Julian Eberius, Maik Thiele, Wolfgang Lehner

# Exploratory Ad-Hoc Analytics for Big Data

**Erstveröffentlichung in / First published in:**

Albert Y. Zomaya, Sherif Sakr, Hgg., 2017. *Handbook of Big Data Technologies*. Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.

DOI: https://doi.org/10.1007/978-3-319-49340-4_11

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

# Exploratory Ad-Hoc Analytics for Big Data

**Julian Eberius, Maik Thiele and Wolfgang Lehner**

**Abstract** In a traditional relational database management system, queries can only be defined over attributes defined in the schema, but are guaranteed to give single, definitive answer structured exactly as specified in the query. In contrast, an information retrieval system allows the user to pose queries without knowledge of a schema, but the result will be a top-k list of possible answers, with no guarantees about the structure or content of the retrieved documents. In this chapter, we present Drill Beyond, a novel IR/RDBMS hybrid system, in which the user seamlessly queries a relational database together with a large corpus of tables extracted from a web crawl. The system allows full SQL queries over a relational database, but additionally enables the user to use arbitrary additional attributes in the query that need not to be defined in the schema. The system then processes this semi-specified query by computing a top-k list of possible query evaluations, each based on different candidate web data sources, thus mixing properties of two worlds RDBMS and IR systems.

## 1 Exploratory Analytics for Big Data

While the term *Big Data* is most often associated with the challenges and opportunities of today's growth in data volume and velocity, the phenomenon is also characterized by the increasing *variety* of data [36]. In fact, data is collected in more and more different forms from increasingly heterogeneous sources. The spectrum of additional data sources ranges from large-scale sensor networks, over measurements from mobile clients or industrial machinery, to the log- and click-streams of ever

J. Eberius · M. Thiele (✉) · W. Lehner
Faculty of Computer Science, Database Technology Group,
Technische Universität Dresden, 01062 Dresden, Germany
e-mail: maik.thiele@tu-dresden.de

J. Eberius
e-mail: julian.eberius@tu-dresden.de

W. Lehner
e-mail: wolfgang.lehner@tu-dresden.de

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

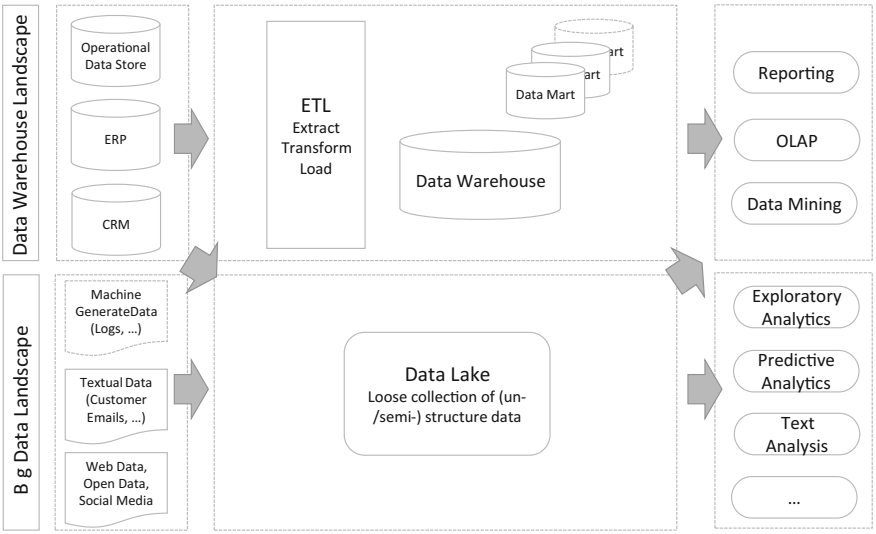



**Fig. 1** The growing big data analytics landscape

more complex software architectures and applications. In addition, there is more publicly available data, such as social network data, as well as Web and Open Data. More and more organizations strive to efficiently harness all forms and sources of data in their analysis projects to gain new insights, or enable new features in their products.

However, conventional data warehouse infrastructures (upper part of Fig. 1) assume controlled ETL processes with well-defined input and target schemata, that define the data pipelines in the organization. The data sources typically are operational databases and common enterprise IT systems, such as Customer Relationship Management (CRM) and Enterprise Resource Planning (ERP) systems. Traditionally, the data sink in such an architecture has been the warehouse or data marts, whose schemata define what is immediately queryable for an analyst. If there is an ad-hoc information need that can not be satisfied with the current schema because external information has to be integrated, an intricate process has to be followed. Because the warehouse is a crucial piece of infrastructure, it is highly controlled: ad-hoc integration and analytics is not a feature that it is designed for. Still organizations today aim at generating value from all available data, which includes novel internal, but also increasingly external sources. Consider the lower part of Fig. 1, which depicts key changes to the traditional architecture. Beside the continued growth of data volume and its increasing heterogeneity, there are also changes at the data consumption side, where we can see a development towards agile and exploratory data analysis overcoming inflexible data warehouse infrastructures. Instead, new information management principles such as data lake [44, 47] MAD [12] arise, that aim to easily ingest, transform, and analyze data in an exploratory and agile manner. In addition, information needs are often ad-hoc or situational [42], or require the use of hetero-

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

geneous or unstructured data that are not integrated in a data warehouse. An instant integration of Big Data is not even desirable, as the future use cases of the data is not known. Implementing the data lake principle allows to store the mentioned variety of data sources. However, while a new wealth of data is available, the integration of a large variety of sources is still a complicated, laborious and mostly manual process that has to be performed by experts and that is required before queries on the combined data can be issued. This limits the ability of non-expert users to include more data into their analysis tasks in exploratory manner. Without additional tool support, the effort of data integration will most likely prevent those users from taking advantage of the wealth of Big Data today.

To illustrate the problems with today's data management tool let us consider the following scenario: Imagine you are a working in the marketing department of a company and you need to select customers that should be targeted by a specific campaign. To achieve this you have to group your customers according to different properties of the their home countries. While the customer master data is part of your enterprise data warehouse detailed country information, e.g. population, life expectancy, GDP, debt, etc., is missing. To get this data you need to identify relevant data sources manually, for example through a regular search engine. Then, the data has to be extracted and cleaned, i.e., converted into a form that is usable in a regular database. In a next step, it needs to be integrated into an existing database, which includes mapping of corresponding concepts, but may also include instance-level transformations. Only after this process is finished, the original query can be issued. Still, the result may not be what the user originally desired, or the user may want to see the query result when using a different Web data source. In this case, another iteration of the process is necessary. The overall process is extremely cumbersome and you will be likely miss a large fraction of all relevant data sources that are available on the Web [37].

To solve this problem we propose a novel method for *Top-k Entity Augmentation* (Sect. 2), that enables us to enrich a given set of entities with additional attributes based on a large corpus of heterogeneous data. This allows for ad-hoc data search and integration in Big Data environments with large collections of heterogeneous data such as data lakes or publicly available data. We extend this approach to an *Exploratory Ad-Hoc Analytics System* called DrillBeyond (Sect. 3), where we incorporate entity augmentation into traditional relational DBMS, to enable their efficient use in analytical query contexts. This enables users to issue analytical ad-hoc queries on a database while referencing data to be integrated as if it were already defined in the database, without providing specific sources or mappings. In case of our marketing scenario the country properties would be seamlessly integrated during query execution and instantly used to answer the user request.

In the remainder of this section, we will derive requirements (Sect. 1.1) for a novel data management architecture focused on ad-hoc integration of heterogeneous data sources with traditional data management systems. Given that, we will propose an architectural blueprint (Sect. 1.2) that also serves as the outline for the rest of the chapter.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

Parts of the material presented in this chapter have already been published in [22] and [21].

## *1.1 Requirements*

To put it succinctly, we identify two trends: first, we observe an ever increasing amount and diversity of available data sources, both inside organizations and outside. Organizations are stockpiling data in a quantity and of a variety not seen in the past. At the same time, developments such as the Open Data trend and sophisticated technologies for Web data extraction lead to higher availability of public data sources. As a consequence, there is an increasing demand to enhance and enrich data by integrating it with external data sources which introduces additional complexity. Second, there is a *growing demand for exploratory analytics and ad-hoc integration*, driven by the broadening spectrum of data users. This demands that the tools and processes of data integration become simpler, and able to cater to a larger audience. While data processing and analysis traditionally were the domain of IT departments and BI specialists, more and more users from other departments are becoming active data users and require easy access to all valuable data sources, both inside and outside organizational boundaries. In addition, the focus of data use has broadened: Traditional BI processes focused on defined data flows, typically from source systems to a data warehouse, and aimed at producing well-defined reports or dashboards. However, a new type of investigative data analysis, often associated with the role of the *data scientist*, increases the demand for ad-hoc integration of the variety of data sources mentioned above.

Still, even with all these new sources for data, the focus of most of the analysis tasks is on the respective core data of each organization. This most valuable data will still be stored in controlled data warehouse environments with defined schemata. Therefore, new ad-hoc data integration techniques need to take this analytics context into account. From these observations, we derive requirements for a novel data management architecture focused on fulfilling the need for exploratory analytics.

*Exploratory Integration and Analytics* The novel data sources discussed above do not lend themselves to classical data integration with centralized schemata and expert-defined mappings and transformations. Data from Web and Open sources, as well as from heterogeneous management platforms such as data lakes, should be integrated dynamically at the time it is needed by a user. The volume and variety of data sources in these scenarios makes single, centralized integration effort infeasible. Instead of consolidating all available sources as soon as they are available, sources should be retrieved and integrated based on a specific analytical need. In such a scenario, data search takes the place of exhaustive schemata covering all available information. We therefore propose methods and systems to enable exploratory analysis over large numbers of heterogeneous sources.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

*Minimal Up-front User Effort* When developing methods for answering ad-hoc information needs, an important factor is the minimization of up-front costs for the user. In other words, our ambition is to minimize the user effort necessary before first results become available. Specifically, we want to reduce the reliance on IT personnel and integration experts, or the need to search data repositories or master integration tools before external sources can be included in a data analysis scenario. In the ideal case, a user should be able to declaratively specify the missing data and be presented with workable answers automatically. To facilitate this, we propose keyword queries that allow to specify the users' information need and which can be iteratively refined during an analytical session. While this approach minimizes the user effort it also introduces ambiguities that need to be resolved by the underlying retrieval system.

