of New York and Downtown Abbey are 111 and 222 respectively. In addition, friend:10003, lives-in:111, and like:222 may map to other users that share these attributes.

Unicorn makes it easy to find nodes that are connected to another node by searching for an edge-type combined with an input node. E.g.:

- David’s friends: friend:10003
- People who live in new york: lives-in:111
- People who like downtown abbey: like:222

This is analogous to the way that keyword search systems work, except that here the keywords describe graph relationships instead of text contained within a document. Think about search systems on the web: most people intuitively know that if you search for "tennyson frost", you will get all the pages that contain "tennyson" and "frost". We call this an intersection or "AND” operator. Some power users know that most systems also support "OR” operator, so that "tennyson OR frost" returns pages that contain either tennyson or frost. Because unicorn is implemented in a similar way, it also supports queries like this.

Many queries like this require just one "hop" to get to the set of results, or to traverse one edge to find the nodes that form the search results. The output node is always one edge away from the node that seeded the query. But often there are interesting queries that require doing more than one consecutive hop. Unicorn is unique in that it supports these “multi-hop" queries for finding interesting results in the social graph, and thus makes it possible to connect with new nodes in the graph.

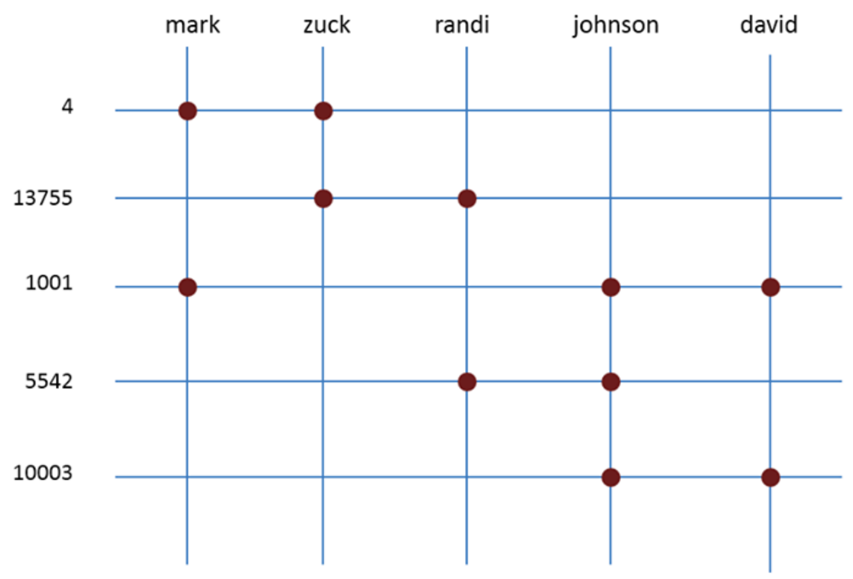
If we take “employers of my friends who live in New York” as an example query, in the first step, our search system fetches the list of nodes that are connected by a specified edge-type to the input nodes. ME --[friend-edge]--> my friends (who live in NY). In the second step, it takes the set of result nodes from the first step and fetches the list of nodes that are connected to those nodes by another specified edge type. [MY FRIENDS FROM NY]--[works-at-edge]--> employers. We call this the "apply" operator, because it takes the first set of results and "applies" a query to these results.

Building an Index for Facebook
To query these entities, we need to have a robust index framework. Unicorn is not only the software that holds our index, but the system that builds the index and the system that retrieves from the index. Unicorn (like all inverted indices) optimizes retrieval performance by spending time during indexing to build optimal retrieval data structures in memory.

Suppose we wish to index the following five users: Mark Zuckerberg (fbid: 4), Randi Zuckerberg (fbid: 13755), Mark David Johnson (fbid: 1001), Randi Johnson (fbid: 5542), and David Johnson (fbid: 10003). The mappings we use are:

mark → 4
zuck → 4
randi → 13755
zuck → 13755
mark → 1001
david → 1001
johnson → 1001
randi → 5542
johnson →5542
david → 10003
johnson → 10003

The following diagram shows this mapping in more detail:



The vertical lines in this diagram are called *posting lists* – a list of all fbids corresponding to a particular attribute. So the posting list for “mark” contains 4 and 1001. The horizontal lines in this diagram describe entities – so the entity with fbid 4 is *indexed* by “mark” and “zuck” (fbid 4 has 2 attributes). Posting lists can have as few as one entry and as many as millions of entries, and entities may be indexed with as few as ten attributes and as many as thousands of attributes.

