

A Structured Search Engine for Deep Web Databases

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Amal Aljohani

Approved by:

Major Professor:

Hasan Jamil, Ph.D.

Committee Members:

Clinton Jeffery, Ph.D.

Xiaogang Ma ,Ph.D.

Jia Song, Ph.D.

Department Administrator:

Terence Soule, Ph.D.

May 2022

ABSTRACT

Throughout the years, researchers have striven to improve the quality of information retrieval from the web; especially for ordinary users who need the proper access to the desired information. Users mostly interact with the web through search engines and they tend to expect precisely what they asked for. As web databases (Deep Web) hold enormous amounts of high quality information that users need to access and leverage, we shed some light on the importance of *searching* the deep web rather than just *querying* it. Searching is more flexible than querying based on fixed variables. Searching the deep web can enhance information accessibility especially if it simulates user behaviour. Research interest is growing in the area of maximizing the usefulness of web search via utilizing the largest portion of the web (Deep Web). Search engines such as Google can only find indexed information that is present in the *Shallow Web*. In contrast, peeking into *Deep Web* databases is not possible for search engines such as Google. Search engines are not able to simulate SQL-like queries of database contents in a traditional or other intuitive human-like manner. In this work, we present a system called *DeepQ* to search the database contents behind the firewalls inside the deep web and show that these contents can be accessed using a structured query language treating them as a deep relational web. We leverage a recent proposed declarative deep web query language called DQL, and we present the contours of its implementation in the *DeepQ* system. We also believe that this work has the potential to demonstrate the Universal Relations model, as the user will be able to interact with databases that are hidden behind firewalls, and also search conveniently for information.

ACKNOWLEDGEMENTS

It is amazing to look back and realize how things have changed over the past few years, how others have influenced me. This Ph.D. dissertation would not be possible without the generous guidance, help, and support from many brilliant people. First and foremost, I want to thank Allah, for the completion of this dissertation. I would like to express my sincere gratitude to my advisor Dr. Hasan Jamil for his guidance, support, and patience. I would not have made it this far if he had not believed in me. He was supportive and kind. He let me be creative, kept me on track, and taught me that imagination is the key of innovation. I am forever thankful to him.

I would also like to thank my committee members, Dr. Clinton Jeffery, Dr. Xiaogang (Marshall) Ma, and Dr. Jia Song for their valuable insights on my dissertation work and for their time and support. I would like to thank all my instructors for their dedication in providing me with valuable education. I would like to thank the department chair Dr. Terry Soule and all staff in the department of Computer Science for their help during my study. Also, I would like to thank the president of U of I Mr. Scott Green for his calming letters during the pandemic.

Moreover, I am thankful for the government of Saudi Arabia, the Saudi Arabian Cultural Mission and Majmaah University for funding my scholarship. I would like to thank my friends: Rawan Almakinah, Amani Alfulaiti, Emtenan Khoj, Yara Alshmrani, Umar Siddiqui, Ali Alelaiwi, Saif Alharthi for their support and patience. Also, I would like to thank my friends in U of I who supported me in this journey: Joel Oduro-Afriyie, Amruta Kale, Zeinab Ghafari, Nuzhat Yamin, Azadeh Nikoukar.

Last, but certainly not least, I would like to thank my mom, dad, brothers and sisters for their understanding, support, encouragement, patience, and love which helped me make this dissertation a reality.

DEDICATION

This work is dedicated to praising Allah and to the people I lost during my Ph.D.: my grandma who passed a way few months ago, and my cousin Fatima. May God bless their souls. It is also dedicated to my father, Nasser Aljohani, my mother Aisha Aljohani, my other grandma, my sisters Ekhlal, Elham, Ahlam and my brothers, Omar and Anas. Finally, it is dedicated to myself that managed to work on Ph.D while living with (mostly fighting) Multiple Sclerosis.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
DEDICATION.	iv
TABLE OF CONTENTS.	v
LIST OF TABLES	vi
LIST OF FIGURES.	vii
1 INTRODUCTION	1
1.1 Between Shallow Web and Deep Web	2
1.2 Motivation and Objectives	4
1.3 Hypotheses	4
1.4 Work Contribution	5
1.5 Research Problem	5
1.6 Author's Related Publication	9
1.7 Dissertation Organization	11
2 BACKGROUND AND RELATED WORK	12
2.1 Web Database Querying and Web Searching	12
2.2 Why do we need to search the Deep Web?	13
2.3 Why is it difficult to search the Deep Web?	15
2.4 Toward Searching and Accessing the Deep Web	15
3 DEEPQ SYSTEM	28
3.1 Leveraging the DQL features	28
3.2 DeepQ Architecture	30
3.3 System Overview	33
4 IMPLEMENTATION AND EXPERIMENTAL RESULTS.	39
4.1 Syntax Checker	39
4.2 Single Domain web databases(Simple Query)- An Example	39
4.3 Multiple Domains web databases(Complex Query)- An Example	41
4.4 Performance and Evaluation Criteria	44
5 CONCLUSION AND FUTURE DIRECTION	47
5.1 Conclusion	47
5.2 Future Direction	48
BIBLIOGRAPHY.	50

LIST OF TABLES

TABLE 2.1	A Comparison between Physical and Virtual Integration Systems . . .	16
TABLE 2.2	Some Systems Strengths and Limitations.	24
TABLE 2.3	Comparison of System Capabilities, Identification (Source Identification), Classification, Integration, UUQI(Unified User Query Interface), SM (Schema Matching), FF (Form Filling), Results Wrapping, Reusability. 25	
TABLE 3.1	Input/Output Attributes for each form.	35
TABLE 4.1	DeepQ Evaluation (UUQI: Unified User Query Interface)	46

LIST OF FIGURES

FIGURE 1.1	A User Interaction With the Deep Web.	3
FIGURE 1.2	Google response to query Q_1	6
FIGURE 1.3	Expansion of the top link for Q_1 showing a partial list of available Honda Civics.	6
FIGURE 1.4	Manual query results of Q_1 at autotrader.com.	7
FIGURE 1.5	Google response to query Q_2	8
FIGURE 1.6	Expansion of a link for Q_2 showing a partial list of available Honda Civics.	8
FIGURE 1.7	One of the results that meet Q_2 (From cars domain).	9
FIGURE 1.8	Results from the Elevation Map.	9
FIGURE 1.9	The manual process to get useful results for Q_2	10
FIGURE 2.1	Databases and Search Engines	14
FIGURE 2.2	Some Selected Previous Work Throughout the years.	26
FIGURE 2.3	An overview toward deep web searching.	27
FIGURE 3.1	DeepQ system	28
FIGURE 3.2	DeepQ architecture.	30
FIGURE 3.3	autotrader.com landing page pre-filter – pick-and-filter querying.	32
FIGURE 3.4	carvana.com landing page filter – direct querying.	32
FIGURE 3.5	DeepQ System Overview for a single domain (Simple Query).	34
FIGURE 3.6	Form 4 after submitting as an example	35
FIGURE 3.7	Form 2 after submitting as an example	35
FIGURE 3.8	Graphical Interface (restricted)	38
FIGURE 3.9	Text Interface	38
FIGURE 4.1	The logic in the Syntax Checker.	40
FIGURE 4.2	Results for Q_3	41
FIGURE 4.3	A small modification in Q_3	41
FIGURE 4.4	Results for Q_4	42
FIGURE 4.5	DeepQ System Overview for the multiple domain querying.	42
FIGURE 4.6	Linking two different deep web sites that have different domains	43
FIGURE 4.7	Results from one of the Elevation Databases	44
FIGURE 4.8	Results for Q_5	45
FIGURE 4.9	a complex Query such as Q_4	45
FIGURE 4.10	DeepQ performance measures.	46

CHAPTER 1

INTRODUCTION

Today, the World Wide Web (also referred to as the web) is the leading infrastructure for information presenting and publishing and it has become the most important platform for e-commerce and business transactions. The number of web pages is growing exponentially, and this massive growth has caused the web to evolve into a data-rich repository (Kabisch, 2011). The web holds millions of searchable data sources, and a large portion of them are hidden behind firewalls and can only be accessed through Query Interfaces (or Web Forms). These query interfaces are the main means to retrieve web database contents; they also provide a glance into the underlying database structure. Users are allowed to submit queries through a query interface to the web database and obtain results as dynamically generated responses from those databases.

The Web contains structured, semi structured, and unstructured content. The structured content that resides in the web databases are called the *Deep Web* or *Hidden Web* (Dragut, 2012); thus, the deep web contains structured data that exists in dynamic generated web pages (Cali and Straccia, 2017). It is also important to note that obtaining the content of web databases requires that users pose structured queries through Web forms. Web databases are usually domain specific, and it is impossible for one web database to carry all information about that specific domain. Thus, users usually search in different web databases, even if they are trying to get one result from a specific domain. It is not difficult to notice the underlying limitations of marrying shallow web searching to deep web querying (Aljohani, 2021a). We believe that ordinary users should be able to pose simple declarative queries that can obtain useful, accurate, and relevant results whether the domain is specific or not. Also, users should be able to search the deep web not just query it.

1.1 BETWEEN SHALLOW WEB AND DEEP WEB

Web sites can be classified based on the content of their pages to *structured* and *unstructured* pages. The unstructured pages contain static HTML pages and the structured pages contain the web database driven contents. It is useful to mention that some predefined *design templates* are usually used to create the structured pages that are generated from the content of web databases (Kabisch, 2011). The term *Shallow Web* (also known as *Surface Web*) refers to the web pages that are indexed by some search engines such as Google. In fact, the unstructured portion of the web is mainly the one that is indexed by search engines. In contrast, the term *Deep Web* (also known as *Hidden Web*) has been used in the past by many authors to refer to non-indexed web pages. There is also the *Dark Web* that should not be confused with the Deep Web as it refers to some web pages that use encryption software such as **Tor** to hide the location and IP addresses (McCoy *et al.*, 2008). Web contents can be categorized into two types: *static*, and *dynamic* based on their impact on crawlers. If some or all content of a web page is generated at run time, it is defined as a dynamic web page. More than 80% of web content is dynamically generated (Raghavan and Garcia-Molina, 2000). In contrast, the web contents that exist on the server and are transmitted after receiving a client request are defined as static web pages. The web in general is dynamically growing, and the size of the indexable web is huge and rapidly increasing. However, the size of the hidden web is much larger (Lawrence and Giles, 1998), and in 2012 this size reached several trillions of web pages (Hernández *et al.*, 2018).

1.1.1 *The Deep Web*

The focus in this work is on the traditional definition of the term “Deep Web” that was proposed in 2001 by Bergman (Bergman, 2001). Deep web refers to the dynamically generated pages that come in response to submitting a query via web forms. It contains a tremendous amount of useful and high-quality information that can only be obtained after submitting a query via web form. In fact, the dynamic query-based data access is provided through query interfaces that provide access to lots of online databases. In other words, the dynamically generated pages that have the structured

data come as responses to a specific query. Deep web content can neither be seen nor retrieved by traditional search engines; it is estimated that about ninety-five percent of deep web sources are accessible publicly for free behind their respective firewalls (Bergman, 2001).

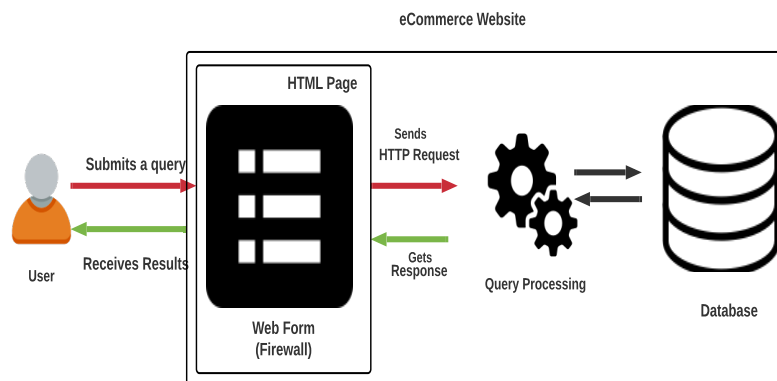


FIGURE 1.1: A User Interaction With the Deep Web.

1.1.2 Deep Web Concepts

There are some important concepts of the deep web such as: Deep Web Source, Query Interface, Result Interface, and Query Interface Elements. These concepts are key elements in making the deep web accessible for users.

DEFINITIONS —

- *Deep Web Source*: A deep web source is a web site that generates its content depending on a database and it can not be indexed by search engines. The deep web source has one or many *fill-out forms* or *query interfaces* that users can fill in order to obtain results through a *result interface*. These two interfaces play an important role in revealing the database schema and they referred to as *interface schema*.
- *Query Interface Elements* A query interface has visible elements (as HTML fields) that represent a specific domain such as *car vendors*.

Figure 1.1 shows the interaction between the user and the deep web. Users submit queries to a *web form* (form interface) that serve as an entry point to the web database. This form returns results as a response to the submitted query. Web forms are also called searchable forms; the non-searchable forms are the ones that only accept input from the user without searching the database behind it or returning results such as signing up forms (El-Gamil *et al.*, 2011).