*Trustworthy Automated Integration* The two goals introduced above amount to reducing user effort and time-to-result in data search and integration. As a consequence, we propose novel methods for automating these highly involved processes. However, exact data integration has been called an "AI-complete" problem [30], and is generally considered not to be automatically solvable in the general case. In fact, all proposed methods return results with varying confidence values, instead of perfect answers, requiring human validation in many cases. We therefore introduce methods that automatize the data search and integration processes as much as possible, while also facilitating user understanding and verification of the produced results.

*System Integration* Finally, ad-hoc integration queries should not introduce an isolated, new class of systems into existing data management architectures. As discussed above, the wealth of Big Data collected by organizations or found on the Web is very promising. However, most analytical tasks will still focus on the core databases inside

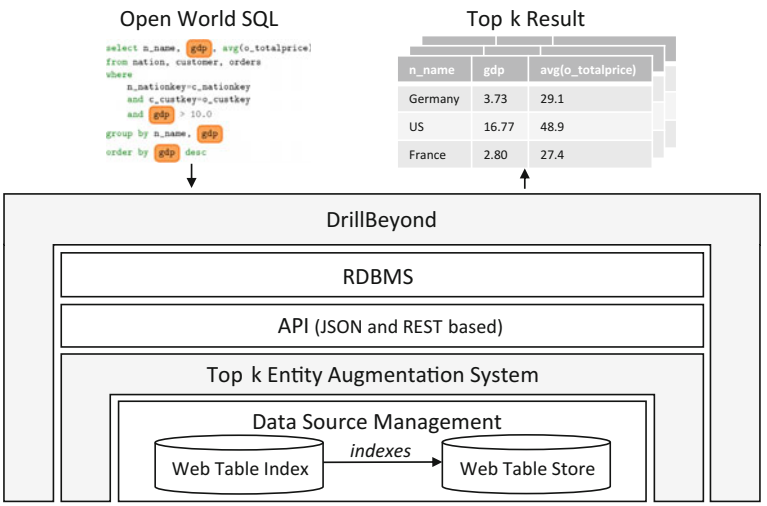



**Fig. 2** Architecture overview of a combined database and information retrieval engine

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

organizations, with ad-hoc integrated sources supplementing, not replacing, them. Therefore, methods developed to support ad-hoc integration can not be deployed in a vacuum, but need to work hand in hand with systems managing core data. In detail, we propose an RDBMS/IR system hybrid system that allows querying and analyzing a regular relational database while using additional attributes for which the values could be found only in external data sources (Fig. 2).

## *1.2 Architecture Overview*

In the following, we provide an architectural blueprint of a combined database and information retrieval engine in order to perform powerful queries over DBMS and heterogeneous data sources in an efficient, easy-to-use and seamless manner that fulfills the requirements outlined in Sect. 1.1. Our proposed architecture consists of a series of layers providing increasingly higher-level services. We assume a large collection of external tabular structured data, e.g. a corpus of Web tables, i.e., data-carrying tables extracted from the open Web, datasets published on an Open Data platform, or spreadsheets part of a corporate *data lake* [44]. Without limiting the generality of the proposed system we utilize Web tables for the remainder of this chapter, since they are freely available such as Dresden Web Table Corpus (DWTC) [19], that consists of 125M Web tables extracted from a public Web crawl. This corpus is indexed by an industry-standard document index server such as Solr[1] or Elastic-Search.[2]

On top of the inverted index we propose an *Entity Augmentation System* (EAS) that forms the basic building block for ad-hoc data integration. EAS's aim at extending a given set of entities with an additional, user-requested attribute that is not yet defined for them. In Sect. 2, we will present a novel approach to the problem which is especially suited for analytical uses cases. It is based on an extended version of the set cover problem, called *top-k consistent set covering* for which we introduce several algorithms. The EAS consists of a *Data Source Management System* providing storage and indexing facilities for Web tables, enabling the higher layers to retrieve raw Web tables based on keyword matches in data, schema or other metadata. Further it orchestrates several schema and instance matching systems, knowledge repositories and ranking schemes to create a candidate dataset D given an augmentation query. Finally, it includes a JSON-based REST API, which enables other systems to easily integrate with it and pose entity augmentation queries.

On top of that, we propose an RDBMS/IR system hybrid system called *DrillBeyond* (see Sect. 3), that allows querying and analyzing a regular relational database while using additional attributes not defined in said database. Therefore, we propose a new type of database queries, which we denote as *Open World Queries*. In short, in an Open World SQL, the user is allowed to reference arbitrary additional attributes

[1] http://lucene.apache.org/solr/.

[2] https://www.elastic.co/.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

not defined in this database. An exemplary query is shown in Fig. 8. The goal is to enable users to specify information needs requiring external data declaratively, just as if only local data was used, without having to integrate data up-front. Leveraging the *Entity Augmentation System* a database can be augmented at query time with Web data sources providing those attributes. The system will not respond to Open World Queries with a single, perfect result, as it would be the case with normal database queries. Instead, it should produce a ranked list of possible answers, each based on different possible data sources and query interpretations. The users can then pick the result most suitable to their information need. To this end, our system tightly integrates regular relational processing with new data retrieval and integration operators that encapsulate our novel augmentation techniques. In Sect. 3 we describe the challenges in processing this new query type, such as *efficient processing of multi-result SQL queries*, and present how our novel DrillBeyond system solves them.

## 2 A Top-K Entity Augmentation System

In the previous section, we discussed the need to support ad-hoc information needs in analytical scenarios with tools for exploratory data search and lightweight integration. Specifically, we want to enable a user working on a dataset to effortlessly retrieve and integrate other relevant datasets from a large pool of heterogeneous sources. One compelling type of query in this context are so-called *entity augmentation queries*, or *EAQ*. These queries are defined by a set of entities, for example a set of companies, and an attribute that has been undefined so far for these entities, for example "revenue". The result of the query should then associate each of the entities with a value for the queried attribute, by automatically retrieving and integrating data sources that can provide them. We will call this attribute the *augmentation* attribute, and the system processing such queries Entity Augmentation System, or *EAS*. The user has to specify the augmented attribute just by a keyword, while the EAS decides on how to lookup data sources, how to match them to the existing data, and possibly how to merge multiple candidate sources into one result. This makes entity augmentation queries both powerful and user-friendly, and thus interesting for exploratory data analysis. Effectively, an EAS aims at automating the process of data search, as well as the following integration step.

In principle, any type of data source could be used for answering entity augmentation queries. For example, a large ontology such as YAGO [52] could answer some EAQ rather easily, though only if the augmentation attribute is defined in this knowledge base. In recent related work, several systems that process such queries on the basis of large Web table corpora have been proposed, for example *InfoGather* [56, 57], the *WWT* system [48, 50], and the *Mannheim Search Join Engine* [37]. An advantage of methods using tables extracted from the Web is that they offer more long tail information, and do not rely on the queried attribute being defined in a central knowledge base. Methods based on Web tables as their data source can, in

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

principle, also be used with any other large collection of heterogeneous data sources. The techniques for automatic data search and integration introduced there could be applied to enterprise *data lakes* [44, 47] as well. In fact, many challenges, such as identifying relevant datasets in light of missing or generic attribute names, bridging semantic or structural differences, or eliminating spurious matches, have already been tackled by existing augmentation systems, which we review in Sect. 2.4.

Though solving these fundamental issues of data integration remains the most important factor for the success of an EAS, we argue that several other challenges are still unanswered in entity augmentation research. These challenges, are discussed in Sect. 2.1 and used to derive design requirements for a novel entity augmentation method in Sect. 2.2. Next, in Sect. 2.3, we will map the entity augmentation problem to an extended version of the *Set Cover* problem, which we call *Top-k Consistent Set Cover* and provide basic Greedy algorithm that solves this problem. Finally, we will survey related work in Sect. 2.4.

Parts of the material presented in this chapter have already been published in [22].

## *2.1 Motivation and Challenges*

In this section, we will discuss an exemplary entity augmentation scenario based on a heterogeneous table corpus with partially overlapping sources. Our example scenario is depicted in Fig. 3, with the query represented as a table on the top, and the available candidate data sources below it. The query table consists of a set of five companies, and the augmentation attribute "*revenue*". The candidate data sources depicted below the query table vary in coverage of the query domain, the exact attribute they provide, and their context. They are further annotated with their relevance with respect to the query, which is depicted as a numeric score on each source. In [22] we provide detailed scores and similarity measurements to compute the overall relevance score of the candidate data sources.

As an introductory example let us assume an algorithm that picks, for every queried entity, the respective value from the source with the highest relevance score. In our example, this naïve algorithm would pick values from the sources $S7$ for "Rogers", $S8$ for "AT & T", $S5$ for "Bank of China" and "China Mobile", and finally $S3$ for "Banco do Brasil", using the highest ranked available source for each queried entity. This means that the algorithm picks a large number of data sources, almost one distinct source for each entity. More sophisticated methods for pruning candidate tables and picking values from the remaining candidates can be used, such as correlating or clustering sources, mapping sources to knowledge bases, or quality-weighted majority voting (again, see Sect. 2.4). However, these methods do not fundamentally change the fact that the integration is performed on a by-entity basis, which leads to a large number of distinct sources being used to construct the result. It has been argued that in data integration, and especially selection of sources to integrate, "less is more" [18]. Choosing too many sources not only increases integration cost, but may even deteriorate result quality if low quality sources are added. For example,

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

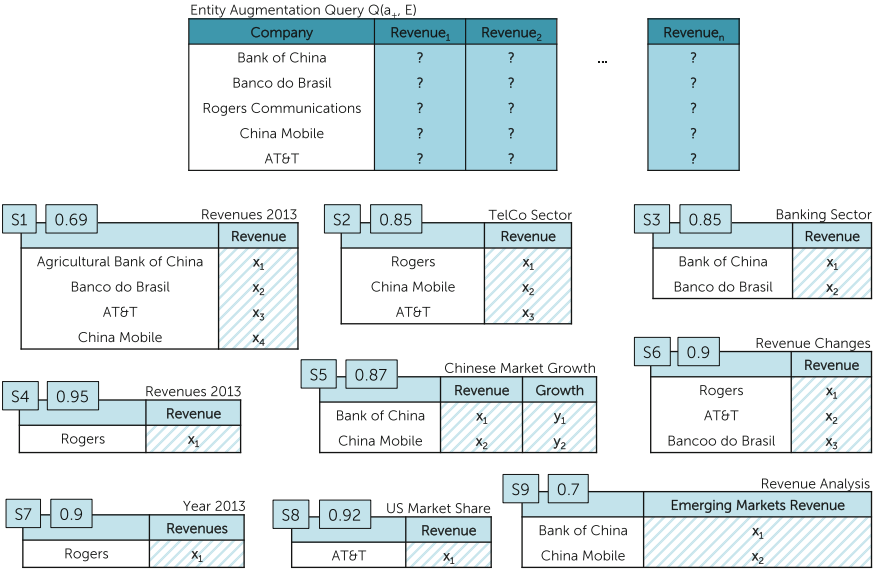



**Fig. 3** Example augmentation scenario: query table and candidate data sources

adding $S8$ to the example result made the result inconsistent: In contrast to the other sources, this one is, in fact, concerned with US revenue only. This is just one intuitive example of a problem introduced by inadequate source selection and combination in the augmentation process. In the following, we will identify specific challenges that are insufficiently solved with existing by-entity fusion methods.

