Freebase Database System Design
SE 305

Xiaowei LIANG
Zhuodi LIU

December 4, 2015

1 Introduction

Freebase was a large collaborative knowledge base consisting of data composed mainly by its community members. It was an online collection of structured data harvested from many sources, including individual, user-submitted wiki contributions. It contains tens of millions of topics, thousands of types and tens of thousands of properties. In this project, we design a database system to store the data in Freebase and to perform several queries efficiently.

2 Objective

1 Extract useful information of the raw data and store it into MySQL database.

2 Design and optimize the database model.

3 Design several meaningful queries of the database.

4 Design a graphical user interface.

3 Data extraction and storage

There are several kinds of data in Freebase. An entity(also called topic) is an instance of a real world object such as China and Albert Einstein. The number of entity accounts for more than 95% of all data. Domain is a collection of types which share the same namespace. Type denotes a category which an object belongs to. Property represents the attributes an object owns.

Data in Freebase is organized as a triple form

(obj1,prop,obj2).

Both obj1 and obj2 can be entities, types or properties and obj2 can also be a number or a string. An example of triples in Freebase is given below:
(⟨http://rdf.freebase.com/ns/m.0jcx⟩, ⟨http://rdf.freebase.com/ns/book.author.works_written⟩, ⟨http://rdf.freebase.com/ns/m.04wg3nh⟩).

The first item FB/m.0jcx (for simplicity, ”FB/” is used to replace ”http://rdf.freebase.com/ns/”) in this triple stands for Albert Einstein. In Freebase, every object has a mid like m.0jcx and m.04wg3nh which is unique for each object. The third item m.04wg3nh represents a book called ”The Born-Einstein Letters”. Therefore the whole triple shows that Albert Einstein wrote ”The Born-Einstein Letters”. We can find that Freebase uses properties to link two objects and we can extract useful information of an object by checking some important properties.

Property type.object.name connects an mid with its name and we can use this property to find the names of all entities, types, properties and domains. The name of an object has many versions including English and other languages and we just consider English. Property type.object.type relates an object with its type and we use this property to classify different type of data. For example if we find a tuple (obj1,FB/type.object.type,FB/type.type), then we can conclude that obj1 represents a type. Similarly property type.object.domain and type.object.property indicate domain and property data. We separately store domain, type, property and entity to improve the speed of query.

Property type.property.schema and type.property.expected_type connect a property and a type. A tuple (obj1,schema,obj2) indicates that for every tuple (o1,obj1,o2), o1 should has type obj2. On contrary, a tuple (obj1,expected_type,obj2) indicates that for every tuple (o1,obj1,o2), o2 should has type obj2.

Besides these fundamental properties, we also consider some extra useful properties. For example, by checking property common.topic.description, the description of an entity can be found. Furthermore, we want to find the image of an entity and it is tricky in Freebase because property common.topic.image of an entity is not the image but the image content. We have to search the image.content.source of that content and the object corresponding to property type.content_import.uri of image source is the url of the image.

The raw data of Freebase freebase.gz is 29.9G. In the first step, we scan every line and save entities with mid. In the rawdata, there are many tuples whose first item doesn’t start with ”FB/m.”. Some of them start with ”FB/g.” which is called gid. However, gid is not further used in Freebase, so we discard gid tuples. Other tuples’ first items are types, domains and properties. We do not store them either because tuples containing their mid are saved. Then we check the second item of the tuple which is the property. We check whether it is one of the important properties and save the object into the corresponding table. For example, one tuple (obj1,type.object.type,type.type) is scanned, we save obj1(which is represented as mid) into type table. According to the database design, the name, idstring and id of obj1 are also stored. The idstring is obtained by searching tuples with property type.object.id. Since mid is a string, it is not suitable to set it as primary key. We give an integer id for each entity with mid.

The overall procedure of the first round of database injection is shown in Figrue 1. After this procedure, we can get all the objects and store them into different tables. Each object has a mid and an id allocated by us. The details of each table can be found in 5.