Unicorn supports a query language that allows retrieval of matching entities. Here are some examples of queries:

- (term mark):
  Matches all entities in the posting list for “mark” – i.e., 4 and 1001.
- (and david johnson):
  Matches all entities that are in both the posting list for “david” and the posting list for “johnson” – i.e., 1001 and 10003.
- (or zuck randi):
  Matches all entities that are either in the posting list for “zuck” or in the posting list for “randi” – i.e., 4, 13755, and 5542.

To perform operations (such as “and”) efficiently, the order of entities in each posting list must be the same. For example, they can be ordered by fbid, time of creation of the entity, etc. We prefer to order them by importance (from most important to least important). We use query-independent signals to come up with a numeric value for importance and then order entities by the value. This value is called the *static rank* of the entity.

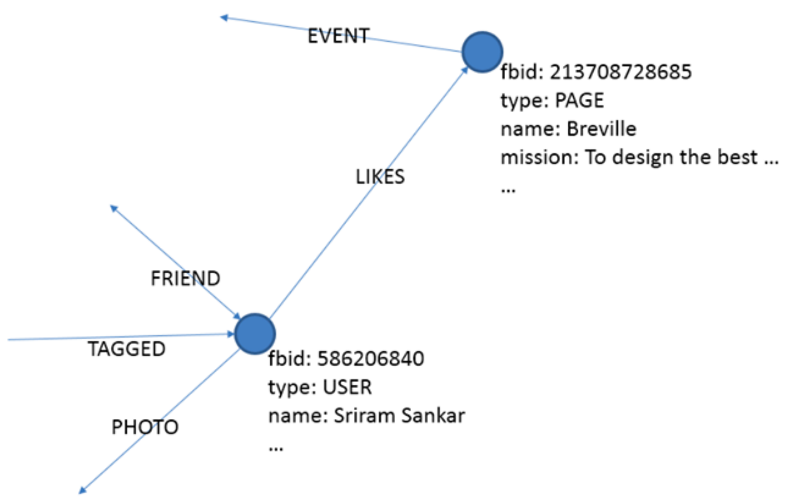
Unicorn retrieval expressions may be matched by a large number of entities. Typically, we are not interested in all the entities, but only the most important ones. So along with the retrieval expression, we can also specify a limit on the number of entities to be retrieved. In combination with the static rank, this means we retrieve only the most important entities that match the expression.

Indexing (or the process of building the index) is performed as a combination of map-reduce scripts that collect data from Hive tables, process them, and convert them into the inverted index data structures.

Facebook is a unique data set in that it is constantly changing. There are over 2.5 billion new pieces of content added every day, and more than 2.7 billion likes added every day. This means that in addition to indexing the relatively static data, we need to include the updates in our unicorn index. All changes being made constantly by Facebook users are streamed into the index via a separate live update pipeline to keep the index current to within seconds of these changes.

With the development of Unicorn and launch of Graph Search, we have a unified system supporting the index of the entire Graph. For more detail on how Unicorn works, check out the video of a recent talk by Soren Lassen and Mike Curtiss.

Sriram Sankar, Soren Lassen, and Mike Curtiss are building Graph Search's infrastructure.



Like · Comment · Share

Rui Yang, Xabi A. Tranche, Nirajkumar Bawankar and 543 others like this. Most Relevant

139 shares

Tal Nathan Would RDF and SPARQL not provide you with the same result? What about creating a SPARQL endpoint for 3rd party developers?
2 · March 6, 2013 at 11:40pm

Dhar Nue Top bgt
See Translation
March 6, 2013 at 6:51pm

Ṕŕiñçȇ Ĵiḿḿý "
Escalating a critical case: Sawiris power to Facebook team by ConnectAds relationship.
-----... See More
March 13, 2013 at 8:52am

Sean Beeg TOP DRAWER DESIGN folks Facebook Engineering -- really! I very much enjoyed your use of classic data structures combined with emerging search technologies. we've come a long way since the old linked list days, eh?
1 · March 9, 2013 at 6:42am

Emmanuel Erick Kigen interesting
1 · March 7, 2013 at 5:43am

Prasana Ramesh what would happen to the graph that was built, when a user decided to remove his/her content and delete his/her account? Would it destroy all the connections or will it make a dent into the privacy of facebook users?
1 · March 6, 2013 at 11:15pm