*Trust and Lineage* Our first argument is concerned with the usability of an entity augmentation result. In many application domains, the user can not blindly trust an automatic integration result, no matter how sophisticated the method used is. Rather, the result will serve as a starting point in a semi-automatic integration process, where the user will manually check the sources proposed by the system, correct errors made by the automated integration techniques, and even switch some of the sources for others. Choosing a large number of sources therefore increases the users' verification effort. We argue that existing fuse-by-entity models diminish *trust* and hinder understanding of data *lineage*, two properties that are important in the overall process of working with data, because the number of distinct sources they fuse is not considered in source selection. In this chapter, we therefore investigate methods that produce not only consistent augmentation results from several sources, but *minimal* augmentations, i.e., augmentations that use a minimal number of sources to facilitate the usage of the result. In the running example, such a result would be $S2$, $S3$, as it only uses two sources to augment all entities, even though the sources' average score is slightly worse than the score of the naïve solution introduced above. To summarize, when coercing a large number of data sources into one result, properties

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

important for data analysis such as transparency, lineage and trustworthiness of the result are diminished.

*Attribute Variations* Another problem that we identified with related work is the underlying assumption of a single true value for any combination of entity and augmentation attribute. This single-truth idea, however, does not reflect the complex realities. For example, the augmentation attribute in our scenario was given simply as "revenue". However, the concept is actually more complex, with many variants such as "US revenue" or "emerging markets revenue" ($S8$ and $S9$) and derived attributes such as "revenue growth" ($S5$ and $S6$). Furthermore, many types of values come with temporal or spatial constraints, such as different years of validity, or may have been measured in different ways, for example using different currencies. Therefore, we argue that even when only sources of high relevance are picked, they may be correct and of high quality when considered on their own, but still do not form a consistent result for the overall query domain when combined. The differences in precise semantics can, in most cases, not be decided based on the extracted attribute values, but on the level of data sources, for example by considering the context of a table. Even though better methods for creating consistent augmentations for some important dimensions such as time and unit of measurement have been proposed [50, 57], source consistency is not considered as a general dimension in entity augmentation so far. To summarize the argument: due to the existence of subtle *attribute variations*, the notion of source consistency needs to be taken into account when combining several sources to a single augmentation result.

*Unclear User Intent* Extending our argument based on the intricacies of attribute variations, we will discuss an additional challenge: the problem of *unclear user intent*. In entity augmentation queries, the information need of the user is relatively underspecified, especially when compared to queries on a database with a defined schema. Even though entity augmentation also operates on structured data, for example on a large-scale Web table corpus, the user is still forced to pose queries on data whose extent or schema is unknown to him or her. Forming precise queries may therefore not always be possible, especially in the presence of attribute variations. In turn, the entity augmentation system may not always be able to pinpoint the exact attribute the user is interested in. To give one example, in the scenario in Fig. 3 it is unclear whether the user is interested in any specific year, or just in the highest ranked sources. In this example, a solution based on the sources $S1$, $S4$ may be more useful than the higher-ranked solution $S2$, $S3$ proposed above, because both sources explicitly state a certain year of validity. However, which solution the user would prefer can not be decided from the query alone.

*Exploratory Search* Even if the user can specify a precise query, in the case of situational or ad-hoc analysis, they may not even yet know which attribute variant is the most relevant. In those situations, the underspecified nature of entity augmentation queries may even be turned into an advantage. For example, a user may want to stumble over new aspects of his or her information need, similarly to the way it may happen with a Web search engine. In the example scenario, the user may have

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

queried only for "revenue", but a solution showing "revenue growth" based on sources $S5$, $S6$ may, in some situations, give the ongoing analysis process a new perspective or direction. However, such serendipitous query answering is not supported with current augmentation systems. One partial exception is the Mannheim Search Engine [37], which allows so-called unconstrained queries, in which the system returns augmentations for all attributes it can retrieve from all available sources. However, this may leave the user with an unfocused, large and hard-to-comprehend result that is not connected to the information need at hand. In other words, the *exploratory* nature of entity augmentation queries is not done justice in current approaches.

*Error Tolerance* Finally, we note that all existing augmentation systems are based on techniques for automated schema matching, entity resolution and automated relevance estimation. All these components by themselves have a certain margin of error that, no matter how sophisticated the underlying methods become, can never be fully eliminated. In combining these systems to higher-level service such as entity augmentation, the individual errors will even multiply. Still, none of the existing systems, while of course striving to optimize their precision in various ways, offer explicit support for *tolerating* possible errors.

## 2.2 Requirements

Having introduced and discussed five significant open challenges in entity augmentation, we now want to discuss *requirements* for a novel entity augmentation method that alleviates these challenges. First, let us discuss the challenges *lineage* and *attribute variants*. We already discussed that using a large number of sources impedes the user's understanding. Further, we discussed that, to detect and correctly exploit the existing variants of the queried attribute, we need to take consistency between sources into account. Therefore, we investigate methods that produce both augmentation results that are both *consistent* and *minimal*, i.e., augmentations that use a minimal number of distinct sources, with each of them representing the same attribute variant. To determine this consistency of datasets, it is possible to measure the similarity between their attribute names, compare global properties of their value sets, or analyze the associated metadata such as dataset titles and context. We will discuss our notion of consistency in more detail and also give formal definitions in the following sections.

Now let us consider two further challenges: *unclear user intent* and *error tolerance*. Both of these challenges result from various forms of uncertainty: The first from uncertainty about the user intent, the second from uncertainty of the utilized basic methods such as schema matching and entity resolution, and the third from uncertainty in the sources. Information retrieval systems solve this problem by presenting not one exact, but a top-k list of possible results. For example, errors in relevance estimation are tolerable, as long as some relevant documents are included in the top-k result list. Unclear user intent or ambiguities in the query keywords can

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

be resolved by returning documents based on multiple interpretations of the query, instead of focusing exclusively on the most likely interpretation [3]. Furthermore, the challenge of *exploratory search* can also be solved in a top-k setting by means of result list diversification, as is common practice in Web search [5] or recommender systems [58]. We argue that for entity augmentation a similar argument can be made: It is advantageous to provide not only one solution, but allow the user to choose from a ranked list of alternative solutions. In other words, we aim at extending entity augmentation to *diversified top-k entity augmentation*.

Let us reconsider our running example shown in Fig. 3. Instead of returning only one result based on $S2$, $S3$ as discussed above, one alternative would be a result based on $S1$, $S4$. It has a worse average score, but has clearly marked year in both sources, which may be more useful for the user on manual inspection, because of the clearly marked year information in the context.

Another aspect are attribute variations, which, due to the exploratory nature of entity augmentation queries, may also be of interest to the user. An example would be a third result based on $S6$ and the second column of $S5$ which represents changes in revenue instead of absolute revenue. Yet another exploratory result could be comprised of just $S9$, which does not cover all entities, but might give the users' analysis a new direction.

Note however, that we want to generate solutions that are real alternatives, such as the three examples above. Because of copying effects on the Web [13, 38], using only the most relevant sources for creating all $k$ augmentation results however would lead to many answers being created from structurally and semantically similar sources. Furthermore, because we fuse results from several sources, naïve solutions would use the same, most relevant sources multiple times in various combinations, leading to *redundancy* in the result list. In the example, one such redundant result would be $S1$, $S7$. However, since $S7$ differs only superficially from source $S4$, the results $S1$, $S4$ and $S1$, $S7$ are very similar. We want to avoid slight variations of one solution as this would add little information to the top-k result. A meaningful top-k list needs to consider a *diverse* set of sources, exploring the space of available data sources while still creating correct and consistent answers. This has been recognized a long time ago in information retrieval [10], but has recently been explored for structured queries [14, 32], and even for Web table search [46], which is highly related to entity augmentation. We will define the notion of result diversity for our specific problem, as well as our means to achieve it, in the following sections.

In a nutshell, we can derive the following two goals: We aim at providing a *diversified top-k* list of alternative results, which are composed of *consistent and minimal* individual results.

## 2.3 Top-k Consistent Entity Augmentation

In this section we will describe our novel method of *Top-k Consistent Entity Augmentation*. Initially, we will formalize augmentation queries and the optimization

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

objectives of our method in Sect. 2.3 and introduce top-k consistent *set covering* as an abstract framework for solving our problem in Sect. 2.3.

**Entity Augmentation Queries** We will now formalize our notion of Top-k Consistent Entity Augmentation, and introduce the optimization objectives that we aim at. First, consider a general entity augmentation query definition.

**Definition 1** *(Entity Augmentation Query)* Let $E(a_1, \ldots_a n)$ denote a set of entities with attributes $a_1, \ldots, a_n$, and $a_+$ denote the augmentation attribute requested by the user. Then, an augmentation query is of the form:

$$Q_{EA}(a_+, E(a_1, \ldots, a_n)) = E(a_1, \ldots, a_n, a_+)$$

In other words, the result of such a query is exactly the set of input entities with the augmented attribute added. To create this new set of entities, the EAS has to retrieve values for attribute $a_+$ for each entity $e \in E$, which we will denote $v_e$. These values will be retrieved from a corpus of data sources $\mathcal{D}$ managed by the EAS, from which it selects a relevant subset $D$ for each query $Q_{EA}$.

Sources $d \in D$ can provide $a_+$ values for some subset of $E$, denoted $\text{cov}(d)$, i.e., they cover $E$ only partially in the general case. Individual values provided by a source $d$ for an entity $e$ are denoted $d[e]$ with $e \in \text{cov}(d)$. Given a heterogeneous corpus of overlapping data sources, an augmentation system will potentially retrieve multiple values for an entity $e$, the set of which we will denote by $V_e = \bigcup_{d_i \in D} d_i[e]$. Finally, the EAS assigns each data source a relevance score $rel : D \rightarrow [0, 1]$ with respect to the query. To determine this relevance score, various measures can be combined. Examples include the similarity between the queried attribute name and the respective attribute's name in the data source, or the quality of the match between the queried instances and those found in the data source. In addition, global measures for the quality, such as the PageRank of the source page, can be integrated.