In the second round, we store the type of each entity and expected_type and schema of each property. Since we store all the type with mid, we can store the id of entity and type in table entity_type. Similarly, we store the id of properties and their expected_type and schema in table property_expectedtype and table property_schema. In the third round, we store information of every tuple whose first item is a mid. The id of obj1, property and obj2 are injected into table relation. For entities, we also calculate its rank for further use in queries. The calculation of rank of an object is straightforward: it just counts how many tuples are correlated with the object (only when the object appears in the first item of a tuple). After several tests, this method proves to be efficient and correct.

When performing injection, several methods are used to increase the speed. Since writing into database costs much time, we collect 10000 pieces of insert query and perform them in one

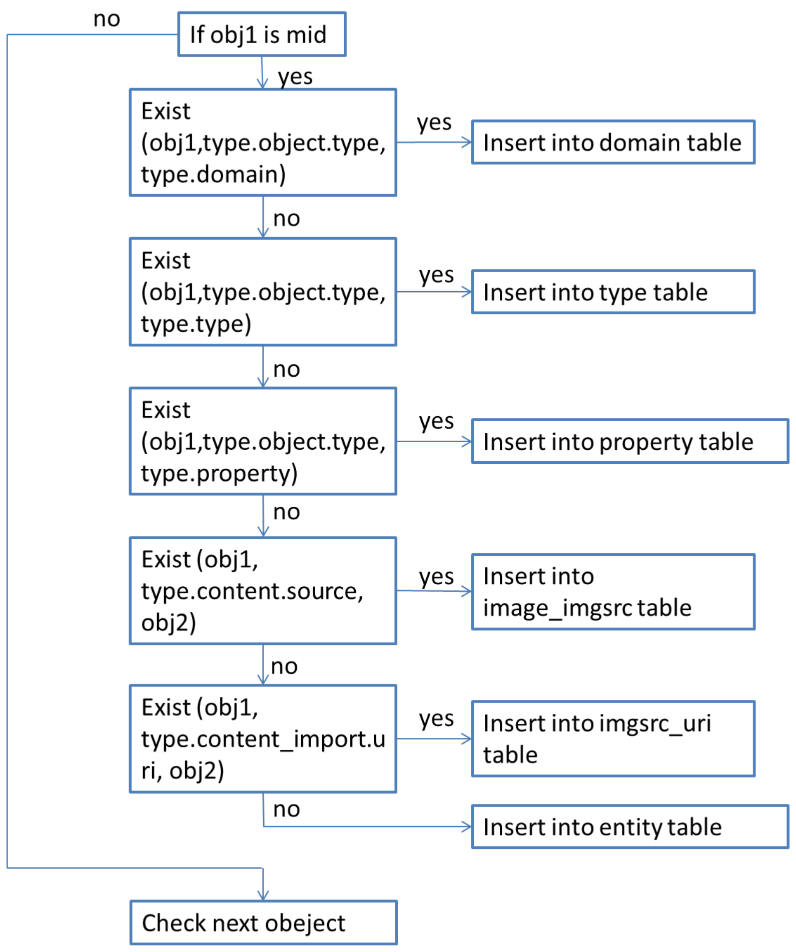

no
If obj1 is mid
yes
Exist (obj1,type.object.type, type.domain)
yes
Insert into domain table
no
Exist (obj1,type.object.type, type.type)
yes
Insert into type table
no
Exist (obj1,type.object.type, type.property)
yes
Insert into property table
no
Exist (obj1, type.content.source, obj2)
yes
Insert into image_imgsrc table
no
Exist (obj1, type.content_import.uri, obj2)
yes
Insert into imgsrc_uri table
no
Insert into entity table
Check next obeject

Figure 1: Procedure of the first round of database injection

run. This method greatly improve the speed of injection. Secondly, when doing injection of table relation, entity_type, property_expectedtype and property_schema, we need to find the id of each object through its mid and idstring. For types and properties, the search is easy since we can read all the tables into memory and find the corresponding id easily. However, the number of entities is too large and we can not read all of them into memory. Therefore, we use a cache to store part of the entities and the size of the cache is 100000. The data structure of the cache is Least Recently Used(LRU) cache. It means that the cache discards the item which is least recently used. We use LinkedHashMap in Java to realize the cache because LinkedHashMap maintains the order of item insertion. The overall hit rate of the cache is about 15% and it improves the speed as well.