1.2 MOTIVATION AND OBJECTIVES

This work focuses on proving that DQL is feasible and that it facilitates a programmatic access to the Deep Web. It is able to facilitate result integration from multiple sources to obtain results that are relevant to a user. Our objectives are:

- To propose an implementation of the DQL query language (Jamil and Jagadish, 2015) for structured querying of deep web databases, and to demonstrate that the DeepQ system holds promise for eCommerce applications.
- To show that the UR view can be beneficial for simplifying the view of the deep web for users (Jamil, 2021).

Our goal is to make the deep web data accessible, searchable, and useful. The main objective is to show that the proposed structured query model is feasible, and that it has the potential to facilitate deep web access, enable leveraging of deep web data, and provide only the desired and useful results for the user (Jamil and Jagadish, 2015). In fact, DeepQ is built based on DQL, thus it has the potential for providing a reusable framework in a way that the research community can use to advance the field jointly.

1.3 HYPOTHESES

- An abstract view of the deep web as a schemaless universal relation is feasible and it enables the design of an SQL-like declarative query language called DQL.
- A translational semantics of DQL can be used as an implementation strategy to build a graphical user interface called DeepQ.

- A restricted subset of DQL has a faithful and intended mapping into DeepQ.

1.4 WORK CONTRIBUTION

Our contribution can be summarized as follows:

1. Presenting DeepQ as an implementation of DQL for structured querying of database contents behind firewalls.
2. Implementing simple and complex query handling.
3. Demonstrating the UR model within the context of the deep web.

1.5 RESEARCH PROBLEM

Search engines such as Google do not allow users to peek into the databases behind web forms. These engines can only access information that resides in the Shallow Web.

Consider for example that Athena is searching online for a car. She submits the following query to Google:

Q₁: Red Honda Civic ex 2015 less than \$5,000 Moscow Idaho

This query returns astonishing an 67,800 results, but none of them precisely responded to the query as shown in Figure 1.2. Google's response was either completely wrong and did not meet query conditions, or partially met some of the query conditions. After clicking on the topmost recommended link as shown in Figure 1.3, we can see that none of the cars fully satisfy the query conditions. Most responses ignored the location condition, and the other responses are actually commercial leads for businesses. These leads might be generated using text to approximate query translation APIs to find a possible match in a database that is close enough (Bianchi *et al.*, 2021) and not truly responsive to the user's intended query. User intents are very difficult to model properly (Elhabbak *et al.*, 2020). In fact, search engines are unable to query deep web databases because they are unable to simulate a search of

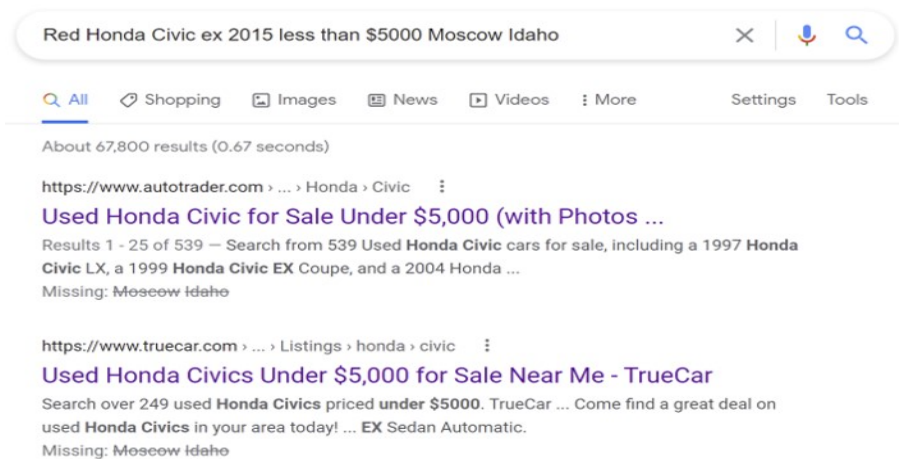


FIGURE 1.2: Google response to query Q_1 .

database contents like a human can do. Accessing deep web sources can only be done through search interfaces that hide those contents.

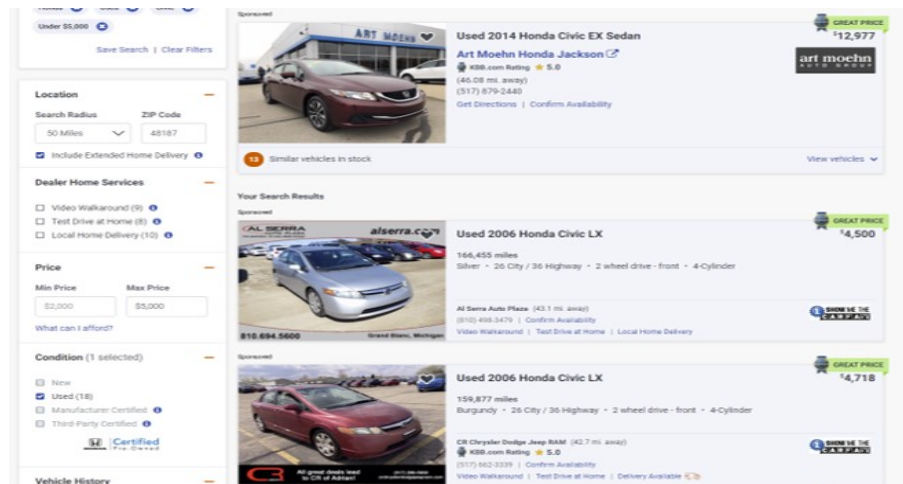


FIGURE 1.3: Expansion of the top link for Q_1 showing a partial list of available Honda Civics.

Google's topmost recommendation as shown in Figure 1.2 was a car vendor called autotrader.com. The banner reads "Used Honda Civic for Sale Under \$5,000" and purports to have about 539 rows (cars) of information. After clicking on that link, it shows the list of cars in Figure 1.3. As shown in that figure, none of the cars satisfy the query conditions. If the user manually uses the search interface and the filter functions to look for a car that has the closest specification, she will end up with a list

of results as shown in Figure 1.4, which is actually close to what she would like to see.

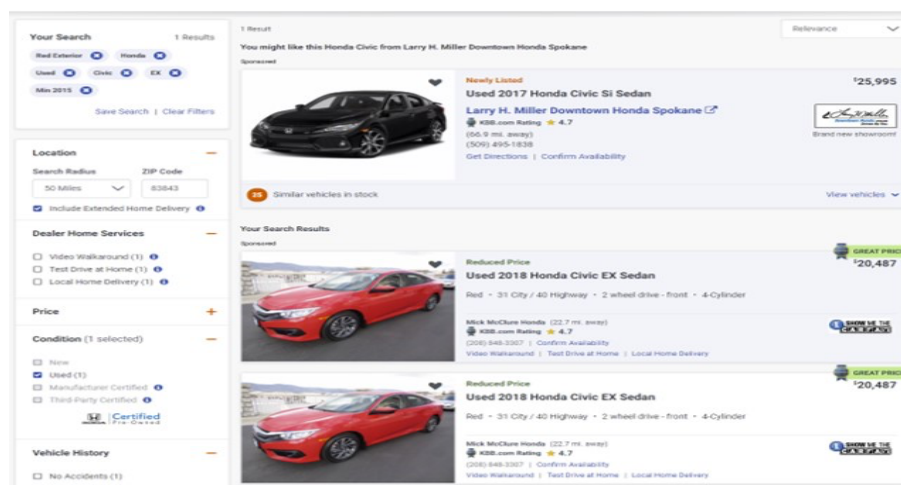


FIGURE 1.4: Manual query results of Q_1 at autotrader.com.

Queries are not always simple and union compatible, some queries are complex and need further construction of universal relation from deep web interfaces. Consider for example that Anne is searching online for a car; she is a 35 year old who has severe *Asthma* condition since she was a child. Because of her condition and the potential of environmental factors that might trigger an attack, she puts her condition at the top front of her mind whenever she plans to go anywhere. Living with asthma has taught her to consider environmental elements such as elevations, climate, and pollution. She always tries to avoid locations with high altitude since it is one of the environmental triggers that impact people living with Asthma (Seys *et al.*, 2013). Anne is trying to find a car that she can buy at proper location in Colorado ¹ considering her condition, so she submits the following query to Google:

Q_2 : Find Red Honda Civic 2015 or newer model less than \$20,000 in cities at elevation lower than 4000 feet

This query returns an astonishing 342,000 results, but as shown in Figure 1.5, results are not accurate since they do not respect the elevation condition and it was

¹Even though Colorado is a state with many high-elevation locations, there are some cities with lower elevation

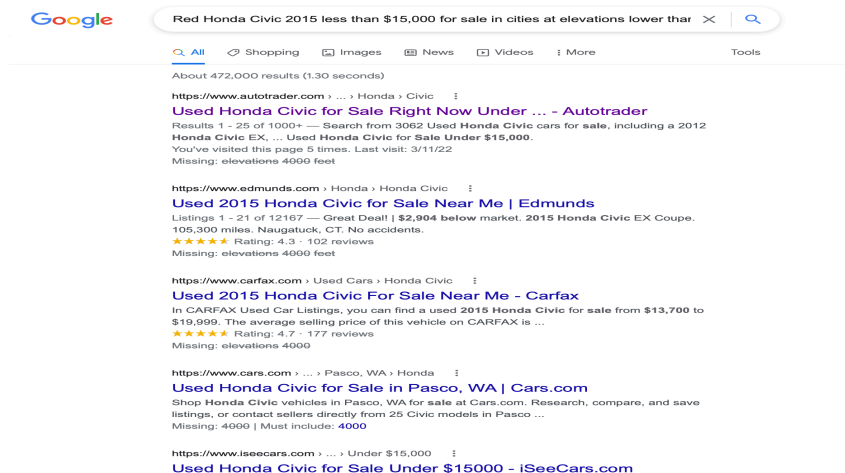


FIGURE 1.5: Google response to query Q_2 .

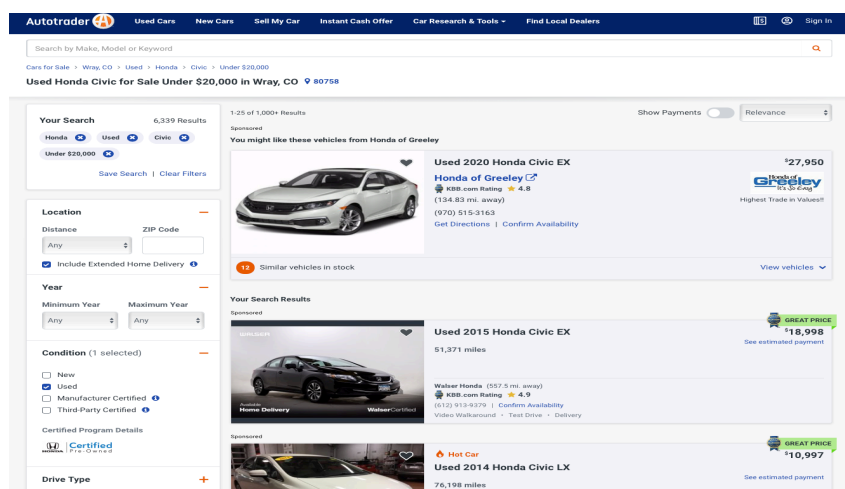


FIGURE 1.6: Expansion of a link for Q_2 showing a partial list of available Honda Civics

completely ignored (*Shallow Web Searching*). After clicking on Google's top recommended vendors (autotrader.com²), she still cannot find the car that meets the price and color, or year conditions (in Figure 1.6).

At this point, the user is left with one choice which is to fill the car vendor search interface manually in order to find the desired car (*Deep Web Querying*). Keep in mind that after all of this work, she has not even begun to process the second critical part of the query: (the elevation) condition. She finally finds a red Honda Civic in Colorado Spring as shown in Figure 1.7. Now Anne needs to dig the elevation information

²<https://autotrader.com/>

Autotrader Used Cars New Cars Sell My Car Instant Cash Offer Car Research & Tools Find Local Dealers

Used 2016 Honda Civic LX

\$16,910 See the KBB Fair Market Range

Explore Financing Capital

Alpine Buick GMC South Inc.
KBB.com Rating: 4.8 (2055)
(719) 296-2119

Delivery Contact for shipping options.
1313 Mosier City Dr, Colorado Springs, CO 80906

Chat with Dealer

Message
I'm interested in your Used 2016 Honda Civic LX listed for \$16,910.

First Name Last Name
Email Phone (Optional)

Send Email

By using this service, you accept the terms of our Visitor Agreement
Yes, I would like to receive price drop alerts on this vehicle and helpful shopping information from Autotrader & its affiliates.