As we described in Sect. 2.1, most systems from literature assume that they can reconcile the set of values $V_e$ into a single correct augmentation value $v_e$ for each entity. An example for such a fusion method would be majority voting, or clustering values and then picking a representative from the highest ranked cluster as in [37, 56].

As motivated in Sect. 2.2, our notion of an entity augmentation query differs in several aspects: First, instead of individual values, it picks subsets of sources that cover $E$, and second, it returns an ordered list of alternative solutions. In other words, its basic result is a list of top-k alternative *selections of sources*.

**Definition 2** *(Top-k Source Selections)* Given a set $D$ of relevant data sources, and a number $k$ of augmentations to create, a top-k Source Selection is defined as:

$$Q_{EA}(a_+, E, k) = [c_1, \ \ldots, \ c_k \mid c_i \subset D \wedge \text{cov}(c_i) = E] \tag{1}$$

We call one such set $c_i$ a *cover* or an *augmentation*, and the list of these augmentations the query result.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

**Fig. 4** MaxRelevance

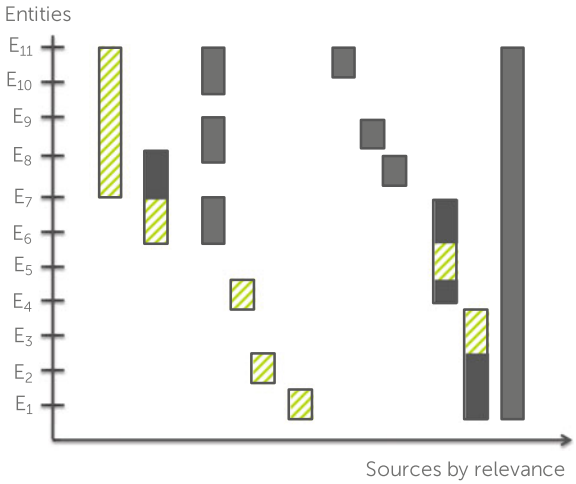
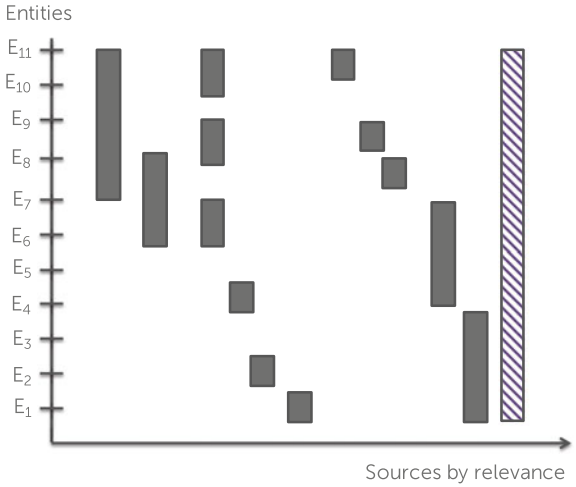



**Fig. 5** MinSources




**Definition 3** *(Cover/Augmentation)* A cover is an ordered subset of $D$ that covers $E$, i.e., $c = [d_i, \ldots, d_x]$ with $\bigcup_{d \in c} \text{cov}(d) = E$. If multiple data sources provide values for a distinct $e$, i.e., if $\exists_e (e \in \text{cov}(d_i) \cap \text{cov}(d_j))$, the augmented value for $e$ is decided by the order of the sources in $c$, i.e., $v_e = d_i[e]$ with $i = min(\{i \mid e \in d_i \wedge d_i \in c\})$.

As discussed in Sect. 2.2, the aim is to enable the user to choose the most promising augmentation from the ranked list of alternatives. This leads to the question of how to construct these subsets $c_i \subset D$, in order to create a valuable list of alternatives for the user. We will now introduce the individual dimensions of this problem, *relevance*, *minimality*, *consistency* and *diversity*, discussing exemplary baseline strategies that optimize for one of each dimension.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

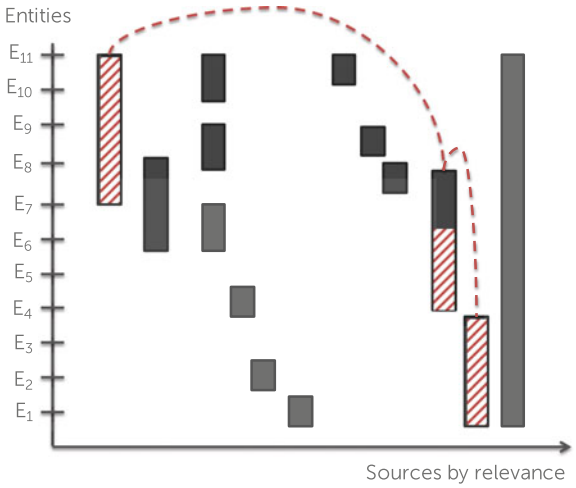



**Fig. 6** MaxConsistency

*Relevance* One naïve baseline strategy, which we call *MaxRelevance*, is depicted in Fig. 4. Starting from the highest ranked data source, it picks all the values it provides for entities that do not have a value yet, then continues to the next most relevant source, according to the relevance function introduced above. While this strategy obviously maximizes the average relevance of the created augmentation, a large number of distinct sources might be picked. This makes it harder for the user to understand the query result and assess its quality, and also has a high chance of producing inconsistencies between sources.

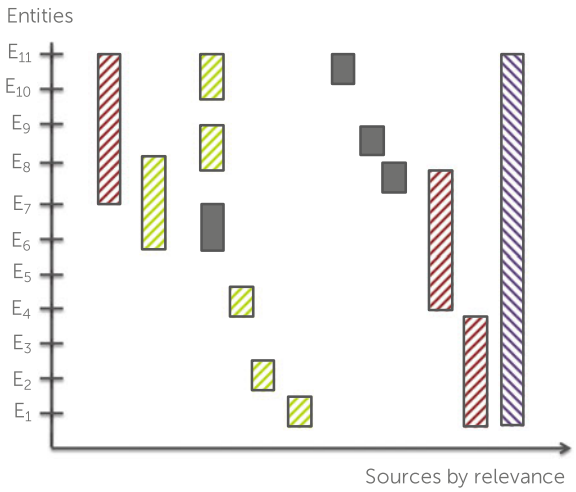
*Minimality* A naïve approach to solve the latter problem would be to prioritize data sources with large coverage of the queried entities. This strategy, called *MinSources*, is illustrated in Fig. 5. As is illustrated in this particular example, while solutions created this way use a minimal number of distinct sources, the other objectives, such as relevance, can be arbitrarily bad.

*Consistency* Next, consider the strategy *MaxConsistency*, in which sources are chosen based on a measure of source similarity, i.e., a function $\text{sim} : D \times D \to [0, 1]$, which is depicted using dashed arrows in Fig. 6. This function captures our notion of attribute variant consistency as discussed in Sect. 2.1. Utilizing such a function to guide source selection will increase the overall consistency of the created augmentations, but will create augmentations that are not necessarily minimal nor highly relevant. It is calculated from measures such as the similarity between the two data source's attribute names, their value sets, and by comparing the associated metadata such as dataset titles and context.

*Diversity* In addition, we will have to devise a method that is able to create multiple meaningful alternative solutions. A naïve solution would be to create one cover $c$ from sources $D$, and then iteratively set $D' = D \setminus c$ and create the next cover from $D'$. This approach, called *NoReuse*, is illustrated in Fig. 7. It has two problems:

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

**Fig. 7** NoReuse




Firstly, each data source can be used in only one alternative, even though several combinations of good sources might be possible. Secondly, just prohibiting reuse of specific datasets does not necessarily lead to diversified solutions, as there may be data sources so that $\exists_{d_i,d_j}|d_i \neq d_j \wedge \text{sim}(d_i, d_j) \approx 1.0$. This occurs, for example, due to frequent copying-effects on the Web [13, 38]. Since we aim at minimizing the pairwise similarity of covers in the query result, we introduce a similarity function $\text{sim} : D \times D \rightarrow [0, 1]$ that compares covers instead of data sources. This lifts the similarity function to the domain of covers $\text{sim}_{\mathcal{A}} : C \times C \rightarrow [0, 1]$, whereas the aggregation function $\mathcal{A}$ can be an *average* or *max*.

We consider these four dimensions to be the decisive factors for a useful top-k entity augmentation result. What we therefore need, is a strategy that creates complete covers, while jointly optimizing all mentioned objectives.

**Definition 4** *(Top-k Consistent Entity Augmentation)* A top-k Consistent Entity Augmentation query produces a top-k Source Selection (Definition 2) that is optimized with respect to the relevance, minimality, consistency, and diversity objectives.

In the next section, we will introduce our algorithmic approach to processing top-k Consistent Entity Augmentation queries.

**Ranked Consistent Set Covering** We propose a new approach for constructing entity set augmentations by modeling it as an extended form of the *Weighted Set Cover Problem*, one of Karp's original 21 NP complete problems [34].

**Definition 5** *(Weighted Set Cover)* Given a universe of elements $U$ and a family of subsets of this universe $S$, each associated with a weight $w_i$, the Weighted Set Cover problem is to find a subset $s \subset S$ with $\bigcup_s = U$, such that $\sum_{i \in s} w_i$ is minimized.

Intuitively speaking, the aim is to *cover* all elements in $U$ using sets from $S$ with minimal cost. In our problem domain, the algorithm input consists of a set of entities

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

$E$ that are to be augmented, corresponding to $U$ in the original problem, and a set of data sources $D = \{d_1, \ldots d_n\}$, as retrieved and matched by the underlying entity augmentation system, corresponding to $S$. The relevance score assigned to each datasource by $\text{rel}(d)$ is used in place of the weights $w$.

So far, we could trivially map our problem to the well known Set Cover problem. Specifically, the *Relevance* and *Minimality* objectives defined in section "Entity Augmentation Queries" correspond closely to the objective $\sum_{i \in s} w_i$ in the set cover problem. Still, there are some crucial differences: In contrast to the original problem, where only a *single* minimal cover is required, the output we aim for is a ranked list of covers, denoted $C = [c_1, \ldots, c_n]$. Furthermore, as illustrated in section "Entity Augmentation Queries", the entity augmentation use case does not only require small covers with high individual relevance, but *consistent* covers, as defined in the *consistency* objective. And lastly, we also introduced the *diversity* objective, i.e., the covers should not consist of the same or similar datasets throughout the top-k list, but be complementary alternatives.