4 Query design

In this section, we introduce the queries we design for Freebase database. There are four basic queries:

1. Search a name, return all entities that match the name, better to rank the entities, like Wikipedia. For example, when we search Einstein, the famous physicist Albert Einstein may

rank first.
2. Given an entity ID, return all types it belongs to.
3. Given an entity ID, return all properties whose schema/expected_type is one type of the entity.
4. Given an entity ID, return all objects that are co-occurred with this entity in one triple.

We also add some meaningful queries in our database system:

1. Search a name, return all ranked entities that match the name and also return url of image and description (if exists) of each entity.
2. Given an entity ID, return all objects that are co-occurred with this entity in one triple and also the properties which connects this entity and correlated objects.

The idea of our additional queries is inspired by the basic 4 queries and website of Freebase (www.freebase.com). Once we search a name, we should not only show the basic information of an entity but also its description and image for preview. Query 2,3 and 4 of basic queries are actually strongly related. When we search an entity, we can find many types of it and then we want to find its relations with other entities like what we do when we search in Wikipedia. Remind the definition of type.object.expected_type and type.object.schema, we can understand why these two properties important.
If ∃(prop,expected_type,type), ∀triple t=(obj1,prop,obj2), obj2∈type. Therefore when we search an entity and know its type, the possible entity correlated with it is obj1 with property prop. Similarly, for type.object.type: If ∃(prop,schema,type), ∀triple t=(obj1,prop,obj2), obj1∈type. Therefore when the type of an entity is accessible, the possible entity correlated with it is obj2 with property prop. After realizing the meaning of query 2,3 and 4, it is tuitional to return the properties in triples where an entity occurs which are subset of the result returned by query 3.

Next, we show how we realize our queries in MySQL.

```
#Search entity with its name, results returning with decreasing rank
SELECT * FROM entity
WHERE name = @searchName
ORDER BY rank desc;

#Search image with entityID
SELECT uri
FROM image_uri
WHERE image = @imageID;

#Search entity with its MID
SELECT * FROM entity
WHERE mid = @searchMID;

#Given an entity , search detailed information of its types
SELECT T.id, T.mid, T.name, T. idstring
FROM (SELECT * FROM entity_type
        WHERE entity = @entityID) AS E
        JOIN type AS T ON E.type = T.id;

#Given a type, search properties with regard to expected_type & schema
SELECT P.id, P.mid, P.name, P. idstring
FROM (SELECT * FROM property_expectedtype
        WHERE expected_type = @typeID) AS E
        JOIN property AS P ON P.id = E.property ;
SELECT P.id, P.mid, P.name, P. idstring
FROM (SELECT * FROM property_schema
        WHERE pschema = @typeID) AS E
        JOIN property AS P ON P.id = E.property ;

#Given an entity & a property , search co−occured entity within ternary relation
SELECT E.id, E.mid, E.name
FROM (SELECT obj1
        FROM relation
        WHERE obj2 = @entityID AND property = @propertyID) AS O
        JOIN entity AS E ON E.id = O.obj1;
SELECT E.id, E.mid, E.name
FROM (SELECT obj2
        FROM relation
        WHERE obj1 = @entityID AND property = @propertyID) AS O
        JOIN entity AS E ON E.id = O.obj2;
```

5 Database Design and Optimization

In this section, we introduce the design of our database. The design of our database is strongly related to the query design and the optimization of our database is based on the query design as well. The schema of the database is:

entity(eid,mid,name,description,image,rank)
domain(did,mid,name,idstring)
type(tid,mid,name,idstring)
property(pid,mid,name,idstring)
image(image_id,mid)
imgsrc(imgsrc_id,mid)
uri(uri_id,url)
entity_image(ei_id,eid,image_id)
image_imgsrc(imgsrc_id,image_id,imgsrc_id)
imgsrc_uri(srcuri_id,imgsrc,uri)
entity_type(et_id,eid,tid)
property_expectedtype(pe_id,pid,tid)
property_schema(ps_id,pid,tid)
relation(relation_id,e1_id,pid,e2_id)

To simplify the process of finding images of entities, we just store one image of each entity. We also perform natural join on entity_image, image_imgsrc and imgsrc_uri for convenient. The overall ER diagram of our model is shown in Figure 2.