95,100 miles Continuously Variable Automatic Transmission
Balbye Red Exterior 2 wheel drive - front
Black/Gray Interior 30 City / 40 Highway

FIGURE 1.7: One of the results that meet Q_2 (From cars domain)

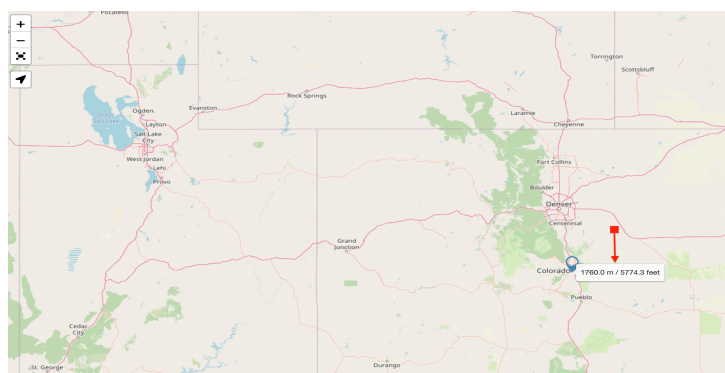


FIGURE 1.8: Results from the Elevation Map

elsewhere and if the elevation condition is not satisfied, then she will query other deep web sites for more cars in different locations. Anne found another deep web site that has an elevation database (such as Elevation Finder ³), see Figure 1.8 and she continues querying and searching. This process is exhausting and time consuming if it is done manually as shown in Figure 1.9.

1.6 AUTHOR'S RELATED PUBLICATION

1. Amal Aljohani. 2021. DeepQ: A System to Peek Inside the eCommerce Deep Web. In The 20th International Conference on Information Knowledge Engineering (Las Vegas, NV).

³<https://www.freemaptools.com/elevation-finder.htm>

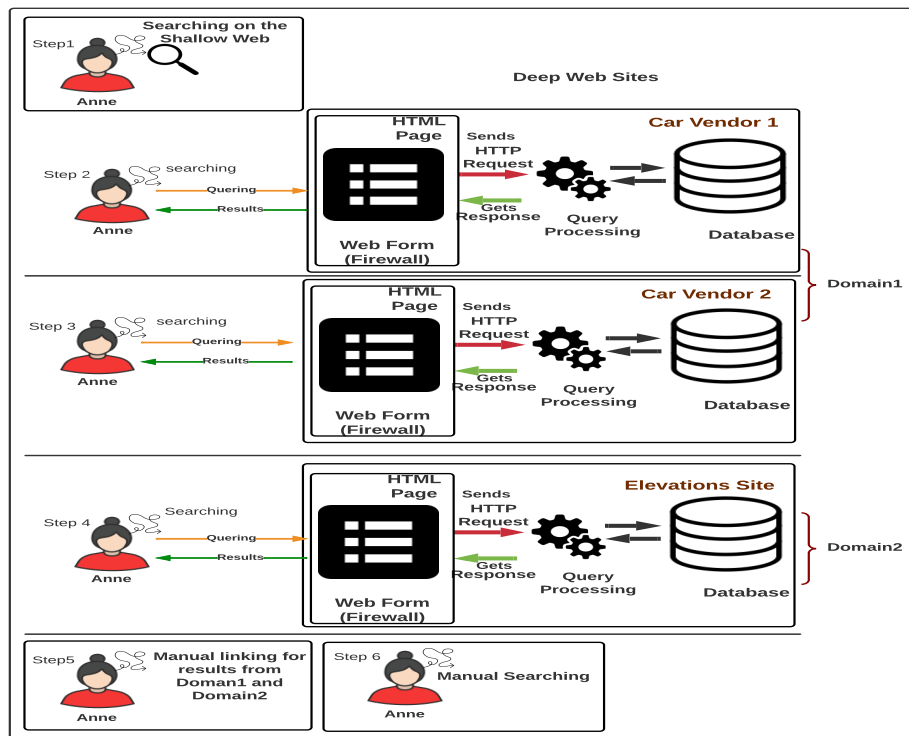


FIGURE 1.9: The manual process to get useful results for Q_2 .

2. Amal Aljohani. 2021. Personalized Question Answering On the Web using an Ontology. In The 7th International Conference on Health Informatics Medical Systems (Las Vegas, NV).
3. Amal Aljohani. A Smart User Interface for Structured Deep Web Search (Submitted).

1.7 DISSERTATION ORGANIZATION

The remainder of this dissertation is organized as follows: Chapter 2 discusses the research background; Chapter 3 presents the DeepQ system; Chapter 4 presents the implementation and experimental results. Chapter 5 includes the conclusion and suggestions for future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 WEB DATABASE QUERYING AND WEB SEARCHING

Database technology has made such remarkable advancement in the past decades that we can now process complex queries on huge data sets. Retrieving information from databases requires knowledge of the database schema in order to be able to form structured queries. The beauty of database querying is that it restricts the results to deliver the exact desired information. In the past, the capability of querying the web (the hidden web) using database techniques was studied. But the focus was limited on the logical level such as data models development and query languages (Florescu *et al.*, 1998). Also, the problem of keyword search in traditional relational databases has been studied heavily in the literature (Agrawal *et al.*, 2002),(Hristidis and Papakonstantinou, 2002); this problem gets more complicated in the context of web databases. Instead of databases, search engines have played a big role in connecting users with the web as it allows them to issue simple keyword-based queries to get the desired result. The current mode of searching for an information on the web is by posing *unstructured queries*, where users pose a keyword query and get a list of URLs as a result. Obtaining information via unstructured queries is not a straightforward process, as users usually need to modify the query and review many results before reaching the desired one. Consider the number of results that would be listed to a user who is searching for a specific piece of information. An unthinkable number of results could be provided which are not necessarily relevant to the issued query. Search engines such as Google fail to fully satisfy the information needs of users (Khelghati, 2016),(Aljohani, 2021a). Users are allowed to issue queries and modify them according to their desires; so they tend to issue several queries until they are satisfied with the results (Jagadish *et al.*, 2007). The The previous examples in section 1.5 show the difference between web database querying and shallow web searching. When users submit the keyword queries, they start with searching the

shallow web, and when they find the search form in the deep web site, they query the database behind that interface. After that they search again among the results. The difference between searching and querying can be summarized as follows:

- *Querying*: users know where to look but they do not know the data.
- *Searching*: users know what they want but they do not know where to look.

Researchers have striven to improve the quality of information retrieval from the web; especially for ordinary users who need the proper access to the desired information.

2.2 WHY DO WE NEED TO SEARCH THE DEEP WEB?

2.2.1 *High- Quality Information*

The deep web carries huge amounts of hidden information that would be very beneficial if accessed and leveraged properly. Traditionally, users are restricted to accessing the content that is provided by search engines from the shallow web. The probability of finding the desired information in the deep web is much higher considering its size and richness (Bergman, 2001). Even though hidden information represents a large proportion of the web, finding and accessing such data through search engines is often impossible (Aljohani, 2021a). Usually, deep web crawlers reach web pages and process them via information extractors that provide structure to the information in order to facilitate its integration in an automated manner (Hernández *et al.*, 2018). There are many useful databases on the web, but users find it difficult to find the right sources and query them. It is not very useful to query only one deep web site at a time; it is more efficient to integrate data from multiple sources (Jou, 2016a). Researchers are cognizant of the fact that it is difficult to query the deep web and integrate it because of access limitations (Calì and Straccia, 2017). Many research projects aim to empower users with the ability to access databases on the web effectively by proposing new approaches and building systems that can make the deep web more accessible and usable (He *et al.*, 2005a).

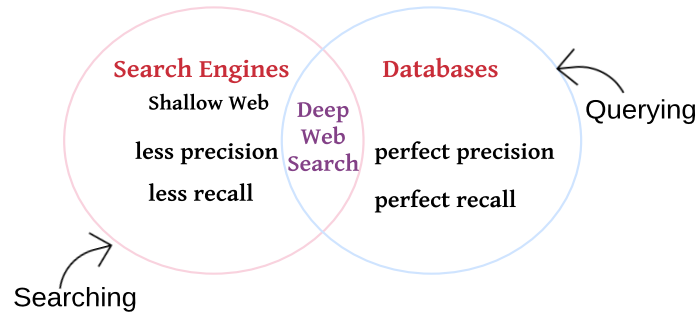


FIGURE 2.1: Databases and Search Engines

2.2.2 Better User Experience

Users encounter frustrating experiences frequently while searching for information on the web. This frustration might be caused by the staggering number of results that might or might not be related to the submitted queries. Even if the results are related to the submitted query, they might need further filtration and examination that would lead to time lost. Researches indicate that user frustration is a real problem and that frustration has an impact not only on the users themselves, but also on how they interact with other people during the day. In fact, one study shows that searching the web is the most frustrating experience for users who participated in that study (Ceaparu *et al.*, 2004). There are important factors that can affect the quality of the users' experience when they search for information on the web. User satisfaction is a very important factor and a key measure for search effectiveness. It is interesting to note that users' expectations are not the same when it comes to dealing with databases and search engines (Jagadish *et al.*, 2007) as shown in Figure 2.1. It has been argued that search engines cannot be built on top of databases, and despite the fact that search engines work properly for the web, they fall short of addressing usability problems that plague database systems (Jagadish *et al.*, 2007). Making the deep web searchable and accessible is desirable for providing more accurate and related results to user queries. This would increase satisfaction, leading to a better search experience.

2.3 WHY IS IT DIFFICULT TO SEARCH THE DEEP WEB?

In general, web search will only continue to grow (Aljohani, 2021b). Not only is deep web data not indexed by search engines, but also their URLs are not defined. Besides, large portions of the hidden web were not explored due to the difficulties associated with finding and accessing hidden web data. The large scale of the dynamically generated content is one of the problems that make deep web crawling a challenge. Also, search interface restrictions make accessing the underlying databases difficult (Raghavan and Garcia-Molina, 2000). In fact, accessing hidden web data requires not only finding the right web forms as sources, but also filling those forms properly (Vieira *et al.*, 2008), processing the queries, and integrating all data. Moreover, the tasks of query translation (across different deep web sources), data retrieval, and integration should be done *on-the-fly* to achieve large-scale data integration. Searching the deep web is considered challenging especially when it comes to coping with the large scale since deep web integration is dynamic, and it is not configured statically for such integration (He *et al.*, 2005a). It has been known that accessing the deep web is demanding as it poses many technical challenges. Additionally, it requires combining multiple techniques from different areas of computer science (Cali, 2017). There exists a gap between the theoretical understanding of the problem of querying the deep web and the practical approaches to it (Cali and Ugarte, 2017). Even the simplest query on the deep web requires the execution of recursive query plans.

2.4 TOWARD SEARCHING AND ACCESSING THE DEEP WEB

The deep web is the biggest source of structured data on the web. However, accessing its contents has historically been a challenge (Madhavan *et al.*, 2009). The rapid increase in the number of data sources and the availability of suitable infrastructure are some of the reasons behind the attention on integrating data from multiple resources. But most important is the hidden data that can only be accessed through web forms because they occupy a larger portion of the web and because of the high data quality that makes it attractive for integrating and accessing. Even though researchers are aware of the difficulties involved with accessing the deep web, they are still working

TABLE 2.1: A Comparison between Physical and Virtual Integration Systems

Physical Integration	Virtual Integration
Data Replication	No Data Replication
Follows a global schema in a central repository	Enables a uniform view of data sources
High integration cost	Reduces integration cost (Rivero <i>et al.</i> , 2011)
Queries are executed against a copy of the data sources	Queries are executed against the original data source (Kabisch, 2011)

on searching or "Googling" the deep web (Calì *et al.*, 2017)(Calì *et al.*, 2015). There are many attempts toward accessing the deep web, but the query processing attempt is the one that was carefully investigated lately. Different perspectives have been tackled such as: data sources integration, query planning and its optimization, and data crawling (Hernández *et al.*, 2018). Certain systems and techniques such as (He *et al.*, 2005a) and (Kabisch *et al.*, 2010) and (Liu *et al.*, 2010) have attempted to dynamically access and leverage deep web content that is relevant and permissible in the context of a query (Jou, 2016a), but these attempts were focusing on specific parts such as interface integration and they do not provide a reusable framework for the research community to contribute in order to improve the field. This approach does not require prior indexing or harvesting of deep web content. It has been noted in the literature that the work on deep web integration is focused on multiple separated aspects such as schema matching (Bhattacharjee and Jamil, 2009). Some approaches were limited to discovering, organizing, and analyzing web forms to provide a form exploration feature (Barbosa *et al.*, 2010), automatic form filling approaches (Toda *et al.*, 2010) and other research focused on building fully integrated systems that start from HTML pages (Vieira *et al.*, 2008). The motion of accessing the deep web is not limited to accessing the web databases directly, but it also focuses on sending multiple requests, receiving multiple responses (possibly integrating them), and finally making a decision (*filtration*) based on user queries. In general, there are two types of integration systems as shown in Table 2.1.