We will now incrementally develop our proposed algorithms for top-k consistent set covering. We start from the well known greedy algorithm for the Weighted Set Cover problem, which, given a universe $U$, a set of sets $S$ with weights $w$, and a set of yet uncovered elements $F$, iteratively picks the set:

$$\underset{S_i \in S}{\text{argmin}} \frac{w_i}{|S_i \cap F|} \qquad \textit{(Greedy Set Cover Algorithm Step)}$$

The algorithm chooses sets $S_i$ until $F = \emptyset$, at which point a complete cover has been formed. Although the greedy algorithm does not create optimal covers, it is still the most frequently employed algorithm for the set covering problem. In fact, it has been shown that the algorithm, achieving an approximation ratio of $H(s') = \sum_{k=1}^{n} \frac{1}{k}$ is essentially the best possible polynomial-time approximation algorithm for the set cover problem.

*Coverage and Relevance.* We therefore also initially base our algorithm on the greedy set covering algorithm. With an initially empty cover $c$ and a free entity set $F = E$, we can use the original greedy Set Cover algorithm to produce an ordered subset of $D$, by picking in each iteration the dataset $d$ that maximizes:

$$\underset{d_i \in D}{\text{argmax}} \, \text{rel}(d_i) \cdot |\text{cov}(d_i) \cap F| \tag{2}$$

until $F = \emptyset$. Note that we maximize scores instead of minimizing weights as this is more intuitive for the problem domain.

An augmentation constructed in this way would roughly correspond to a middle ground strategy between the *MaxRelevance* and *MinSources* strategies discussed in section "Entity Augmentation Queries". This implies, however, that it can potentially create augmentations from very heterogeneous data sources.

*Cover Consistency*. To counteract this effect, we explicitly model consistency between the datasets that make up a cover. We utilize the similarity function

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

between datasets $\text{sim} : D \times D \rightarrow [0, 1]$, as defined in section "Entity Augmentation Queries", which models the consistency between data sources. Given an initially empty cover $c$ and an aggregation function $\mathcal{A}$ such as *average* or *max*, we can greedily generate covers using consistent datasets by picking in each iteration the dataset $d$ that maximizes:

$$\underset{d \in D}{\text{argmax}}\, \text{rel}(d) \cdot |\text{cov}(d) \cap F| \cdot \text{sim}_{\mathcal{A}}(d, c) \tag{3}$$

This means we encourage picks of data sources that are similar to data sources that were already picked for the current cover. We assume as a special case that $\text{sim}_{\mathcal{A}}(d_i, \emptyset) = 1$, which implies that the first data source chosen will be the same as in regular set covering. Subsequent choices on the other hand will be influenced by already selected sources. This also implies that datasets with a low relevance or coverage, that are not picked initially, may still be chosen in a later iteration, if they fit well with those chosen so far. Since we require $|\text{cov}(d) \cap F|$ to be greater than zero, the algorithm will still make progress with every step, as only datasets that provide at least one new value can be selected.

Using objective function (3), the algorithm picks datasets to create covers that are not only highly relevant to the query, but also fit well together according to $\text{sim} : D \times D$.

However, using only this objective function, there is still no intuitive way of creating useful top-k augmentations. The naïve approach re-running the same algorithm with $D \setminus c$ as the set of candidate data sources would not lead to useful alternative solutions, as discussed in Sect. 2.3.

*Top-k Results and Diversity.* Let $C$ denote the set of previously created covers, with $|C| \geq 1$. This set could be initialized with a single cover created, for example, using the greedy algorithm and objective function (3). Our core idea is to perform consecutive runs of the greedy algorithm using the same input datasets, with each run placing greater emphasis on datasets that are dissimilar to datasets picked in previous iterations, i.e., dissimilar to datasets in $\bigcup_C$. Implementing this idea naïvely however, for example by dividing function (3) by $\sum_{d_i \in \bigcup_C} \text{sim}(d, d_i)$ does not yield the expected results. While the second iteration might then choose datasets from a different part of the similarity space than the first iteration, the term becomes increasingly meaningless with more iterations as $\bigcup_C$ grows. This is because newly picked datasets are compared to a larger and larger subset of the candidate set $D$, leading to an increasingly uniform value for $\sum_{d_i \in \bigcup_C} \text{sim}(d, d_i)$.

Instead, we introduce a more complex dissimilarity metric based on individual entities in $E$ and the datasets that were used to cover them in previous iterations. We define a function $coveredBy(e, C)$ which yields the datasets that were used to augment the entity $e$ in covers $C$ created in previous iterations.

$$coveredBy(e, C) = \{d \mid \exists_c \in C : d \in c \wedge e \in \text{cov}(d)\} \tag{4}$$

We can then define our final scoring function as

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

**Algorithm 1** Top-k consistent set covering: Greedy

**function** GREEDY- TOPK- COVERS($k$, $E$, $D$)
  $C \leftarrow \emptyset$
  $U \leftarrow \begin{pmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{pmatrix}_{|E| \times |D|}$ ▷ Usage matrix
  **while** $|C| < k$ **do**
    $c \leftarrow$ COVER($E$, $D$, $U$)
    **for all** $(e \rightarrow d) \in c$ **do** ▷ Update Usage Matrix
      $U[e, d] \leftarrow U[e, d] + 1$
    **if** $c \notin C$ **then** ▷ Remove duplicates
      $C \leftarrow c$
  **return** $C$

**function** COVER($E$, $D$, $U$)
  $c \leftarrow \emptyset$
  $F \leftarrow E$ ▷ Free set, uncovered entities
  **while** $|F| > 0$ **do**
    $d \leftarrow \text{argmax}_{d \in D} \frac{\text{rel}(d) \cdot |\text{cov}(d) \cap F| \cdot \text{sim}_{\mathcal{A}}(d,c)}{\text{REDUNDANCY}(d, D, F, U)}$
    **for all** $e \in F \cap \text{cov}(d)$ **do**
      $F \leftarrow F \setminus e$ ▷ Update free set
      $c \leftarrow c \cup (e \rightarrow d)$ ▷ Update cover
  **return** $c$

**function** REDUNDANCY($d$, $D$, $F$, $U$)
  $r, norm = 0, 0$
  **for all** $e \in F \cap \text{cov}(d)$ **do** ▷ Coverable by d
    $u \leftarrow U[e]$ ▷ Sources used to cover e
    $r \leftarrow r + \sum_{i=0}^{|u|} u[i] * \text{sim}(d, D[i])$
    $norm \leftarrow norm + \sum u$
  **return** $\frac{r}{norm}$

$$\underset{d \in D}{\text{argmax}} \frac{\text{rel}(d) \cdot |\text{cov}(d) \cap F| \cdot \text{sim}_{\mathcal{A}}(d, c)}{\text{redundancy}(d, F, C)} \tag{5}$$

where

$$redundancy(d, F, C) = \sum_{e \in F \cap \text{cov}(d)} \text{sim}_{\mathcal{A}}(d, \text{coveredBy}(e, C)) \tag{6}$$

In other words, we penalize picks that would cover entities with data sources that are similar to datasets that were already used to cover these entities in previous iterations. By penalizing similarity to previous covers, we avoid using the same similar datasets repeatedly for all covers in the top-k list, but we also do not strictly disallow the re-use of data sources in new combinations. Objective function (5) forms the core of our proposed entity augmentation algorithms, which we will introduce in the next sections.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

**Basic Greedy Algorithm and Extensions** With the scoring function in place, we can construct a greedy consistent set covering, shown in Algorithm 1, that produces consistent individual augmentations, as well as diversified solutions when run with $k > 1$. In Algorithm 1, the function *Greedy-TopK-Covers* produces $k$ covers by calling the function *Cover* $k$ times, while keeping an $|E| \times |D|$ matrix called $U$ as state between the calls. While the *Cover* function performs the basic greedy set cover algorithm with the objective function defined above, the main function updates the matrix $U$ after each iteration by increasing the entry for each entity/dataset combination that is part of the produced cover. The function *coveredBy* used in the *redundancy* term of objective function (5) is realized in the algorithm by summing up the matrix row values $U[e]$, which record the datasets used to cover $e$ in previous covers. Note that the main function also discards duplicate solutions, which may occur if the influence of the *redundancy* function is not strong enough to steer the search away from an already existing solution. Still, the matrix $U$ is updated even if a solution is rediscovered, so that further choices of the same data sources become more and more penalized, guiding the search into a different part of the solution space.

The greedy approach, while being easy to implement and fast to execute, will not necessarily construct the best possible list of solutions, as our evaluation in [22] shows. This is mainly due to exploring only a small part of the search space, i.e., considering only $k$ different covers. Therefore, we developed two further algorithms as extensions of the basic framework: the first one is based on the observation that the first $k$ solutions produced by Algorithm 1 may not necessarily be the best solutions. After the first solution has been produced the search is mainly guided by using different datasets for each solution, and thus new combinations of previously used data sets are often not considered in the basic greedy algorithm. One simple extension is called *Greedy**-algorithm, which uses the basic greedy algorithm to create more covers than requested, and then introduces a second phase to the query processing called *Select*, in which the $k$ best solutions are selected from a pool of $s \times k$ possible solutions, with $s$ being the scale factor. In comparison to the *Greedy* algorithm, the *Greedy** approach should find better solutions as it searches a larger portion of the search space, at the cost of a runtime that increases with the scale factor $s$, plus some overhead for the selection phase. However, the way it explores the solution space is relatively naïve. For this reason, we also developed a genetic approach which naturally fits to our problem as it intrinsically generates a pool of solutions from which $k$ can be picked, and both consistency and diversity of the results can be modeled intuitively. Specifically, consistency can be modeled as part of the fitness function, and diversity can be introduced through a suitable population replacement strategy.

A detailed evaluation and comparison of all three algorithms is provided by [22].

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

## 2.4 Related Work

*Entity Augmention* An early publication on Web table-based entity augmentation is [8], which is concerned with automating the search for relevant Web tables. The paper does not aim at fully automated table extension, but proposes a set of operators that are to be used in a semi-automatic process, enabling the user to search for tables, extract their context, but also to extend a found table with information from a related table. This last operator, called *extend*, corresponds to our notion of entity augmentation. The paper proposes an algorithm called *MultiJoin*, that attempts to find matching Web tables for each queried entity independently, and then clusters the tables found to return the cluster with the largest coverage. However, it does not try to construct consistent solutions, but returns the set of possible values for each entity.