To accelerate the research on our database and improve the structure of tables, we build some indexes on our tables. We set a series of experiment to examine the performance and necessity of these indexes. In Table 1, we can see that after establishing an index on name in entity table, which contains about 100 million records, the performance of searching improves a lot. From Table 2, we know that index on mid can also accelerate search on table entity. In fact, mid is an integer in 32-system, so we get an idea that we can accelerate the search by an index on integer. When an index is established on a column of int, the proficiency is more

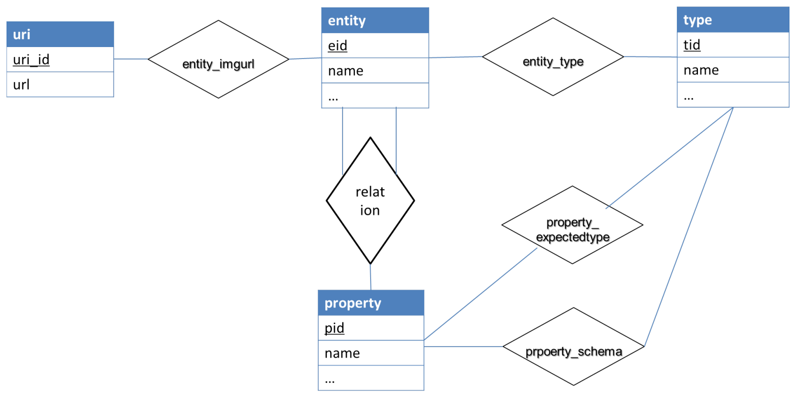


Figure 2: ER diagram of the model

Name	China	Shanghai	Einstein	Average
With Index(name)/s	0.73	0.15	0.63	0.50
Without Index/s	58.23	56.69	57.09	57.34
Return rows	539	187	108	278
Proficiency Improved	98.3%	99.7%	98.9%	99.0%

Table 1: Search entity by name

obvious. As Table 3 shows, there is a huge improvement after an index is established, from about 60 seconds to less than 0.5 second. Moreover, we tried to experiment on some other tables to examine the effect of index. We set various index on table type and we found that, as shown in Table 4, there is also huge improvement when set index on id in this table. All in all, these indexes accelerate our search and do not cost a lot in storage. We carefully compare various indexes (size, performance, significance and priority) and choose these ones to store in our database.

6 GUI Design

We design a website based on ASP.NET with MVC structure to show the performance of our database. With a website, we can better manage the interaction between users and database, and design proper connection methods between them. Though security is not a main concern in this program, we convert data to JSON format when sending data, so data is secure in our system.

As shown in Figure 3, we have a search page and user can choose to search by name or mid of an entity. When user choose to search by entity's name, he or she will get results like Figure 3. The page of website is dynamically loaded so it does not open a new page after researching.

MID	010006m6	010050rb	0100f34	Average
With Index(mid)/s	0.50	0.01	0.01	0.17
Without Index/s	64.81	64.82	65.06	64.90
Return rows	1	1	1	1
Proficiency Improved	99.2%	99.9%	99.9%	99.7%

Table 2: Search entity by mid

ID	1	100	10000	Average
With Index(id)/s	0.00	0.00	0.00	0.00
Without Index/s	86.85	91.74	92.06	90.25
Return rows	1	1	1	1
Proficiency Improved	100%	100%	100%	100%