Also, there are two *data integration* approaches: the *virtual integration* that follows the data integration paradigm for accessing the deep web where a mediator is constructed for a specific domain (Madhavan *et al.*, 2009). This approach allows users to

pose their queries over a mediated schema that is exposed as a search interface (web from). Virtual integration was considered unsuitable for search engines especially with the domain classification challenges. Domains could not be clearly defined on the web ten years ago (Madhavan *et al.*, 2009), but with the advent of ontologies, it is now possible to address this challenge (Huan *et al.*, 2020). On the contrary, the *surfacing approach* (also known as crawling (Álvarez *et al.*, 2007), (Madhavan *et al.*, 2008)) works by pre-computing selected sources to index the results at the end (the keyword search is possible here). The majority of recent research on querying the content of the structured web has a focus on leveraging either the exposed tables in an indexable document form, or using traditional query methods after exposing the content (Jamil and Jagadish, 2015). In addition, some deep web query approaches focus on exposing the page content first, then querying it using standard techniques. Many approaches have been used for querying deep web content by exposing the tables based on their predicted use and then querying the exposed content (He *et al.*, 2004). Also, modeling deep web sources as relational tables enables deep web querying and accessing while respecting access limitations. Besides that, there are other data integration types that are determined based on the domain such as *horizontal integration* that aims to integrate multiple web databases from a single domain, and *vertical integration* that uses multiple sources from multiple domains. Different systems and approaches address different issues related to searching the deep web from multiple perspectives. Obviously, it is hard to ignore the increasing number of data sources that are available over the web. But over the years, the need for integrating these data sources has been growing. Some systems focus on interface integration and matching (*web source integration*); other systems focus on data level integration (*data integration*). We will discuss source integration in section 2.4.1, data integration in section 2.4.2, and finally we will present the concept of the UR model in 2.4.3 to highlight its role in deep web integration.

2.4.1 Deep Web Sources Integration

The integration of query interfaces received much attention in the past (Dragut *et al.*, 2009), (Su *et al.*, 2014), (Wu *et al.*, 2003). Initially, researchers attempted to model

all interfaces with flat schemas while considering 1:1 field mappings over interfaces. This kind of work required arduous parameter tuning. Previous approaches such as (Wu *et al.*, 2004) focused on overcoming earlier limitations of interface matching. The improvements included better interface matching accuracy across different domains. Yet, intervention of users was not avoidable.

Integrating Deep Web sources is most commonly handled by performing the domain-wise integration that is useful but challenging. For instance, the Visual Query Interface Integration System VisQI (Kabisch *et al.*, 2010) was proposed as an integration system that extracts query interfaces from web pages, transforms them into a hierarchical representation, clusters them to different domains, and matches fields from different interfaces within their shared domain. The system was built in a modular way and it only addresses web integration developers. It can help developers to support their deep web integration systems and evaluate extraction and matching algorithms. The architecture of VisQI consists of different components that perform schema tree extraction, classify application domains, and match semantically equivalent nodes from different interfaces into clusters. Two different modes can be used in the system: the extraction mode and the mapping and classification mode. In general, the quality of the extracted data structures outperforms other methods such as (He *et al.*, 2003) and (Zhang *et al.*, 2004). Recently, other techniques were borrowed from the area of distributed information retrieval to be used within a framework for integration of deep web sources such as (Cali and Straccia, 2017). This method adapts the *Global As View* (GAV) (Lenzerini, 2002) approach in order to integrate the deep web resources. The proposed framework aimed to compute queries against a mediated schema that represented the structure of the web sources to perform integration of deep web sources. In this proposed framework, queries are posed on a mediator that performs some tasks such as query rewriting, resource selection, and ranking of results. This approach was supposed to return partial answers to the submitted query. However, it was never implemented.

Likewise other approaches, such as The Prudent Schema Matching Approach for web forms (PruSM) (Nguyen *et al.*, 2010), focused on automating schema matching for web forms. This automated schema matcher can work effectively with a large, hetero-

geneous collection of forms focusing on the frequency of attributes. The availability of the information about web form interfaces is crucial for form filling and schema matching. Also, researchers have attempted to access the deep web and perform query interface integration based on incremental schema matching and merging, such as is proposed with DWQII (Jou, 2016b). The problem with this approach is that it gathers all results in one file without further filtering based on the needs of the user.

A very recent approach (Huan *et al.*, 2020) was also presented to handle query interface schema extraction for the sake of enabling an effective access to the hidden web contents based on domain ontology. Even though many research projects addressed the schema extraction problem, this one is relatively new in the sense that it uses the internal code for the extraction within an ontology. This system and some others such as (He *et al.*, 2005b) are limited to the interface level.

2.4.2 *Deep Web Data Integration*

There is great potential in integrating and re-purposing the enormous amount of structured data available on the Web. For instance, OCTOPUS (Cafarella *et al.*, 2009) is a system that enables users to obtain new datasets gathered from those available on the Web. The idea is to automate manual tasks (e.g. data searching, extracting, and cleaning) using specific operators. The data that is manipulated by OCTOPUS is extracted from HTML tables and lists. There are three main integration operators that OCTOPUS performs:

- *The Search Operator* takes user keywords and a set of relations and returns a cluster of tables that are related to the *user's* keyword query string.
- *The Context Operator* takes one relation and modifies it by adding a data column to the obtained table derived from the source web pages corresponding to the relation.
- *The Extend Operator* provides more relevant data columns to the existing table by performing a join.

The limitation with this system is that it only deals with the shallow web tables.

Moreover, several strategies have been proposed for retrieving hidden web data automatically. For instance, Siphon++ is a Hidden web crawler for retrieving data that are hidden in the deep web behind keyword-based form interfaces. Some approaches use a fixed strategy for query generation, but the strategy in Siphon++ adapts query generation and selection by detecting features of the index underlying the search interface (Vieira *et al.*, 2008). The adaptation to the features of the underlying search interface indexes improves the crawler coverage. The proposed architecture consists of two main components: *the adaptive component* that discovers and selects index features using probe queries that are sent against the search interface, and the *heuristic component* that generates queries to retrieve the hidden content. The heuristic component has two phases: *a sampling phase* that aims to assemble a representative sample of the database, and *a crawling phase* that crawls the database using the most common word in the document of the sample. A sample of the indexes is built at the beginning, before the crawling starts, unlike other crawlers that build the sample within the process. The quality of the sample might affect the performance of Siphon++, and that kind of dependency can be considered as a weakness. Other attempts focused on attribute matching for data integration by applying data mining-based techniques (Liu *et al.*, 2010). The work not only targeted the input-input or output-output attribute matching, but also input-output matching.

It is important to mention federated integration systems such as MetaQuerier (He *et al.*, 2005a) that were proposed for deep web integration (follows the virtual integration approach). MetaQuerier was the first fully integrated system concerned with streamlining and automating web interfaces, and allows on-the-fly translation between unseen sources. It provides a mediated schema that facilitate abstracting the web database forms. Although it is a good integration system, it has some limitations such as not providing a reusable general framework for the community to advance the field. Also the study was limited to one domain and it does not perform searching. Besides that, the interface integration causes some lagging and the output schema is not considered. The system consists of a sequence of three subsystems as follows:

- An Interface Extractor takes HTML pages containing web forms as an input and provides extracted attribute information as an output.

- A Schema Matcher handles the web interface attributes and its semantic correspondences.
- An Interface Unifier generates the unified interface and the mapping that links it with the internal interfaces.

In addition, *Pay As You Go* (PAYGO) (Madhavan *et al.*, 2007) is another system with a framework that has a similar architecture to MetaQuerier. Researchers are motivated to work on virtual integration for the deep web, since it has been realized that the actual value of the deep web comes from integrating its data.

Overall, it is not appropriate for the deep web to follow the traditional access techniques; thus, there is a need to develop a useful and effective technique for accessing the deep web. Some research projects have explored a direction that aims to access the deep web upon user querying by redirecting the user to the appropriate database to be searched through its search interface. For example, MetaQuerier followed this direction (He *et al.*, 2007). There has been a lot of attention on data integration; different systems were constructed to tackle different aspects of deep web data integration. In 2001, a task specific hidden web crawler called the *Hidden Web Exposer* (HiWE) (Raghavan and Garcia-Molina, 2001) was built to address the problem of extracting data from the deep web. The framework was domain specific; and as a crawler it retrieves all content with no consideration of user requirement specification or the relevance of results. Although it automates deep crawling to a great extent, it still requires substantial human intervention. A recent survey addresses the performance measures related the deep web crawling and emphasises the need to create effective crawlers that can be useful in a real world context (Hernández *et al.*, 2018). Although deep web data sources have been usually modeled as relational, some recent work aims to export deep web data as Linked data sets (Calì *et al.*, 2018).

In this section, we outlined most important proposed or developed systems in this area so far and we highlighted its strengths and limitations. We have noticed that the majority of existing projects focus on the interface level and do not preform a complete access for users. After analysing the previous and current systems, we

become cognizant of the fact that searching the deep web was not addressed from the prospective of searching the results of queried web databases.

2.4.3 *The Universal Relations (UR) Model for the Deep Web Search*

The Universal Relations (UR) Model that was proposed by Ullman aims to allow the user to see the database as a simple view that is more easily queried (Ullman, 1982). The simplicity of the view comes from its representation as a single semantic view for the entire database (Jamil, 2021). In the UR concept, a relational database was viewed from a different perspective. This concept was supported by many proposed formal foundations such as (Atzeni and Chan, 1989). Previously however, supporting the UR model was a burden to system developers. Querying the relational database while considering the UR model was a complex task. Also, using the UR model required theoretical and system support. The above challenges made it unattractive to support the UR model. Nevertheless, it happens that the UR model offers some attractive properties that can be leveraged in the design of a structured query language for searching the Deep Web (Jamil, 2021). The UR as a user view is similar to a natural language interface to databases, and researchers have developed several implementations for UR interfaces over the years (Levene, 1992).

As web databases contain large amounts of structured information from different domains, the need for developing automated techniques to access the deep web becomes increasingly important. Web database access and integration requires combining and leveraging multiple techniques and approaches from different domains. As stated above, building such systems requires accurate and scalable schema matchers, form filling capability, and data extraction software (Wrappers). Most importantly, one of the recently proposed models (Jamil and Jagadish, 2015) for the Deep Relational Web is a structured query model that can be used to query the deep web as a virtual database. This model also proposes a **Deep Query Language (DQL)**. DQL is a simple SQL-like language that facilitates searching the deep web as a virtual database. We have highlighted the differences between some related work on integrating and accessing the deep web. Table 2.2 shows some of those systems, their approaches, strengths, and limitations. As shown in the table, the problem had been tackled from

different perspectives and levels that are implemented as subsystems since a complete query-answering task is not completely achieved.

Table 2.3 compares the capabilities of those systems. Some systems perform form identification or selection, some systems perform classification at the interface level or domain level. Other systems apply interface integration, where others apply schema integration. We notice that most of these systems do not provide a reusable framework, are limited to one domain, require some user intervention, do not provide a unified user query interface, and if they provide one it is restricted to the access limitations of the underlying interface forms or only for developers. Also, the majority of these systems do not provide final integrated results to the user through a unified interface and they do not allow searching. Furthermore, the focus is directed either to interface level integration, or to presenting incomplete deep web integration systems.

TABLE 2.2: Some Systems Strengths and Limitations.

System	Approach	Strengths	Limitations
MetaQuerier (He et al., 2005a)	Form modeling and Query translation	<ul style="list-style-type: none"> - On-the-fly Querying - Form modeling and query translation - Dynamic Fashion - In the context of the query - Unifies web interfaces automatically 	<ul style="list-style-type: none"> - No tasks identification - No capability of integrating better schema matchers - Users intervention (form selection) - No unified result for users queries - The framework is not reusable and general - No final result filtering
VisQI (Kabisch et al.,2010)	Hidden data behind Keyword-based interface	<ul style="list-style-type: none"> - Adaption to the indexes of the features underlying the web interface 	<ul style="list-style-type: none"> - The coverage is the most important factor - Quality of the sample dependency - Harvesting-based system
OCTOPUS (Cafarella et al., 2009)	Data integration from relational web (Shallow web)	<ul style="list-style-type: none"> - Uses operators for searching and integrating data 	<ul style="list-style-type: none"> - Not fully automated - Users intervention - Real time user interaction is not supported
WISE-integrator (He et al., 2003)	Automatic integration of Web Interfaces of Search Engines	<ul style="list-style-type: none"> - Automatically integrates the interfaces of e-commerce search engines - Not only schema integration, but also attributes values, format, and layout integration 	<ul style="list-style-type: none"> - Interface level integration - Users add and removes interfaces - Integrated interfaces needs to be maintained periodically
WISE-iExtractor (He et al., 2005b)	Interface Schema Extraction	<ul style="list-style-type: none"> - Introduced schema model for search interface - Presented a tool for automatically extracting and deriving information to construct the schema 	<ul style="list-style-type: none"> - Interface level - Single domain - Limited to output the schema of search interfaces
DWQII (Dragut et al., 2012)	Query Interface Integration	<ul style="list-style-type: none"> - Interface integration is implemented based on incremental schema matching and merging - It handles form submission for sites with cookie 	<ul style="list-style-type: none"> - Limited to one domain - Limited to user's preference web sites - Users review is needed for confirming the schema matching - No proper results handling
- (Huan et al., 2020)	Domain Ontology based	<ul style="list-style-type: none"> - Interface schema extraction based on domain ontology - Proposed a new presentation for query interface attributes - Combining attributes semantically 	<ul style="list-style-type: none"> - Interface level integration - Dose not obtain actual results

TABLE 2.3: Comparison of System Capabilities, Identification (Source Identification), Classification, Integration, UUQI(Unified User Query Interface), SM (Schema Matching), FF (Form Filling), Results Wrapping, Reusability.