A strongly related work is the InfoGather system [56], and its extension InfoGather+ [57]. The first system introduces Web table-based entity augmentation, as well as related operations such as attribute name-based table queries. InfoGather improved the state of the art especially by identifying more candidate tables than a naïve matching approach, while eliminating many spurious matches at the same time. They also introduce methods for efficiently computing the similarity graph between all indexed tables offline. InfoGather+ improves the system by tackling similar consistency issues as our work: it assigns labels for time and units of measurements to tables, and propagates these labels along the similarity graph described above to other tables where such labels can not be found directly. While InfoGather+ tackles the problem of producing more consistent results from various possible Web sources, it does not produce top-k results, or minimize the number of sources used.

The basic problem that there will be more than one correct answer for many augmentation queries, e.g., multiple *revenue* values for a single company because of different years of validity, is also explored in [50]. Specifically, the work targets quantity queries, i.e., queries for a numeric attribute of a certain entity. The earlier InfoGather+ already allows the user to specify a unit of measurement and a year-of-validity, and will only try to retrieve a single attribute value with these specific constraints. The QEWT system presented in [50], on the other hand, solves this problem by modeling the query answer as a probability distribution over the retrieved values, and then returning a ranked list of intervals as the final query answer. This work is similar in spirit to ours, in that it does not try to simplify complex real-world attributes into single values, but deals with the uncertainty of data explicitly.

In [37], a table augmentation system called *Mannheim SearchJoin Engine* is proposed that, in addition to entity augmentation given a specific attribute, also supports *unconstrained* queries, i.e., queries in which, given only a set of entities, all possible augmentation attributes are to be retrieved. Their method of dealing with multiple, possibly conflicting sources, by merging values using clustering and majority voting, is similar to [56].

In [45] a comprehensive system for so-called *transformation queries*, that largely correspond to entity augmentation queries, is envisioned. The paper's main con-

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

tribution is that it proposes a system, named *DataXFormer*, that includes multiple transformation subsystems, based on Web tables, wrapped Web forms, as well as crowdsourcing, although it does not give specific methods of combining the subsystems. Furthermore, it also relies on returning a single value for each queried attribute.

*Set Covering* The set covering problem as one of Karp's original 21 NP complete problems [34] implies the need for heuristic solutions and led to many optimization techniques. Our methods for generating top-k covers are inspired by multi-start optimization methods [43], such as GRASP [26] and Meta-RaPS [15], which have been applied to the set cover problem among others in [7, 35]. On a high level, these methods combine multiple iterations of a randomized *construction* phase and a local *improvement* phase. In the first phase, a solution is created using some heuristic, e.g., the Greedy approach, but applying some form of randomization. The randomization allows the algorithm to create slightly different results in multiple runs. This is achieved for example by randomly making non-optimal decisions in the individual steps of the respective algorithm.

A second inspiration for our top-k approach is *tabu search* [27], in which an initial solution is the starting point for multiple iterations of a neighborhood search aimed at improving the solution. The distinguishing feature is the *tabu list*, which stores already visited parts of the solution space. It is used to prohibit the algorithm from returning to an already visited part of the solution space, unless a so-called aspiration criterion is met, or a certain amount of iterations has passed. In this way, the search is prevented from getting stuck in, or repeatedly revisiting, the same local optima. Our approach tracks which entities have already been covered by which datasets in previous solutions by using a *usage matrix* (see Algorithm 1 in Sect. 2.3). Similarly to tabu lists, this matrix is then used to discourage the algorithm from making those choices again, preventing future iterations from creating similar covers. This was partly inspired by the concept of tabu lists, except that our approach does not prohibit certain choices, but only adjusts their weights.

**Fig. 8** Exemplary Open World SQL query, ad-hoc integrated attributes *gdp* and *creditRating* highlighted

```
select
    nation.creditRating,
    avg(o_totalprice)
from nation, customer, orders
where
    n_nationkey=c_nationkey
    and c_custkey=o_custkey
    and nation.gdp > 10.0
group by nation.creditRating
```

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

# 3 DrillBeyond – Processing Open World SQL

So far, we only studied top-k entity augmentation queries (see Sect. 2) in an isolated context, i.e. for a single table and simple attribute queries $Q_{EA}(E, a_+, k)$. However, it is natural to assume that ad-hoc data integration will be most useful in analytical Big Data scenarios, in which the user works with complex databases, and the augmentation query is only one step in a chain of analytical operations. We will exemplify this on a data analytics scenario illustrated in Fig. 8. There, we see a TPC-H query with two highlighted augmentation attributes "GDP" and "creditRating" that are not defined in the TPC-H schema, i.e. the query is not immediately processable in a traditional relational system with a closed schema. A possible way to approach the above query would be to export the "nation" relation, and feed it into a stand-alone augmentation system, such as the *REA* system introduced in the previous chapter. However, part of our solution to these challenges is to introduce the top-k entity augmentation paradigm, in which the system produces several alternative solutions from which the user has to choose from. This process would therefore result in a top-k list of possible augmentations for the exported table. Since a standard DBMS can not process the query based on a multi-valued augmentation, the user would have to choose one of the augmentations while in the independent context of the augmentation system, and then re-import the selected augmentation into the DBMS. Again, iterations of this process may be necessary if the initial result is not satisfactory.

In the next section, we will discuss the challenges that arise when entity augmentation is utilized in analytical scenarios involving traditional database management systems. From the identified challenges, we will derive the need for closer integration of top-k EA systems and RDBMS, and derive requirements for a hybrid system in Sect. 3.2. In Sect. 3.3, we will introduce the system architecture of our DrillBeyond system, and describe its core, the DrillBeyond plan operator. We will also discuss peculiarities of hybrid augmentation/relational query processing, and introduce Drillbeyond's ways of dealing with them. Finally, we will survey related work in Sect. 3.5.

Parts of the material presented in this chapter have already been published in [20, 21].

## *3.1 Motivation and Challenges*

We already introduced how top-k entity augmentation can be applied in analytics scenarios to fulfill ad-hoc information needs. However, we also argued that in complex analytical scenarios, using a standalone entity augmentation system to answer ad-hoc information needs has several deficiencies that should be discussed in the following

*Context-Switching* First, there is a cost associated with context-switching, both with respect to *user effort*, but also with respect to *data locality*. The user would be

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

required to move the data that is to be augmented into the specialized data search and integration system, such as REA presented in Sect. 2, inspect and verify the solution in this context, and then move the data back into the actual analytics system. On the one hand, this introduces a considerable overhead into the analysts workflow, requiring additional effort that may even discourage from performing certain ad-hoc exploratory queries at all. On the other hand, it also introduces physical overhead of moving the data between systems. This overhead may be negligible if a small dimension table, e.g. the "Nation" table in the example, is to be augmented. For larger tables, however, data transfer times can be significant, and further impede an interactive analytics workflow.

*Incompatible Query Model* A second challenge when introducing top-k entity augmentation into a traditional analytics workflow is the mismatch in query models. Augmentation systems such as REA produce top-k results, while other parts of the environment, such as DBMS, work with exact, single results. In this respect, top-k augmentation systems are more similar to information retrieval systems that handle the uncertainty of their results by producing a list of possible results. However, a traditional DBMS is not natively prepared to handle multi-variant data. This gap needs to be bridged in order to enable effective combination of the two system types.

*Loss of Context Information* Third, by using a separate system for augmentation queries, the broader context associated with the analytical task is not taken into account. A generic entity augmentation system uses only the set of entities and the augmentation attribute as input to guide its data search and integration process. However, the query context may contain valuable hints that can improve the augmentation system performance or precision, if the systems were able to exploit them. For example, in an SQL query, other tables that are joined with the augmented table may provide useful context for the data source retrieval and matching process in the augmentation system. Similarly, predicates used in the original query add semantics that can be used to improve the precision of downstream augmentation system. In an optimal combination of DBMS and entity augmentation system, such context information incurring in one system would be utilized in the other.

*Unused Optimization Potential* Finally, by using separate systems, query *optimization potential* is wasted. For example, DBMS use cardinalities and estimated selectivities to choose an optimal join order for a given query. If a manual entity augmentation has to be performed before the query can be executed, then the cardinality of the augmented relation, or the selectivity of predicates on the augmented attribute can not be exploited in this optimization process. Depending on the exact circumstances, it can be beneficial to intermingle the normal DBMS query processing and the augmentation query processing to achieve optimal performance.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

## 3.2 Requirements

From the challenges we identified in Sect. 3.1, it is easily recognizable that a closer integration of the two system types DBMS and EAS (Entity Augmentation System) is necessary. In this section, we will introduce requirements for a hybrid system that is able to close this gap.

*DBMS-integrated Entity Augmentation* In the previous section, we discussed how the lack of integration between DBMS and EAS systems leads to an increased user effort for situational one-of analysis queries. This effort could be reduced if the DBMS would directly support looking up and integrating Web data sources as part of its query processing, and allow the specification of such queries declaratively in SQL. However, there are several differences to bridge. First, the two system types differ in the type of data they manage: DBMS deal with structured and cohesive databases, while EAS deal primarily with large heterogeneous corpora of Web data sources. Further, DBMS work with fully specified queries in a structured language, while EAS, lacking a defined schema, accept keyword queries. We therefore require a design that blurs the line between these classes of systems in all three aspects mentioned above: type of data managed, query language used, and nature of the query result. The resulting hybrid system should be able process mixed SQL/EA queries, which we will call *Open World SQL queries*. In these queries, the user may reference arbitrary additional attributes not defined in the schema. We will use Fig. 8 as our running example for such a query. The system will associate values of these additional attributes to instances at query processing time, avoiding an explicit data retrieval and integration step. This is achieved by executing top-k entity augmentation queries at runtime, which are integrated as a new type of subquery of regular relational queries. Since a top-k augmentation query will return multiple augmentations as described in Sect. 2.3, an Open World SQL query will, instead of returning a definitive query answer, return multiple alternative query results as well. Figure 9 gives an intuitive overview of our goal. It illustrates how an Open World query is processed by integrating entity augmentation into query processing, producing $k$ alternative SQL query results.

In conclusion, the system should produce structured results of exactly the form specified by the user query, just as a regular DBMS would, but also presents several possible versions of the result, similarly to an information retrieval system. However, producing multiple alternative SQL results for a single query has performance implications, which we will discuss next.