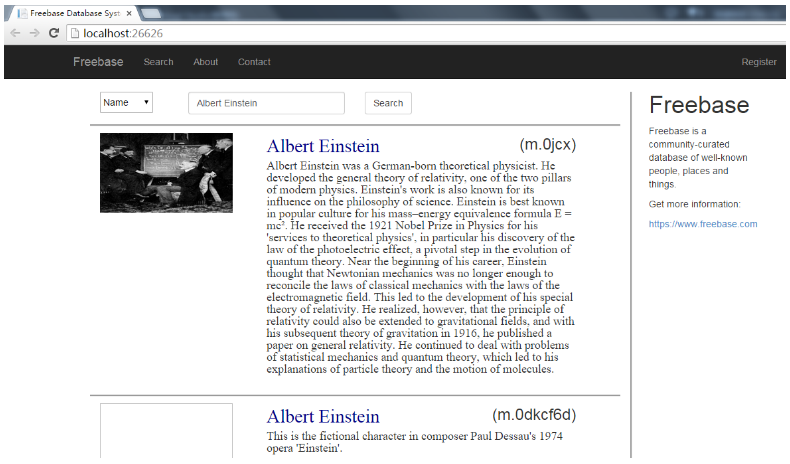
Table 3: Search entity by id

	Red Altert	Over you	Bitter Sweet	Average
With Index(type.id)/s	0.56	0.01	0.24	0.27
Without Index/s	71.81	72.82	73.32	72.65
Return rows	5	2	5	4
Proficiency Improved	99.2%	99.9%	99.7%	99.6%

Table 4: Search type by id

The example in Figure 3 is 'Albert Einstein'. We can see that results are listed by rank and the famous scientist Albert Einstein is in the first place. Together with his name, we also show his picture, mid, description in this page. We have to note that freebase only contains limited pictures and many directories of pictures are invalid, so most of entities show no pictures in our website.

If you search an ID in the search box and choose ID in the select box, or just click on an entity, you can get access to Figure 4. We gather most functions of our database in this search, except the search in last paragraph. The top of the page shows basic information of this entity, similar as searching by name. After that, all types of this entity are shown with their name string and mid. For every type, this entity can have connect with other entities by several properties. We list all these properties after the specific type and note that whether it is an expected type or a schema. Furthermore, if the entity is co-occurred with another entity through this property, we show all these entities under the property. Certainly, detailed information of properties are also shown. It is not easy to show all these information in one page, and we try our best to make this page seeming simpler. Taking Figure 5 as an example, Albert Einstein can be a innovator, an astronomer or an award winner. For innovator type, he is a creator (this is an expected type) and he created general relativity. In Figure 6, we show another example, Rafael Nadal.



Albert Einstein (m.0jcx)

Albert Einstein was a German-born theoretical physicist. He developed the general theory of relativity, one of the two pillars of modern physics. Einstein's work is also known for its influence on the philosophy of science. Einstein is best known in popular culture for his mass–energy equivalence formula E = mc². He received the 1921 Nobel Prize in Physics for his 'services to theoretical physics', in particular his discovery of the law of the photoelectric effect, a pivotal step in the evolution of quantum theory. Near the beginning of his career, Einstein thought that Newtonian mechanics was no longer enough to reconcile the laws of classical mechanics with the laws of the electromagnetic field. This led to the development of his special theory of relativity. He realized, however, that the principle of relativity could also be extended to gravitational fields, and with his subsequent theory of gravitation in 1916, he published a paper on general relativity. He continued to deal with problems of statistical mechanics and quantum theory, which led to his explanations of particle theory and the motion of molecules.

Albert Einstein (m.0dkcf6d)

This is the fictional character in composer Paul Dessau's 1974 opera 'Einstein'.

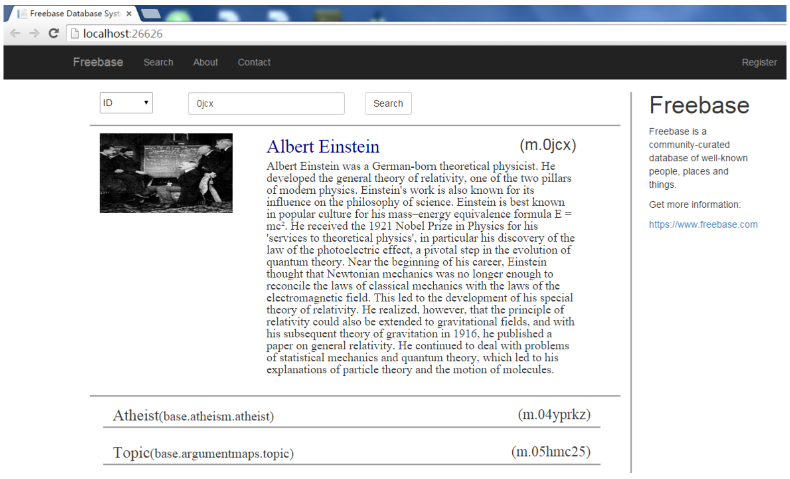
Freebase

Freebase is a community-curated database of well-known people, places and things.