System	Identification	Classification	Integration	UUQI	SM	FF	Wrapping	Reusable
MetaQuerier (He <i>et al.</i> , 2005a)	✓	✓(Clustering)	✓	✓	✓	-	-	X
DeepPeep (Barbosa <i>et al.</i> , 2010)	✓	✓	X	X	✓	X	X	-
VisQI (Kabisch <i>et al.</i> , 2010)	✓	✓	✓(Interface Level)	X (Developers Only)	✓	X	-	✓
Siphon++ (Vieira <i>et al.</i> , 2008)	✓	- (Sampling and Ranking)	-	✓	X	X	-	-
OCTOPUS (Cafarella <i>et al.</i> , 2009)	✓	✓	✓	✓	X	X	X	X
WISE-integrator (He <i>et al.</i> , 2003)	X	✓	✓(Interface Level)	✓(Interface Integration)	✓	X	X	X
WISE-Extractor (He <i>et al.</i> , 2005b)	✓(Attribute-element Level)	✓(Interface Level)	✓	X	X	X	X	X
- (Liu <i>et al.</i> , 2010)	✓	✓(Clustering)	✓	X	✓	X	X	X
DWQII (Draigt, 2012)	X	X	✓(Interface integration)	✓	✓	✓	X	✓
-(Huan <i>et al.</i> , 2020)	✓	✓	X	X	X	X	X	✓

There have been notable attempts toward making the deep web usable and accessible. We recognize considerable achievements in many tracks toward deep web data integration. It is also noticeable that there is no project to our knowledge that contribute to all subcategories in a modular way except the DQL framework (Jamil and Jagadish, 2015) that emphasis on the importance of making reusable framework that the community can work on together. Figure 2.2 shows some systems that aim to address the problem of leveraging deep web data. Two different approaches are shown in the figure, both of them are actually moving toward making the deep web usable. We have presented these approaches as two different directions that complement each other.

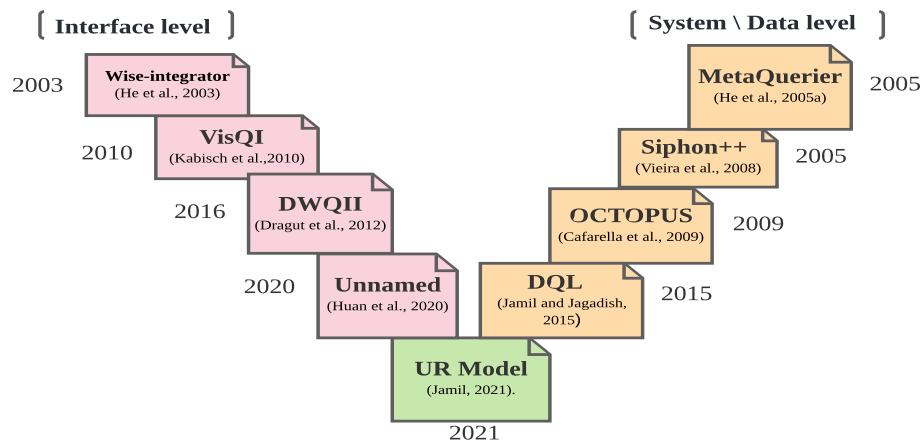


FIGURE 2.2: Some Selected Previous Work Throughout the years

Although there are many other approaches, we consider them as sub-elements in the vision for searching the deep web. The two main levels that can be imagined as a car. The *interface level* which can be seen as the body and the actual *system or data level* can be seen as an engine. Both directions can collaborate with each other alighted by the concept of the UR Model to make deep web searchable in a human-like manner. For instance, while some projects focused on the Form crawling, other projects perform deeper tasks such as schema matching and forms extraction. Obtaining useful results require diving deeper to perform data integration and searching on the retrieved deep web data. An overview of the future direction for searching the deep web from interface level to data level is shown in Figure 2.3. It is important

to mention that this overview is meant to show the big picture and the tasks are not limited to the ones shown in the figure. Some high profile deep web applications such as Metaquerier already exist, but those are very inflexible and do not improve the user experience of querying and searching deep web. They do allow querying, but they do not allow searching.

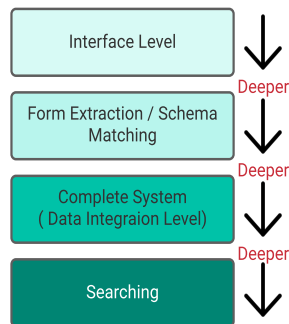


FIGURE 2.3: An overview toward deep web searching.

CHAPTER 3

DEEPQ SYSTEM

This chapter describes our work and the implementation flow of our system *DeepQ* that is utilized to search the database contents behind the firewall inside the Deep Web. The building blocks of DeepQ is shown in Figure 3.1. The DeepQ leverages a declarative query language DQL and simplifies the deep web for users through leveraging the concept of the universal relational model. We will begin with describing how we leveraged the DQL features in section 3.1, then the DeepQ Architecture in section 3.2.

3.1 LEVERAGING THE DQL FEATURES

The model consists of four steps that can be implemented and developed independently. The four steps are listed below:

1. Site Selection
2. Schema Mapping and Form Filling
3. Table Extraction
4. Output Processing

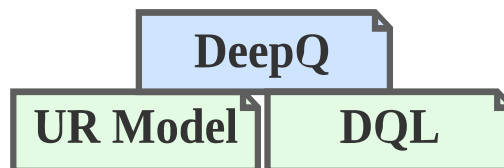


FIGURE 3.1: DeepQ system

The second and third steps are implemented as operators: a *Transform Operator* and a *Combine Operator* respectively. The DQL syntax is simple as it is a SQL-like language that was proposed within a structured query model to query the relational web. DQL has only two clauses: **list** and **where**. The **list** clause is followed by all attributes that users need to obtain in the result separated by a comma, and the **where** clause is followed by the conditions that should be satisfied by the system, using the needed relational operators such as ($= > <$). DQL includes a sentence that stands in parallel to SQL's *select from where* that is used to extract tables from deep web databases. The general structure of the extract statement is as follows:

```

extract Attribute List
using matcher  $\mu$  filler  $\phi$  wrapper  $\omega$ 
from  $\varphi$ 
where  $\theta$ 
submit  $r$ 

```

Although the model allows operations over unstructured and semi structured data, it is completely relational. The ultimate goal is to have a uniform view of data as relations. In the above statement, the *Attribute List* returns a table, and the input argument to the extract statement are the rows of values in the table r . These values are sent as parameters to the optional list of functions μ (a schema matcher for schema heterogeneity resolution), ϕ (a form filling function for deep web form filling), and ω (a wrapper for table structure identification and selection), to extract information at the URL at φ . The filler function is not needed if the URL is a shallow web site (i.e., a document). Finally, θ is the query condition.

It is important to mention that an ontology can be built and associated with the system. This can enable writing general queries without restricting users to specific keywords (Roitman and Gal, 2006). Also a good schema matcher can be beneficial for this task.

3.2 DEEPQ ARCHITECTURE

The functionality of the *extract* statement is leveraged to implement a graphical interface in order to facilitate search as DeepQ system interface. The user can provide a query that has two parts: a list of attributes that they want to obtain as a response for their query, and a set of Boolean conditions that must be satisfied. The detailed DeepQ architecture is illustrated in Figure 3.2.

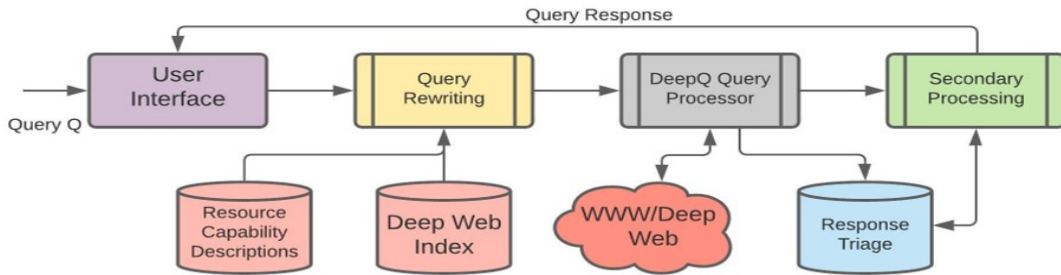


FIGURE 3.2: DeepQ architecture.

For example, a user might want to have a list of Honda Civic EX vehicles which does not have more than 120,000 miles on it, the exterior color is red, it is priced below \$5,000, and the dealer is located in Moscow, Idaho. He wants to know the dealer name and address, car's VIN number, exact price and the dealer's contact phone. She will formulate her DeepQ query as follows in the user interface:

List: *DealerName, DealerAddress, VIN, Price, DealerContactPhone*

Conditions: *miles \leq 120000 and make = Honda and model = Civic and trim = EX and color = red and price \leq 5000 and zipcode = 83843*

Once the query in the above form is submitted, the Query Rewriting unit of DeepQ then consults the Deep Web Index it maintains to find websites that can answer the query such as autotrader.com and cargurus.com and reformulate the queries as the following extract statements (similarly for cargurus.com). We adopt a syntax along the following line from a data integration language called BioFlow (Jamil and El-Hajj-Diab, 2008):

```

extract DealerName, DealerAddress, VIN, Price, DealerContactPhone
using matcher filler wrapper
  
```

where $miles \leq 120000$ and $make = Honda$ and $model = Civic$ and
 $trim = EX$ and $color = red$ and $price \leq 5000$ and $zipcode = 83843$
 from *autotrader.com*

Once the query in the above form is submitted, the Query Rewriting unit of DeepQ then consults the Deep Web Index it maintains to find websites such as *autotrader.com* and *cargurus.com* and reformulate the queries as the following statement.

extract *DealerName, DealerAddress, VIN, Price, DealerContactPhone*
 using matcher, filler, wrapper
 where $miles \leq 120000$ and $make = Honda$ and $model = Civic$ and
 $trim = EX$ and $color = red$ and $price \leq 5000$ and $zipcode = 83843$
 from *autotrader.com*

These extract queries are then annotated and reformulated extensively to simulate the web site interaction to be able to interrogate the deep web site.

For example, in the *autotrader.com*, the front-end only allows submission of vehicle make, model name, and zip code. In the next step, it allows filters to weed out unwanted responses. DeepQ therefore will break down the above query into the following two queries:

extract *
 using matcher filler wrapper
 where $make = Honda$ and $model = Civic$ and $zipcode = 83843$
 from *autotrader.com*

extract *DealerName, DealerAddress, VIN, Price, DealerContactPhone*
 using matcher filler wrapper
 where $miles \leq 120000$ and $trim = EX$ and $color = red$ and $price \leq 5000$
 from *autotrader.com*

These statements are executed in tandem and the query variables are matched with the site variables using a schema matcher. Finally, after extracting the data using a wrapper, DeepQ stores them in the Response Triage to process later.

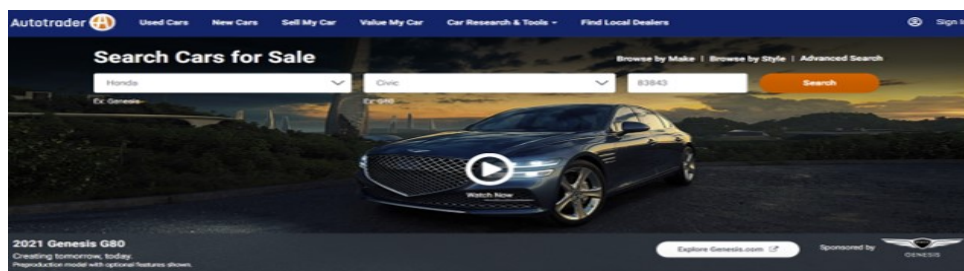


FIGURE 3.3: autotrader.com landing page pre-filter – pick-and-filter querying.

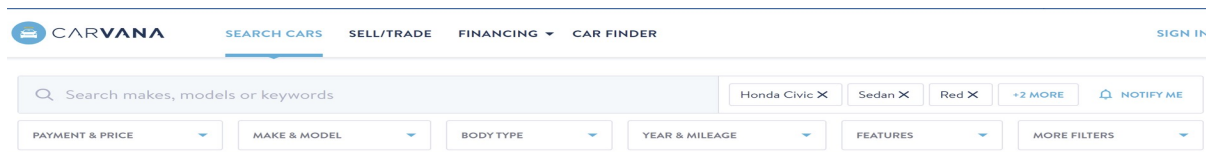


FIGURE 3.4: carvana.com landing page filter – direct querying.

The Query Processing unit processes these queries to collect all tentative responses as candidates. Any further filters that could not be processed at the source, are then applied at the DeepQ local database Response Triage using a simple SQL query, and the final response returned to the users.