*Efficient Multi-result Query Processing* A naïve approach to bridging the different query models of DBMS and top-k EA systems would be to process the EA query, and then process the SQL query multiple times. For example, if $k$ possible augmentations are requested, the runtime of the SQL query is increased by this factor $k$. This is not acceptable, since in many Open World SQL queries, the majority of the processing time will still be spent processing local data, which does not change between runs of the query. For example, consider again the example Open World SQL depicted in Fig. 8. Here, a large part of the work consists of local joins between the relations

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

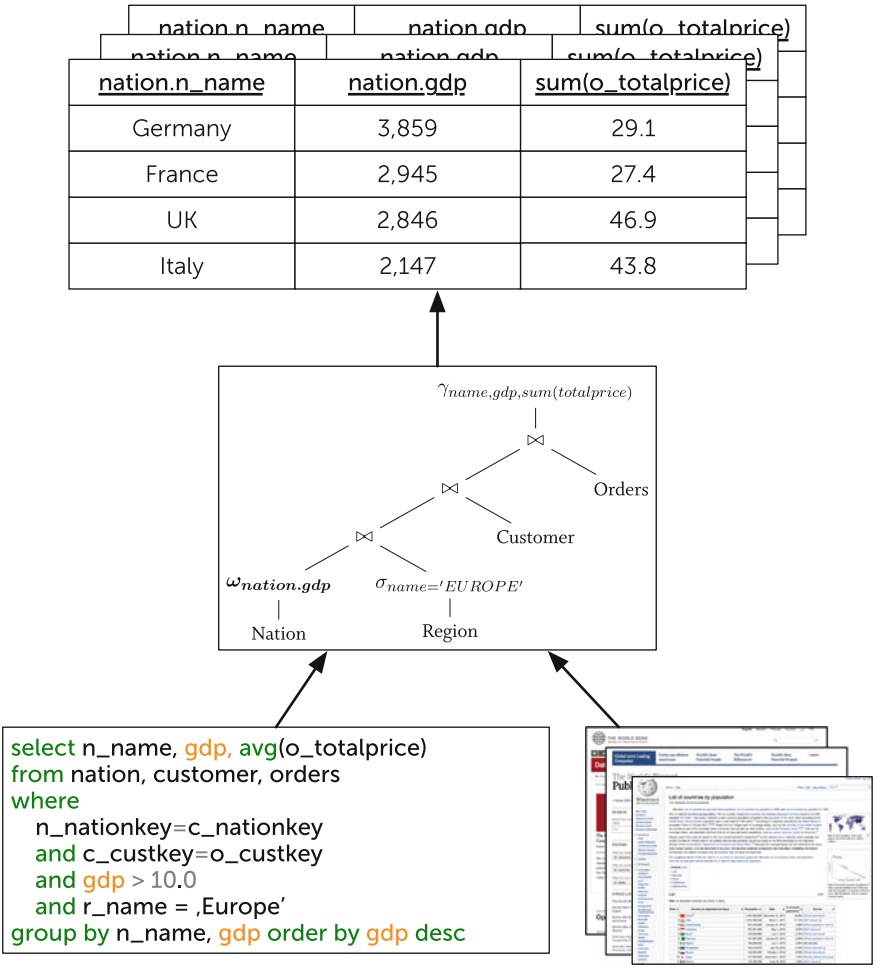



**Fig. 9** RDBMS-integrated top-k augmentations using web tables

`Customer` and `Order`, and aggregation of the local attribute `o_totalprice`. When processing this query multiple times based on different augmentations, only the set of nations that pass the predicate on `gdp`, as well as the order of result tuples would change, but not the aggregates for the individual nations. Consequently, the hybrid system should process Open World SQL queries in a way that minimizes duplicate work between query variant executions.

*Open World Query Planning* The third requirement arises from the fact that properties of data sources used in augmentation are not fully known at plan-time. For instance, estimating the selectivity of a predicate over an augmentation attribute is not easy, as the set of data sources that will be used is not known at planning-time. The same

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

holds for determining the open attributes' metadata, such as the data type, since we do not require the user to specify it in the query. Therefore, the system should be able to plan queries even if some attributes are only fully known at run-time.

With these requirements established, we will introduce the *DrillBeyond* system and its entity augmentation operator in the following section.

## *3.3 The DrillBeyond System*

To solve the challenges identified in Sect. 3.1 and enable entity augmentation queries as part of relational query processing, we designed the *DrillBeyond* system. It is an RDBMS/EAS hybrid, that embeds entity augmentation sub-queries into standard RDBMS query processing. The next sections will detail the required changes to the RDBMS architecture to realize this mixed query processing.

**System Architecture** Processing open world SQL queries requires a top-k entity augmentation system, including a data source management system, as well as modifications to three core RDBMS components: the analyzer, the planner and the executor. Figure 10 gives an overview of the modified and the novel components, and further includes a high level description of the changes in control flow. The core augmentation functionality is introduced through the new *DrillBeyond plan operator*, which will be discussed in detail in Sect. drillbeyond:sec:DrillBeyondspssystem. In the following, we will first give a general overview of all the novel or modified DBMS components in DrillBeyond.

*Data Source Management System* A standard RDBMS is tailored to manage a relatively small set of relations that form a coherent schema. An EAS, on the other hand, manages a large corpus of heterogeneous individual Web data sources. We aim at enabling one system to process both kinds of data. The DrillBeyond system does not make assumptions regarding the nature of the data sources and system that they are managed by. A generic system that exposes an interface for keyword-based dataset search is sufficient. For example, when using our proposed EAS (see Sect. 2), an industry-standard document index server such as Solr[3] or ElasticSearch[4] is sufficient. The necessary source selection, matching and integration operations are performed in the integrated entity augmentation system, which we discuss next.

*Entity Augmentation System* This component implements the actual top-k entity augmentation processing inside the DBMS. It interfaces with the data source management system to retrieve Web data sources, and with the core RDBMS components to provide augmentation services to the executor and the planner. DrillBeyond extends the generic augmentation query definition (see Definition 1) using *query context hints* $H$, which are extracted from the respective outer SQL query. These hints are used to guide the search and the ranking of Web data sources. For example, if the

[3] http://lucene.apache.org/solr/.

[4] https://www.elastic.co/.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

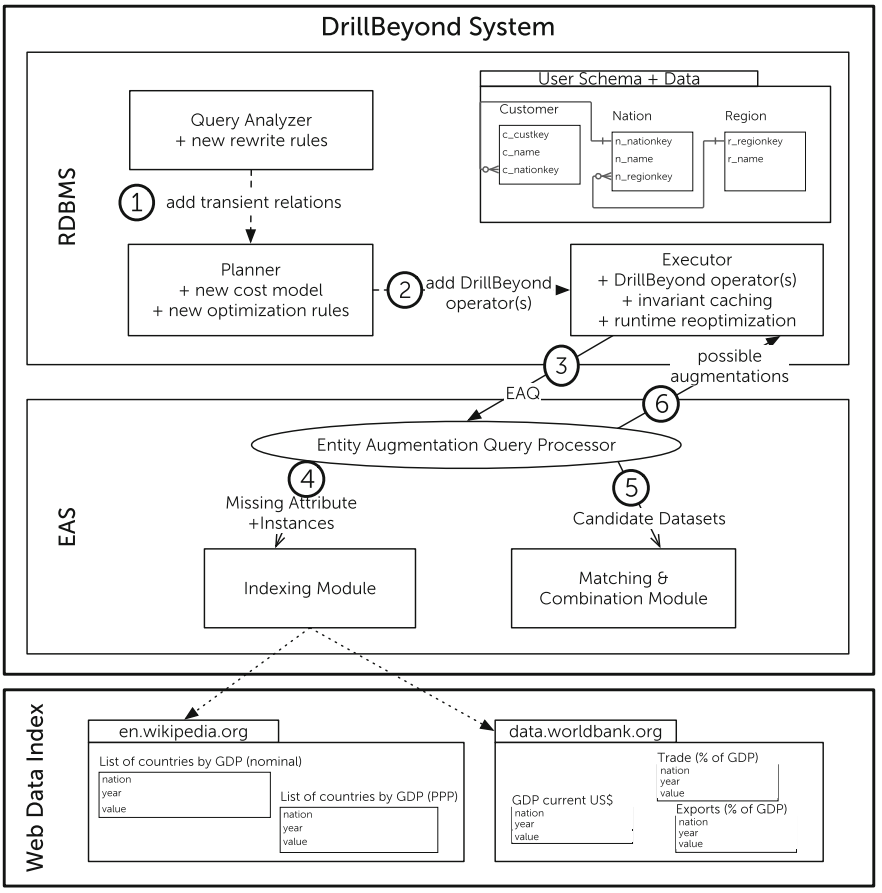



**Fig. 10** System architecture and high level control flow

outer SQL query includes a numeric predicate on the augmentation attribute, this fact can be used as a query hint by instructing the augmentation system to only retrieve data sources that provide numeric values. We discuss the exact nature and usage of these context hints in section "Pushing SQL Query Context". The complete interface used by the executor is therefore $Q_{EA}(a_+, E, k, H)$. Having introduced the novel components necessary for entity augmentation in the DrillBeyond system, we will now give an overview of the modifications to existing RDBMS core components.

*Query Analyzer* The first step in DrillBeyond query processing is triggered by the query analyzer, which maps tokens in the SQL query string to objects in the database's metadata catalog. Unrecognized tokens, such as *gdp* lead to an error in a typical RDBMS. In the DrillBeyond system, we take a minimally invasive approach: we introduce transient metadata for the duration of the query, so that the regular analysis can continue. The query is then rewritten to include an additional join with a transient

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

relation, effectively introducing a source for the missing attribute into the query processing, and also paving the way for the DrillBeyond operator to be placed by the regular join order planning mechanisms of the DBMS. The analyzer is also responsible for determining the type of all expressions in the query. In the case of augmentation attributes, which are not represented with a type in the database catalog, we include a type inference mechanism. It first tries to infer the data type syntactically, by considering filter and join predicates the attribute is used in, and comparing it with the types of regular attributes and constants in these expressions. If syntactic inference is not possible, DrillBeyond infers a type statistically. It uses the augmentation system to determine the most common data type occurring for attributes named $a_+$ in the dataset corpus using a fast probe query. Having created the necessary query metadata, the analyzed statement can be passed on to the modified query planner.

*Query Planner* In DrillBeyond, the query planer has several new tasks to perform compared to a regular RDBMS. First, it needs to place the DrillBeyond operator in the query plan. This placement is crucial to the execution time of the query but also influences the augmentation quality. Furthermore, while regular query plans are created to be optimal for a single execution, multi-solution processing requires plans that minimize the overhead of creating multiple result variants based on top-k augmentations. This is impeded by the lack of plan-time knowledge about the data sources, requiring plan adaption.

*Executor* The executor is modified to repeatedly execute the planned operator trees, creating the top-k query result. In each iteration, it orders the DrillBeyond operators to augment incoming tuples with values from a different augmentation. It further tags the SQL results produced with a distinct augmentation id, by adding a column carrying this id to each finished result. This allows external tools using the top-k SQL result to distinguish between the tuples belonging to alternative results. The majority of new functionality, however, is part of the DrillBeyond operator itself, which we will detail in the next section.