Get more information:

https://www.freebase.com

Figure 3: Search 'Albert Einstein as a name



Albert Einstein (m.0jcx)

Albert Einstein was a German-born theoretical physicist. He developed the general theory of relativity, one of the two pillars of modern physics. Einstein's work is also known for its influence on the philosophy of science. Einstein is best known in popular culture for his mass–energy equivalence formula E = mc². He received the 1921 Nobel Prize in Physics for his 'services to theoretical physics', in particular his discovery of the law of the photoelectric effect, a pivotal step in the evolution of quantum theory. Near the beginning of his career, Einstein thought that Newtonian mechanics was no longer enough to reconcile the laws of classical mechanics with the laws of the electromagnetic field. This led to the development of his special theory of relativity. He realized, however, that the principle of relativity could also be extended to gravitational fields, and with his subsequent theory of gravitation in 1916, he published a paper on general relativity. He continued to deal with problems of statistical mechanics and quantum theory, which led to his explanations of particle theory and the motion of molecules.

Atheist(base.atheism.atheist) (m.04yprkz)

Topic(base.argumentmaps.topic) (m.05hmc25)

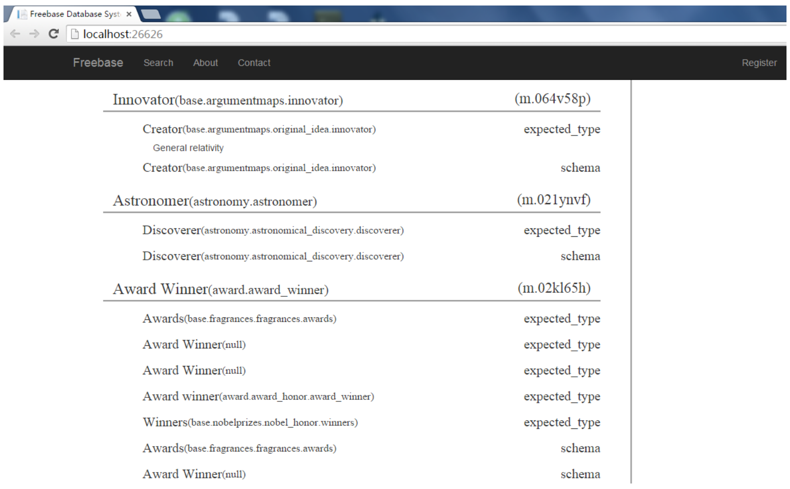
Freebase

Freebase is a community-curated database of well-known people, places and things.

Get more information:

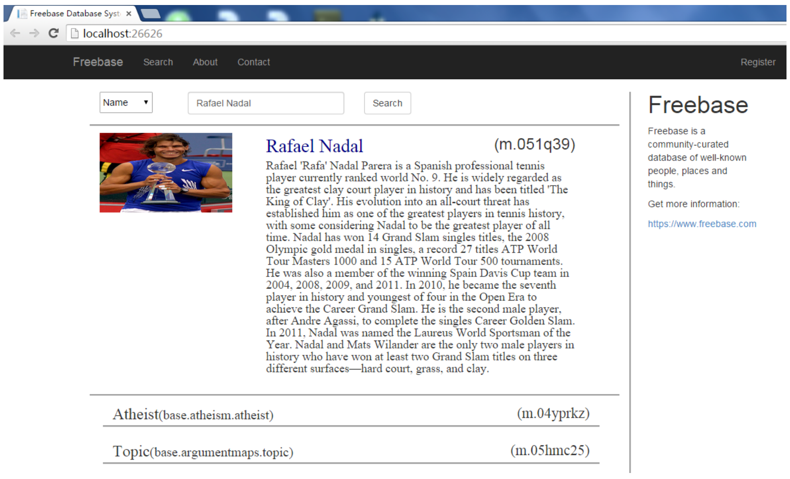
https://www.freebase.com

Figure 4: Search '0jcx' as a mid



Innovator(base.argumentmaps.innovator)	(m.064v58p)
Creator(base.argumentmaps.original_idea.innovator)	expected_type
General relativity	
Creator(base.argumentmaps.original_idea.innovator)	schema
Astronomer(astronomy.astronomer)	(m.021ynvf)
Discoverer(astronomy.astronomical_discovery.discoverer)	expected_type
Discoverer(astronomy.astronomical_discovery.discoverer)	schema
Award Winner(award.award_winner)	(m.02kl65h)
Awards(base.fragrances.fragrances.awards)	expected_type
Award Winner(null)	expected_type
Award Winner(null)	expected_type
Award winner(award.award_honor.award_winner)	expected_type
Winners(base.nobelprizes.nobel_honor.winners)	expected_type
Awards(base.fragrances.fragrances.awards)	schema
Award Winner(null)	schema

Figure 5: More information when searching mid



Rafael Nadal (m.051q39)

Rafael 'Rafa' Nadal Parera is a Spanish professional tennis player currently ranked world No. 9. He is widely regarded as the greatest clay court player in history and has been titled 'The King of Clay'. His evolution into an all-court threat has established him as one of the greatest players in tennis history, with some considering Nadal to be the greatest player of all time. Nadal has won 14 Grand Slam singles titles, the 2008 Olympic gold medal in singles, a record 27 titles ATP World Tour Masters 1000 and 15 ATP World Tour 500 tournaments. He was also a member of the winning Spain Davis Cup team in 2004, 2008, 2009, and 2011. In 2010, he became the seventh player in history and youngest of four in the Open Era to achieve the Career Grand Slam. He is the second male player, after Andre Agassi, to complete the singles Career Golden Slam. In 2011, Nadal was named the Laureus World Sportsman of the Year. Nadal and Mats Wilander are the only two male players in history who have won at least two Grand Slam titles on three different surfaces—hard court, grass, and clay.

Atheist(base.atheism.atheist) (m.04yprkz)

Topic(base.argumentmaps.topic) (m.05hmc25)

Freebase

Freebase is a community-curated database of well-known people, places and things.

Get more information:

https://www.freebase.com

Figure 6: Another example

7 Conclusion

We reviewed the knowledge learned in database course and get access to a lot of new technology through this experiment. We practiced how to design a database, how to extract information for real environment and import them into database, how to improve the performance of our database and how to query and manage in the database. With this experiment, knowledge of SQL is no longer some structures or some searching sentences on the textbook, but a useful tool for us. As computer science major students, database is a basic tool for us in further study or work, so this experiment bring us a valuable chance to improve our skills.

Things are never in a plain sailing and we met a lot of difficulties in this experiment. At first, the huge size of database of freebase really surprised us: 100 GB in .GZ file and more than 300 GB after unzip. From the website of freebase, we know that it contains more than 3 billion rows, so the design of database is a main concern in this experiment. Our first version of database stored all triples in one table, which reduced the difficulty of process data from origin file, did not return satisfying searching speed. Thus, we apply for some extra time and rebuild our database – this is costly in both time and energy for us, but it turned out to be a right choice.

After that, the choice of GUI also troubled us. To get a better user experience and express our results better, we choose to build a website. Our database is running on a linux system so it is not easy to run our C# program on it, and it is time costing to run website on other computers – most time is spent on connecting to database instead of do queries. All of our queries cost less than 1 second but it may cost up to 10 second to connect to the database. Thus, we do a lot of effort and setup a virtual machine in the linux system and run our website on it. It returns results fast when we do queries.

All in all, we harvest a lot through this experiment. Thanks to the help of Kenny Zhu, TA and some kind classmates, we learn a lot in this course. And this is an unforgettable experience for us.

8 Distribution

Xiaowei Liang: Design Database, Build Database
Zhuodi Liu: Process Origin Data, Implement Query, Website