3.2.1 Pick-and- Filter Interface

The design of the deep web interfaces vary widely even though there are similarities. Many of these data sources use a pre-filter in their landing page to steer the queries to a more focused exploration, e.g., autotrader.com. In its landing page it only allows to query based on make, model and zipcode, which then leads the users to second page where a plethora of filter conditions can be set to find the vehicle of interest. We term these types of deep web interfaces as *Pick-and-Filter* type interfaces. On the other hand, sites such as carvana.com use the landing page itself for user query submission, which we define as direct filtering-based exploration. Five major conditions are listed for users to choose from – payment and price, make and model, body type, year and mileage, and features as well as more filters, if needed. Figure 3.3 and Figure 3.4 show both types of landing pages while Figure 1.3 shows the secondary filtering page for autotrader.com’s pick-and-filter type interface.

3.2.2 *Event-Driven Resource Indexing*

In the current edition of DeepQ, the databases Deep Web Index and Resource Capability Descriptions are semi-manually extracted, stored and maintained. However, the next edition of DeepQ will have a module to extract such meta-data fully automatically. Fortunately, many of the databases today use identical designs because many of them are designed using services such as Shopify⁴ and Wix⁵, even though the number of such sites are small. This will expedite the process of meta-data collection because we are able to apply almost the same set of filter conditions across various sites. One such obvious example is that Expedia.com and Orbitz.com have almost identical interfaces even though they are not designed using the likes of Shopify or Wix.

3.3 SYSTEM OVERVIEW

The system we are implementing mimics the Structured Query Model for the Deep Relational Web (He *et al.*, 2005a). A simple diagram of the implemented system is illustrated in Figure 3.5. It shows three main parts: a text user interface, a CGI (Common Gateway Interface) with related Databases, and deep web forms. In order to access the deep web effectively and obtain the useful information, users will interact directly with a text interface. The text interface allows users to submit a query that is written in DQL to obtain an output that is gathered not only from one form interface, but also from all forms that can provide the desired results. The user text interface runs a CGI script that provides a dynamic web page dependent upon the submitted DQL query. The CGI script works as an intermediary between the text interface and the other deep web forms.

3.3.1 *Deep Web forms*

Deep web forms (or search interfaces) are designed to facilitate human-computer interactions; in order to access the deep web via DQL, this interaction needs to be

⁴<https://www.shopify.com/>

⁵<https://www.wix.com/>

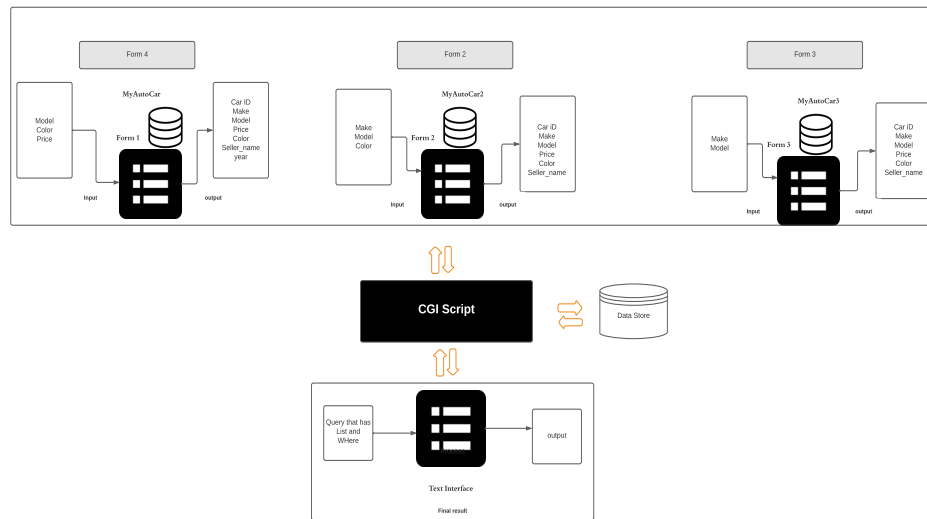


FIGURE 3.5: DeepQ System Overview for a single domain (Simple Query).

simulated automatically. Users are able to fill in fields in web forms with specific parameters; usually each form is an interface for an online web database with unique schema and does not necessarily belong to one specific domain. Search forms are treated as black boxes in the DeepQ, but in order to understand the underlying web databases and leverage their contents, search forms should be understood clearly. Forms usually have selectable input requirements (access limitation), and expected output. In traditional scenarios, if users tend to access one of these forms directly, they would need to fill the form, send the query, and the result is automatically generated after querying its related database. For example, in Figure 3.5 there are three different forms that belong to a single domain (car sales) from three different web sites. Form4 allows users to select model, color, and price to find a car that meets the user's selections and retrieve the available information from the database as the following: Car Id, Make, Model, Color, Price, Seller Name, and Year (see Figure 3.6), also another example is Form2 in Figure 3.7. In addition, the Table 3.1 shows the input and output attributes for five forms.

It is important to mention that no dynamic scripts such as JavaScript were involved with the building process of any of our forms. Web sites that use JavaScript have been

Search Vehicles

Customized Search

Select car model (Jeep)

Model :

Select a color:

Black Blue Red Silver White Gray


Select a price range :

5,000 or less 10,000 or less 15,000 or less
 20,000 or less 30,000 or less 40,000 or less

Result:

Car IDDD	Make	Model	Price	Color	Seller_name	year
350	Jeep	Compass	24995	Black	Rogers Chrysler Dodge Jeep RAM	2020

FIGURE 3.6: Form 4 after submitting as an example



Find your perfect car

Autocar 2

Make
Select Make:

Model
Select Model:

Color
 Red Silver Brown BLACK

Car ID	Make	Model	Price	Color	Year
1	Jeep	Cherokee	27085	Silver	2020

© 2021 MyAuto.com. All rights reserved.

FIGURE 3.7: Form 2 after submitting as an example

known to pose a challenge for web crawling. This is one of the unsolved issues when dealing with web site crawling (Hernández *et al.*, 2018).

TABLE 3.1: Input/Output Attributes for each form.

Fom	Input	Output
Form 2	Make, Model, Color	ID, Make, Model, Color, Price, Seller Name
Form 3	Make, Model	Car ID, Make, Model, Color, Price, Seller Name, Zipcode
Form 4	Model, Color, Price	Car ID, Make, Model, Price, Seller Name, Year
Form 5	Zipcode	Zipcode, State, City, Elevation
Form 6	Make, Model	Car ID, Make, Model, Shade, Cost, Seller Name

3.3.2 *The CGI Script and related databases*

The creation of dynamic web contents is assisted by some mechanisms and technologies like server-side programs that are the most common/oldest method for generating web pages on the fly. CGI is an example (Raghavan and Garcia-Molina, 2000).

The CGI script performs the following on submit:

- Get keys from user query by identifying attributes associated with the where clause and the list clause.
- Recommending candidate forms based on both input-output attributes. Form input is considered as access limitations that enforces internal constraints. The system specifies relevant candidate forms that can be queried and premised to obtain the desired output. This interface identification task can be implemented by using a recommender system; the implemented algorithm mimics the recommender system task as follows:

For example, if there is a form that accepts only make and model as inputs, but the user query contains not only make and model. What if she asked for also color or seller information that only can be found within the output? Considering only form input is not efficient, the information desired might exist in a specific form but will not be obtained if the system just considers form access limitations (input). It is more efficient and useful to consider also output information; see the example in section 4.3.

- Schema matching and form filling
- Get all response and write them to one table on-the-fly (SQLite is used)
- Evaluate user query against that table and output processing

OPERATORS — We have implemented the two operators proposed in (Jamil and Jagadish, 2015) as following:

- The transform operator is implemented as a function that takes three parameters: a URL, θ , and list attributes (output). The theta part(θ) represents the Boolean conditions in the 'where' clause. And the list attribute are simply the attributes that users asked for. This function returns a table depending on sites capabilities.
- The combine operator is responsible for output processing, it applies UNION on simple queries and JOIN on Complex queries.

3.3.3 User Interface

We are experimenting with two dramatically different query interfaces even though both rely on the principles of the DQL query language.

- *The graphical user interface* allows users to select the output field names and the query conditions that corresponds to (list, conditions) in DeepQ and return response. See Figure 3.8.
- *The free text user interface* shown in Figure 3.9, is a simple interface that has a text box and submit button and it allows unrestricted queries. Sophisticated users can write their queries directly in DQL without worrying about manual picking and filtering, filling multiple forms, making separate decisions, and comparing between results. The simple free text user interface reduces user intervention while accessing the deep web, which can improve scalability (Hernández *et al.*, 2018).

Search Vehicles by make , model , and color

Make : Model Select Model
 Cherokee
 Wrangler
 Grand Cherokee

Select a color :

Black Blue Red Silver
 White Gray
 Granite Crystal

Select a price range :

5,000 or less 10,000 or less 15,000 or less 20,000 or less 30,000 or less 40,000 or less

Select a year :


2020 2018 2019 2009

Results should include

Select here for results : Price Year Seller Color

Result:

FIGURE 3.8: Graphical Interface (restricted)



DEEPQ

Search :

© 2021 DeepQ.com. All rights reserved.

FIGURE 3.9: Text Interface

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have implemented a desktop version of the system that is able to process queries on deep web databases on the localhost as a proof of concept. The system was implemented using Apache web server, PHP, and MySQL. The user interface page runs a CGI script when a query is submitted. Our CGI script is implemented in Python, and it interfaces with SQLite.

4.1 SYNTAX CHECKER

The syntax checker function checks the following:

- If the query begins with "list"
- If there is more than one attribute in the (list clause) attributes, they should be separated by a comma
- If the condition part (the where clause) begins with "where"
- If the (where clause) has pairs that are compared by (=,<,>,<=,>=)
- OR and limit are optional
- OR separates two blocks in a query, see Figure 4.1

4.2 SINGLE DOMAIN WEB DATABASES(SIMPLE QUERY)- AN EXAMPLE

The deep web sites' interfaces are mostly heterogeneous. Processing deep web queries in a single domain is considered as a simple querying process (Jamil, 2021).

Assume that a user is searching for a black Jeep Cherokee and submits the following query to the Text User Interface:

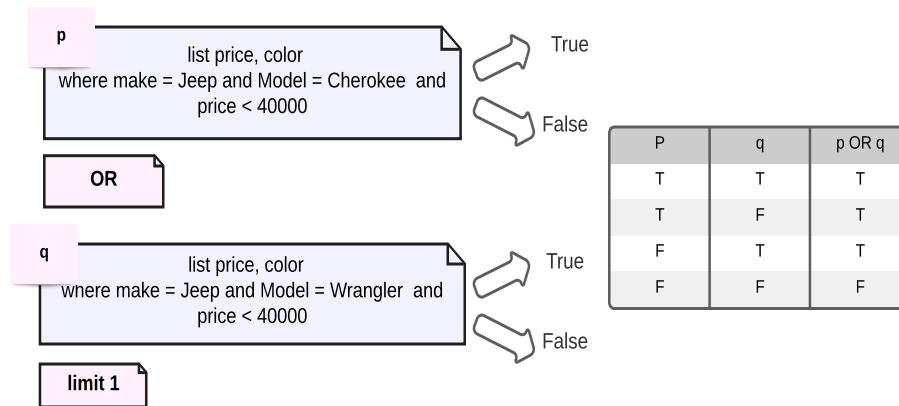


FIGURE 4.1: The logic in the Syntax Checker.

Q₃: list price, color where make = Jeep and model = Cherokee and color = Black

RESULTS FOR SIMPLE QUERIES — The result is shown in Figure 4.2; the output show price and color in the result as the user requested. Figure 4.3 shows results for the same query with a small modification (Color = Red) as the following:

Q₃: list price, color where make = Jeep and model = Cherokee and color = Red

The obtained results might have different schema. For example some forms provide *shade and cost* in the output, while others have *price and color*. Consider the following query:

Q₄: list shade, price where make = Jeep and model = Cherokee and price < 30000

The result is shown in Figure 4.4.

DEEPOQ

Search :

list price, color
where make = Jeep and model = Cherokee and color = Black

Submit

price	color
20340	Black

Duration: 0.179 sec

© 2022 DeepQ.com. All rights reserved.

FIGURE 4.2: Results for Q₃

DEEPOQ

Search :

list price, color
where make = Jeep and model = Cherokee and color = Red

Submit

○ No results !!

Duration: 0.042 sec

© 2022 DeepQ.com. All rights reserved.

FIGURE 4.3: A small modification in Q₃

4.3 MULTIPLE DOMAINS WEB DATABASES (COMPLEX QUERY)- AN EXAMPLE

Our examples in section 1.5 show that queries can also be complex and they are not always limited to a single domain. An overview of the process that deals with querying the multiple domain web databases is shown in Figure 4.5.