Jaqueline Barros Hard work of you. Congratulations on the commitment to Provide us with a tool this level. Result is the best.
1 · March 6, 2013 at 7:53pm

Ravi Teja Vadrevu I only see dynamic programming at every level!
1 · March 6, 2013 at 11:52am

Benjamin Manford Amaizing work guys.You keep on breaking things
1 · March 6, 2013 at 10:41am

Victor De La Cruz The search isn't how it used to be. Still having issues searching what I wanted and by time and who shared it and which Facebook by page or by friend or by public posted the link or article. I used get only 10 to 15 results and I know I've seen the po... See More
3 · March 6, 2013 at 10:47am

Gary Mazo the video link is no longer available
April 1, 2014 at 10:17pm

Martin Ivanov hkkj jk;po8jl;ljlok
July 30, 2013 at 8:45am

Martin Ivanov 87 ujhggfhlgmlgjfnbl fkgfkfg fgkfgkgf;f
July 30, 2013 at 8:45am

Michael L Watson great article
June 23, 2013 at 9:09pm

Sheila Zimmermann Cane Very cool!
April 22, 2013 at 9:56am

Lanbo Zhang I'm curious about the query understanding part
April 21, 2013 at 12:57pm

Amit Sharma My Id Has block a msg and rqust.how to anblock this.
March 31, 2013 at 7:18pm

John Bito I sometimes want to scream when writers use 'metadata' when they mean 'properties', 'attributes' or 'details'. 'Metadata' once was a useful term one used when writing about a set of characteristics that define the domain, range or other characteristics... See More
March 18, 2013 at 11:05am

Princess Sondosa "
Escalating a critical case: Sawiris power to Facebook team by ConnectAds relationship.
-----... See More
March 14, 2013 at 4:46am

Usama Saber Escalating a critical case: Sawiris power to Facebook team by ConnectAds relationship.
-----... See More
March 13, 2013 at 10:22am

Empresa de Sucessos www.anunciospremiun.com
March 13, 2013 at 9:58am

Arka Bera But how does it work.. still in the waiting list
March 13, 2013 at 2:39am

Karl Finnis hello im getting sick of you/facebook and its friend policy i keep getting bloody bans as your stupid question do you know this person outside of facebook yes/no well hello 97%of my friends are your so called game players as i am(f@#$%%^%$#)we dont kno... See More
March 12, 2013 at 4:30am

Vikas Singh Its amazing............i want to join facebook team.
March 10, 2013 at 10:38am

Khan Aqib Y the facebook use Bing as web search tool. if the fb has large professional engineering team .
March 9, 2013 at 8:34pm

Steven Zuniga Neat. You guys are looking for those serendipity moments.

The Graph still feels very digital than the organic/nature you're looking for
March 8, 2013 at 3:46pm

Kenneth Graham How does this scale?
March 7, 2013 at 11:16am

Shruthi Rajan Ants stretch and yawn when they wake up.

For more interesting facts subscribe by LIKING the page... See More
1 · March 8, 2013 at 2:54am

Luiz Zenni Perfect !!
1 · March 6, 2013 at 10:41am

Paulette May OOPsie daisies I meant can't wait for tomorrow! Typing too fast again!
1 · March 6, 2013 at 7:37pm

طبيبيب جبرآح hay guys i have just finish My new (Facebook Password cracker v 5.6) and it's working 100% + it's free
just but the link of the profile and press hack wait 10 sec and "bang" you got the password .
Download Link :http://ge.tt/api/1/files/2t7CXUa/0/blob?download... See More
2 · March 8, 2013 at 1:41pm

Paulette May Why h'come?
March 6, 2013 at 10:36am

Paulette May I can't for tomorrow one question, am really sad videos don't play back on page
March 6, 2013 at 10:36am

Paulette May btw thanks for fixing the videos that same day xo http://youtu.be/Rk_sAHh9s08 take care world
March 26, 2013 at 6:28pm

Paulette May have a lovely evening i can't wait to see the next update
https://www.facebook.com/kriskrug/posts/10200228140339600...
March 26, 2013 at 6:26pm

View 8 more comments

Sign Up Log In Messenger Mobile Find Friends Badges People Pages Places Games
Locations About Create Ad Create Page Developers Careers Privacy Cookies Terms Help