Consider the following query:

Q₂: Find Red Honda Civic 2015 or newer model less than \$20,000 in cities at elevation lower than 4000 feet



FIGURE 4.4: Results for Q4

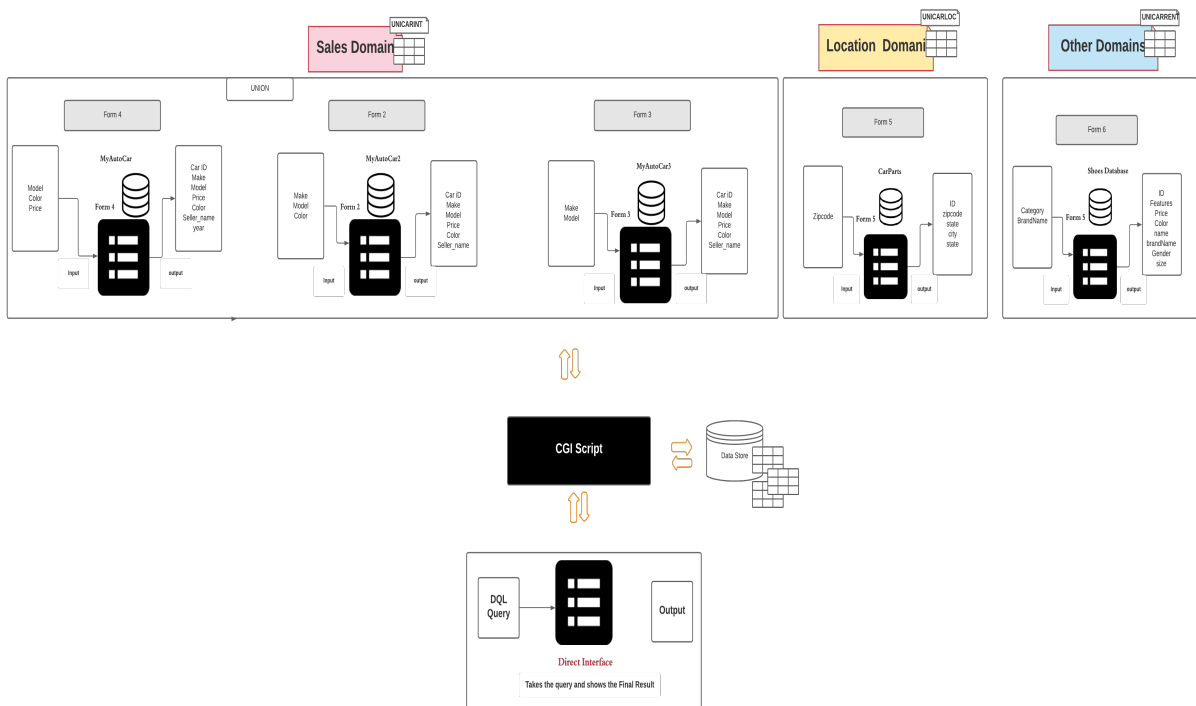


FIGURE 4.5: DeepQ System Overview for the multiple domain querying.

This query can be successfully computed by linking retrieved information from different deep web sites, see Figure 4.6. As you can see, Q_2 has two different parts since its requirements are not limited to one domain.

part1: Red Honda Civic 2015 or newer model less than \$20,000 -> (Cars Domain)

part2: cities at elevation lower than 4000 feet -> (Maps/Elevations Domain)

Linking between the two parts in this complex query requires leveraging the results from the first part to retrieve results for the second part and then link them (Jamil, 2021). For example, the result shown in Figure 1.7 has some information about the car itself and the location in which it exists (Colorado Springs, CO). The city information can be used with another deep web site that has an elevation database, see Figure 4.7.

In general, the simplest description is; when a user sends a query, the system performs the following:

1. Determine candidates forms by determining the mandatory input keys for each form.
2. Check if other requirements in the query are available within the form's output. (Extras)
3. Send queries and receive results.
4. Filter results based on user query.

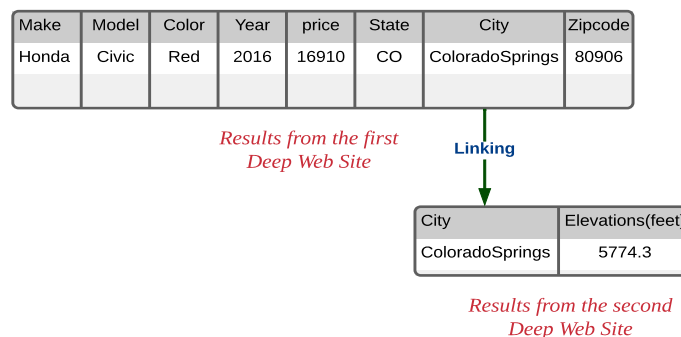
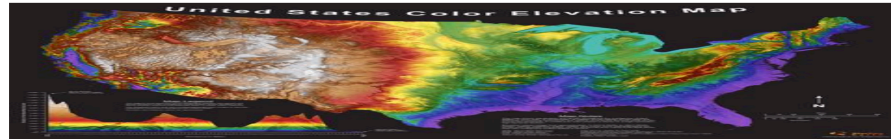


FIGURE 4.6: Linking two different deep web sites that have different domains



US Elevations Map

Search

zipcode:

Result:

zipcode	state	city	elevation
99163	WA	Pullman	2352

FIGURE 4.7: Results from one of the Elevation Databases

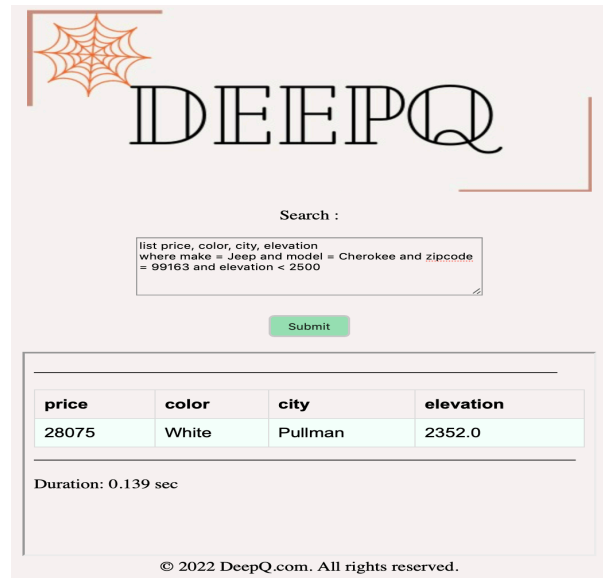
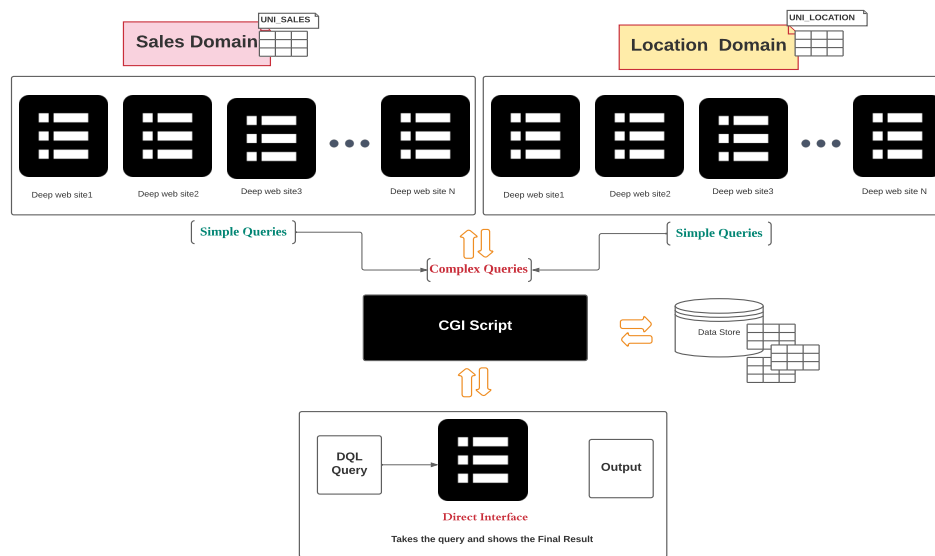
RESULTS FOR COMPLEX QUERIES — Assume that a user submits the following query to the Text User Interface.

Q₅: list price, color, city, elevation where make = Jeep and model = Cherokee and and zipcode = 99163 and elevation < 2500

This complex query is targeting two different domains with multiple deep web sites in each domain. The result is shown in 4.8. The user asks for (price, color, city, elevation) ; the first three attributes can be found in some or all forms that belong to cars domain (based on sites capabilities). But the last attribute (elevation) belongs to another domain. See Figure 4.9

4.4 PERFORMANCE AND EVALUATION CRITERIA

Our goal is to empower users with the ability to search databases behind firewalls. Generally, the performance of deep web searching systems can be captured in different ways, and several factors can be taken into account. For instance, in web crawlers, it is important to capture *effectiveness* and *coverage* (Vieira *et al.*, 2008). Furthermore, with form submission, none of the crawling metrics capture the challenge of dealing with hidden web form submission (Raghavan and Garcia-Molina, 2000). Also, relevance is usually one of the performance measures with web crawlers. One of the goals of the implemented model is enabling users to obtain useful and to-the-point

FIGURE 4.8: Results for Q₅FIGURE 4.9: a complex Query such as Q₄

results. The DeepQ system has the potential of capturing three important factors as shown in Figure 4.10:

- *Effectiveness and Usefulness*: The user gets only the desired result(s).
- *Usability*: The system is easy to be used.

It has been known that the concept of the Universal Relation model can actually solve the *usability* problem because it allows users to interact with databases with no

TABLE 4.1: DeepQ Evaluation (UUQI: Unified User Query Interface)

System	Querying	UUQI	Integrated Result	Searching	Multiple Domains
MetaQuerier	✓	✓	✓	X	X
Siphon++	X	✓	X	X	X
WISE-integrator	X	✓	X	X	X
DeepQ	✓	✓	✓	✓	✓

prior knowledge of its structure (Levene and Loizou, 1994). Also, the system accesses multiple forms in a single or different domains (*Accessibility*), fills them and gets results back, while respecting the form’s access limitations.

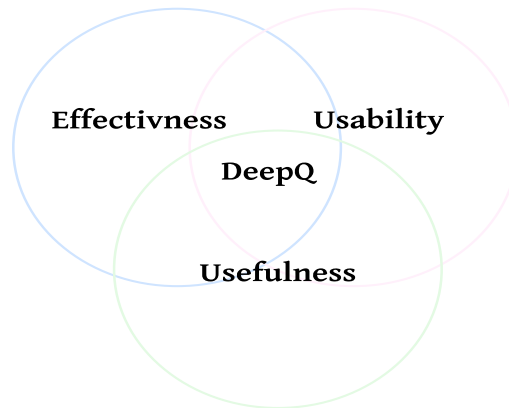


FIGURE 4.10: DeepQ performance measures.

4.4.1 DeepQ Evaluation

A qualitative evaluation for DeepQ compared with other systems is shown in Table 4.1. As shown, DeepQ allows to pose structured queries over a text user interface, query deep web, obtain results through the user interface, search the queried results, and able to perform complex queries not limited to single domain. Most web sites use dynamic scripts such as JavaScript and Ajax for content generating (Hernández *et al.*, 2018). In this work, we are limited to the websites that are not using such dynamic scripts.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTION

5.1 CONCLUSION

As has been noted, accessing and leveraging the deep web is desirable and attractive. We have experimented with the vision that we distilled from all the literature and discussed in this dissertation, which is encapsulated in our system called *DeepQ*. We envision a system that is able to search the database contents behind the firewalls inside the deep web. The vision is not limited to enabling effective access to the deep web; it is also concerned with obtaining useful results that meet a user's queries on-the-fly from multiple domains. This requires performing and integrating several tasks such as schema matching, forms filling, result extraction, integration and searching. We have experimented with some of these features in *DeepQ* using examples in the automobile sales industry. Our experiments show that implementing these features is feasible. These contents can be accessed using a structured query language when they are treated as deep relational web. We have leveraged a recently proposed declarative deep web query language called *DQL*, and we have presented the contours of its implementation in the *DeepQ* system. We believe that this work has the potential to demonstrate the Universal Relations model as the user will be able to interact freely with databases that are hidden behind firewalls. *DeepQ* provides a single database illusion of the underlying databases using a query engine and site specific query reformulation. Users need to obtain the desired results in a useful and effective way without being overwhelmed with having to manually search multiple web forms. Queries from same or different domains can be computed as discussed in our example in section 1.5. Moving from searching the shallow web and perform deeper searching in the deep web can enrich the user experience and increase the quality of the obtained results, therefore enhances the quality of people's lives. We have presented an example from cars domain and elevation, other people might want to find a house in safe location or least-polluted locations. The idea is that different web

databases can be queried and integrated this way. Making the deep web searchable and accessible is desirable for providing more accurate and related results to user queries. This would increase satisfaction which leads to better search experience. Finally, the complexities and the involvement of a diverse set of component tools (e.g., wrappers, matchers, form fillers, etc.) also raises novel questions about efficiency and optimization opportunities, some of which we plan to explore as our future research.

5.2 FUTURE DIRECTION

Moving forward to the future direction requires the involvement of a diverse set of component tools (e.g., wrappers, matchers, form fillers, etc.). There are multiple open issues that need to be addressed and improved in deep web research as highlighted in (Hernández *et al.*, 2018). First, site indexing automation is an important issue that we need to address in our future work. This can be implemented in two ways:

1. Design of a crawler to find those sites a head of time.
2. Implementation of individual query-based search. Every DeepQ query will first be submitted in Google for possible identification of deep web sites that are relevant as determined by Google's powerful search engine. We will choose the top K number of deep web sites to explore and possibly index. We use the schema matcher to find out what fraction of our query variables match the front-end and back-end filters, what variables are returned, and their types. We do so by submitting a probe query using the conditions in the DeepQ query, not necessarily to obtain a result, but just to explore the schema of the site. Once we have explored all K sites, and possibly indexed, we move on to actually process the query using the information we have. This is how we increase our reach into the deep web database space on the internet. Obviously, there are several choices on how we can do this because it has query processing implications. We can decide to just log the sites for now, and process them for meta-data collection during DeepQ idle times as opposed to at query processing time.

Also, we are not handling any transformation of values. A value is a value to us. If someone is looking for a price in Pounds, heights in feet or meters, etc, we cannot deal with that for now. This requires an additional engine to handle those conversions. It is important to keep in mind that web databases are still databases even if they are hidden behind firewalls. Keeping that in mind can affect the way we address the related overhead regarding making the deep web usable and accessible. Moving toward designing and implementing smarter user interfaces for a structured deep web is one of our future goals. Besides that, we are planning to investigate leveraging the relational web to be integrated with the deep web. This process might be needed if a part of the query needs information that is not provided in a deep web site but can be found in the relational web. Also, provenance information is important for assessing integrity and data value and it has been recognized as fundamental to trust in data (Buneman and Tan, 2007). Thus, as general databases provide some support for provenance tracking (Buneman *et al.*, 2006), this support could also be provided to some extent to web databases after they have been accessed and integrated. The question is how to make these systems or interfaces that aim to access the deep web "provenance-aware", and to what extent this awareness should be. It is understandable that with the hidden web, we are limited to capturing some provenance related to the source of the integrated data such as URLs and the time of integration. The captured provenance must respect the nature of the hidden web databases.

BIBLIOGRAPHY

- Agrawal S., Chaudhuri S., and Das G. 2002. Dbxplorer: a system for keyword-based search over relational databases. *In Proceedings 18th International Conference on Data Engineering*, pages 5–16.
- Aljohani A. 2021a. Deepq: A system to peek inside the ecommerce deep web. *In The 20th International Conference on Information Knowledge Engineering*.
- Aljohani A. 2021b. Personalized question answering on the web using an ontology. *In The 7th International Conference on Health Informatics & Medical Systems*.
- Álvarez M., Raposo J., Pan A., Cacheda F., Bellas F., and Carneiro V. 2007. Crawling the content hidden behind web forms. *In International Conference on Computational Science and Its Applications*, pages 322–333, Springer.
- Atzeni P. and Chan E.P. 1989. Efficient optimization of simple chase join expressions. *ACM Transactions on Database Systems (TODS)* 14:212–230.
- Barbosa L., Nguyen H., Nguyen T., Pinnamaneni R., and Freire J. 2010. Creating and exploring web form repositories. *In Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10*, page 1175–1178, Association for Computing Machinery, New York, NY, USA.
- Bergman M.K. 2001. White paper: The deep web: Surfacing hidden value. *The journal of electronic publishing : JEP*. 7.
- Bhattacharjee A. and Jamil H. 2009. Ontomatch: A monotonically improving schema matching system for autonomous data integration. *In 2009 IEEE International Conference on Information Reuse Integration*, pages 318–323.
- Bianchi F., Tagliabue J., and Yu B. 2021. Query2prod2vec grounded word embeddings for ecommerce. *arXiv preprint arXiv:2104.02061* .
- Buneman P., Chapman A., and Cheney J. 2006. Provenance management in curated databases. *In Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 539–550.
- Buneman P. and Tan W.C. 2007. Provenance in databases. *In Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1171–1173.
- Cafarella M.J., Halevy A., and Khoussainova N. 2009. Data integration for the relational web. *Proc. VLDB Endow.* 2:1090–1101.
- Cali A. 2017. Querying and searching the deep web. *In Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017* (R. Akerkar, A. Cuzzocrea, J. Cao, and M. Hacid, eds.), page 3:1, ACM.
- Cali A., Martinenghi D., and Torlone R. 2015. Keyword search in the deep web. *In CEUR Workshop Proceedings*, 1378, CEUR.

- Cali A., Martinenghi D., Torlone R., *et al.* 2017. Googling the deep web. *In Proceedings of the 25th Italian Symposium on Advanced Database Systems, Squillace Lido (Catanzaro), Italy, June 25-29, 2017*, vol. 2037 of *CEUR Workshop Proceedings*, page 49, CEUR-WS.org.
- Cali A., Noia T.D., Lynch T.W., and Ragone A. 2018. Processing SPARQL queries on deep web sources. *In Proceedings of the 26th Italian Symposium on Advanced Database Systems, Castellaneta Marina (Taranto), Italy, June 24-27, 2018 (S. Bergamaschi, T.D. Noia, and A. Maurino, eds.)*, vol. 2161 of *CEUR Workshop Proceedings*, CEUR-WS.org.
- Cali A. and Straccia U. 2017. Integration of deep web sources: A distributed information retrieval approach. *In Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*, pages 1–4.
- Cali A. and Ugarte M. 2017. Accessing the deep web with keywords: A foundational approach. *In Semantic Keyword-based Search on Structured Data Sources*, pages 177–183, Springer.
- Ceaparu I., Lazar J., Bessiere K., Robinson J., and Shneiderman B. 2004. Determining causes and severity of end-user frustration. *International journal of human-computer interaction* 17:333–356.
- Dragut E.C. 2012. Deep web query interface understanding and integration. *Synthesis Lectures on Data Management*, Morgan Claypool, San Rafael, Calif. (1537 Fourth Street, San Rafael, CA 94901 USA).
- Dragut E.C., Kabisch T., Yu C., and Leser U. 2009. A hierarchical approach to model web query interfaces for web source integration. *Proc. VLDB Endow.* 2:325–336.
- El-Gamil B.R., Winiwarter W., Božić B., and Wahl H. 2011. Deep web integrated systems: current achievements and open issues. *In iiWAS'2011 - The 13th International Conference on Information Integration and Web-based Applications and Services*, 5-7 December 2011, Ho Chi Minh City, Vietnam, pages 447–450, ACM.
- Elhabbak H., Descamps B., Fischer E., and Athanasiadis S. 2020. Contextualisation of ecommerce users. *arXiv preprint arXiv:2011.01874* .
- Florescu D., Levy A., and Mendelzon A. 1998. Database techniques for the world-wide web: A survey. *ACM SIGMOD Record* 27:59–74.
- He B., Patel M., Zhang Z., and Chang K.C.C. 2007. Accessing the deep web. *Communications of the ACM* 50:94–101.
- He B., Zhang Z., and Chang K.C.C. 2004. Knocking the door to the deep web: Integrating web query interfaces. *In Proceedings of the 2004 ACM SIGMOD international conference on Management of Data*, pages 913–914.
- He B., Zhang Z., and Chang K.C.C. 2005a. Metaquerier: querying structured web sources on-the-fly. *In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 927–929.

- He H., Meng W., Yu C., and Wu Z. 2003. Wise-integrator: An automatic integrator of web search interfaces for e-commerce. *In Proceedings 2003 VLDB Conference*, pages 357–368, Elsevier.
- He H., Meng W., Yu C., and Wu Z. 2005b. Constructing interface schemas for search interfaces of web databases. *In International Conference on Web Information Systems Engineering*, pages 29–42, Springer.
- Hernández I., Rivero C.R., and Ruiz D. 2018. engDeep web crawling: a survey. *World wide web (Bussum)* 22:1577–1610.
- Hristidis V. and Papakonstantinou Y. 2002. Discover: Keyword search in relational databases. *In VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 670–681, Elsevier.
- Huan A.Z., Panfei Y., and Zitong Y. 2020. Query interface schema extraction for hidden web resources searching. *In 2020 7th International Conference on Information Science and Control Engineering (ICISCE)*, pages 1058–1062, IEEE.
- Jagadish H.V., Chapman A., Elkiss A., Jayapandian M., Li Y., Nandi A., and Yu C. 2007. Making database systems usable. *In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD '07*, page 13–24, Association for Computing Machinery, New York, NY, USA.
- Jamil H. and El-Hajj-Diab B. 2008. Bioflow: A web-based declarative workflow language for life sciences. *In 2008 IEEE Congress on Services - Part I*, pages 453–460.
- Jamil H.M. 2021. Ur strikes back, again - for deep web search. Tech. rep., Department of Computer Science, University of Idaho, Moscow, ID.
- Jamil H.M. and Jagadish H.V. 2015. A structured query model for the deep relational web. *In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1679–1682.
- Jou C. 2016a. Deep web query interface integration based on incremental schema matching and merging. *In Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016*, pages 1–7.
- Jou C. 2016b. Deep web query interface integration based on incremental schema matching and merging. *In Proceedings of the The 3rd Multidisciplinary International Social Networks Conference on SocialInformatics 2016, Data Science 2016, MISNC, SI, DS 2016*, Association for Computing Machinery, New York, NY, USA.
- Kabisch T. 2011. Extraction and integration of web query interfaces. ph.d. thesis .
- Kabisch T., Dragut E.C., Yu C., and Leser U. 2010. Deep web integration with visqi. *Proceedings of the VLDB Endowment* 3:1613–1616.
- Khelghati M. 2016. Deep Web Content Monitoring. Ph.D. thesis, University of Twente, Netherlands, SIKS dissertation series no. 2016-31.
- Lawrence S. and Giles C.L. 1998. Searching the world wide web. *Science* 280:98–100.

- Lenzerini M. 2002. Data integration: A theoretical perspective. *In* Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '02, page 233–246, Association for Computing Machinery, New York, NY, USA.
- Levene M. 1992. The Nested Universal Relation Database Model. Lecture Notes in Computer Science, Springer.
- Levene M. and Loizou G. 1994. The nested universal relation data model. *Journal of Computer and System Sciences* 49:683–717, 30th IEEE Conference on Foundations of Computer Science.
- Liu T., Wang F., and Agrawal G. 2010. Instance discovery and schema matching with applications to biological deep web data integration. *In* International Conference on Data Integration in the Life Sciences, pages 148–163, Springer.
- Madhavan J., Afanasiev L., Antova L., and Halevy A. 2009. Harnessing the deep web: Present and future. arXiv preprint arXiv:0909.1785 .
- Madhavan J., Jeffery S.R., Cohen S., Dong X.L., Ko D., Yu C., and Halevy A. 2007. Web-scale data integration: You can only afford to pay as you go. *In* CIDR.
- Madhavan J., Ko D., Kot L., Ganapathy V., Rasmussen A., and Halevy A. 2008. Google's deep web crawl. *Proceedings of the VLDB Endowment* 1:1241–1252.
- McCoy D., Bauer K., Grunwald D., Kohno T., and Sicker D. 2008. Shining light in dark places: Understanding the tor network. *In* International symposium on privacy enhancing technologies symposium, pages 63–76, Springer.
- Nguyen T.H., Nguyen H., and Freire J. 2010. Prusm: a prudent schema matching approach for web forms. *In* Proceedings of the 19th ACM international conference on Information and knowledge management, pages 1385–1388.
- Raghavan S. and Garcia-Molina H. 2000. Crawling the hidden web. Tech. rep., Stanford.
- Raghavan S. and Garcia-Molina H. 2001. Crawling the hidden web. *In* Proceedings of the 27th International Conference on Very Large Data Bases, VLDB '01, page 129–138, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Rivero C.R., Frantz R.Z., Ruiz D., and Corchuelo R. 2011. On using high-level structured queries for integrating deep-web information sources. *integration* 16:37.
- Roitman H. and Gal A. 2006. Ontobuilder: Fully automatic extraction and consolidation of ontologies from web sources using sequence semantics. pages 573–576, Springer.
- Seys S.F., Daenen M., Dilissen E., Van Thienen R., Bullens D.M., Hespel P., and Dupont L.J. 2013. Effects of high altitude and cold air exposure on airway inflammation in patients with asthma. *Thorax* 68:906–913.

- Su W., Li Y., and Lochovsky F.H. 2014. Query interfaces understanding by statistical parsing. *In Proceedings of the 23rd International Conference on World Wide Web, WWW '14 Companion*, page 1291–1294, Association for Computing Machinery, New York, NY, USA.
- Toda G.A., Cortez E., da Silva A.S., and de Moura E.S. 2010. A probabilistic approach for automatically filling form-based web interfaces. *Proceedings of the VLDB Endowment* 4:151–160.
- Ullman J.D. 1982. The ur strikes back. *In Proceedings of the 1st ACM SIGACT-SIGMOD symposium on Principles of database systems*, pages 10–22.
- Vieira K., Barbosa L., Freire J., and Silva A. 2008. Siphon++ a hidden-webcrawler for keyword-based interfaces. *In Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1361–1362.
- Wu W., Yu C., Doan A., and Meng W. 2004. An interactive clustering-based approach to integrating source query interfaces on the deep web. *In Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 95–106.
- Wu Z., Raghavan V., Qian H., Rama K., Meng W., He H., and Yu C. 2003. Towards automatic incorporation of search engines into a large-scale metasearch engine. *In Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 658–661.
- Zhang Z., He B., and Chang K.C.C. 2004. Understanding web query interfaces: Best-effort parsing with hidden syntax. *In Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 107–118.