**The DrillBeyond Operator** In its basic form, the DrillBeyond plan operator, denoted $\omega$, is designed to resemble a join operator, which facilitates integration with the existing system architecture. Specifically, it acts like an outer join: it adds new attributes to its input tuples based on join keys, but will not filter original tuples if no partner is found. Instead, it adds null values if the augmentation system can not produce a value. In this way, the part of the query operating on local data can still be processed. However, in contrast to a regular join, only one of the joined tables is known at plan-time, while the other table, as well as the join keys, are decided at query processing time. These run-time decisions are made by the entity augmentation system based on the input tuples of the operator. Specifically, the operator extracts distinct combinations of textual attributes from input tuples, as these are used to functionally determine the values of the augmented attribute.

Algorithm 2 shows the specifics of the state kept in the operator and the implementations of its iterator interface and helper functions. DrillBeyond uses a traditional

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

**Algorithm 2** DrillBeyond operator

```
function INIT
    state ← 'collecting'
    tuplestore ← ∅
    augMap ← HashMap()
    n ← 0                                   ▷ Current Iteration, runs from 0 to k − 1

function NEXT
    if state = 'collecting' then
        COLLECT()
        AUGMENT()
        state ← 'projecting'
    return PROJECT()

function COLLECT
    while true do                           ▷ Retrieve all tuples
        t ← NEXT(childPlan)
        if t = NULL then
            break
        tuplestore ← t
        augKey ← TEXTATTRS(t)
        if augKey ∉ augMap then
            augMap[augKey] ← ∅

function AUGMENT
    augReq ← (∀k ∈ augMap | augMap[k] = ∅)
    for all augKey, [augValues...] ∈ SEND(augReq) do
        augMap[augKey] ← [augValues...]

function PROJECT
    t ← NEXT(tuplestore)
    if t = NULL then return NULL
    augKey ← TEXTATTRS(t)
    t[a+] = augMap[augKey][n]
    return t

function RESCAN
    state ← 'collecting'
    tuplestore ← ∅

function NEXTVARIANT
    RESCAN(tuplestore)
    n ← n + 1
```

row-based iterating executor. The conventional interface functions *Init()*, *Next()* and *ReScan()*, as well as the novel *NextVariant()* function, are called by the DBMS during regular query processing. The other functions shown in Algorithm 2 are used internally by the operator.

The *Init()* function, which is called by the RDBMS executor before processing the query the first time, initializes operator state. This includes a tuple store for material-

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

izing the lower operator's output, a hash table mapping local textual attribute values to augmented values called $augMap$, and two variables $state$ and $n$, determining the behavior of the operator when *Next()* is called.

The *Next()* function is called by the executor and produces augmented tuples. This is done in three phases: *Collect()*, *Augment()* and *Project()*. On the first call to *Next()*, since no augmented values are available, the first two phases are triggered. In the *Collect()* phase, the operator pulls and stores all tuples that the lower plan operator can produce, making DrillBeyond a blocking operator. The reasons for blocking are discussed in section "Augmentation Granularity". In this phase, the operator also stores the textual attributes originating from the augmented relation and its context in a hash table, to obtain all distinct combinations of textual values in the input tuples. In the *Augment()* phase, all entries in the augmentation map that do not yet have values associated with them are passed to augmentation system as one augmentation context. After successfully retrieving values for all collected tuples, the operator is put into the *projecting* state, and produces the first output tuple. Output tuples are produced in the *Project()* function by replaying the stored tuples and filling the augmentation attribute by looking up values in the hash table.

The *ReScan()* function is called by the DBMS executor when subtrees have to be re-executed, e.g., in dependent subqueries or below a nested loop join. Here, the operator empties its tuplestore and changes state to collect new input, but keeps its augmentation hash table, to prevent expensive re-augmentation for values that have already been seen. Finally, *NextVariant()* is an interface extension not seen in typical RDBMS operators, which is necessary for producing the multi-variant query results as discussed in Sect. 3.2. When called, the operator's tuplestore is prepared for another iteration over the stored tuples using *ReScan()*, and the iteration counter $n$ is incremented. This makes sure that in a new execution of the query plan, operators below the DrillBeyond operator are not called again. Instead their materialized output will be replayed, and augmented with the next augmentation variant in the *Project()* function.

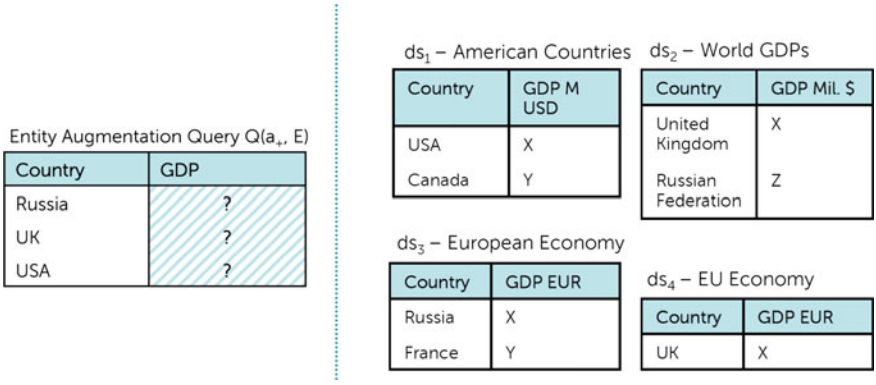



**Fig. 11** Example augmentation problem

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

The functions *Init()* and *Next()* are part of the traditional iterator interface used in RDBMS and are called by the executor in regular query processing. The *NextVariant()* function is different however, and requires novel functionality in the executor. Specifically, when the top operator of the plan returns null, the executor usually assumes that all data has been sent and stops the processing. DrillBeyond however, keeps a global iteration count $n$, running from 1 to $k$, to track the number of produced alternative results. In case the plan has finished, $n$ is increased, *NextVariant()* is called on the DrillBeyond operator, and then the whole plan is restarted. In case there is more than one augmentation attribute, and thus more than one DrillBeyond operator, the system produces a crossproduct of alternative results. This is achieved by systematically calling *NextVariant()* only one operator in each iteration, to iteratively produce all combinations of possible results for all augmented attributes.

Having introduced the basic operator functionality, and the way it is called by the DBMS executor, we will discuss the operator characteristics in the following subsections.

**Augmentation Granularity** A naïve entity augmentation operator working *tuple-at-a-time* would be most compatible to the iterator-based query processing used in most traditional RDBMS operators. However, augmenting each tuple on its own implies looking up and matching Web data sources for each tuple individually. Consider the simplified augmentation example shown in Fig. 11, where the table to be augmented is on top, and the available Web data sources at the bottom. With the *tuple-at-a-time* style, the augmentation system may choose $ds_1$, $ds_2$ and $ds_4$ for the USA, Russia, and UK tuples respectively. These sources match each individual tuple best, and the individual tuples are what the augmentation system can process in this case.

Still, we can see that the augmentation system can not perform optimally with regard to the query as a whole, as it is not provided with the overall query context. While the chosen sources are the best fitting for each individual tuple, they do not form a consistent joint result, as the units of currency do not match. If the augmentation system is instead provided with the set the complete set of tuples as the input for one augmentation query, the more consistent solution comprised of $ds_1$ and $ds_3$ can be constructed, even though the individual entity matches are slightly worse. This is a similar argument to those we mentioned in Sect. 2 about result consistency. Extending single tuples individually would correspond to the *By-Entity Fusion* augmentation strategy as introduced in section "Entity Augmentation Queries" leading to results with similar deficiencies as discussed there. In [22], we showed that our approach to entity augmentation leads to higher quality results, but requires all entities to be queried in a group. Otherwise, the consistency of the source selection can not be ensured. We conclude that for reasons of result quality, the DrillBeyond operator needs to be a blocking operator, i.e. it consumes tuples from underlying operators until they are exhausted, then hands them over to the augmentation system, and produces the first result tuple only when it returns. Referring back to Algorithm 2, this blocking behavior is realized in the state "collecting" in function *Next()*.

**Context-Dependent Results** Having established the DrillBeyond operator as blocking, we will now consider the question of where to place it in a query plan. Let us

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

assume two queries. The first one performs the augmentation directly after the scan of the TPC-H `Nation` table, while in the second query $\omega$ is executed after `Nation` has been joined with table `Region` and was filtered for European countries only. In the first case, the augmentation system will retrieve, match, rank, and combine datasets for all countries in the local database. In the second case, the local join will remove tuples about non-European nations, so the results of the entity augmentation will be more likely to be based on data sources specifically about Europe. Furthermore, completing the join with the `Region` table does not only limit the scope of the query, it also adds context to each tuple by adding `Region`'s attributes. So even if the join would not act as a filter limiting the number of tuples, adding information about the region name will improve the accuracy of the augmentation system. For example, while augmenting a set of *City* tuples may be hard and error-prone because of the ambiguity of common city names such as "Springfield", augmenting after a join with a *State* will be a more realistic task for the augmentation system. We have identified the following property of the $\omega$ operator:

**Definition 6** *(Selection Dependency)* The DrillBeyond operator is not associative with respect to selection in the general case. When augmenting relation $R$ with attribute $a_+$ and selecting with predicate $p$ on $R$, then $\omega_{R,a_+}(\sigma_p(R)) \neq \sigma_p(\omega_{R,a_+}(R))$.

Note that the augmentation system uses only the textual attributes $textAttr(R)$ for matching with Web data sources. Therefore, in the special case that the set of distinct textual attribute values is invariant under predicate $p$, the augmentation results are also invariant under selection with $p$. For example, if a selection $\sigma_p(Nation)$ returns at least one tuple for each `Nation` in its result set, then $\omega_{Nation,a_+}(\sigma_p(Nation)) = \sigma_p(\omega_{Nation,a_+}(Nation))$. This however, is not the general case, and can not be relied upon.

A similar dependency property holds for another form of query context, namely for the set of attributes of the input tuples of an operator $\omega$. As mentioned, the key for searching matching Web datasets for $R$ are its textual attributes $textAttr(R)$. For example, not projecting the attribute `n_name` of the TPC-H `Nation` relation, possibly because it is not part of the desired query result, will make augmentation impossible. No Web data source can be found if the natural key of the relation, the nation's name, is not part of $\omega$'s input.

**Definition 7** *(Projection Dependency)* The DrillBeyond operator is associative with respect to projections only if it includes all textual attributes of $R$. Formally, when augmenting relation $R$ with attribute $a_+$, and projecting to a set of attributes $\mathcal{A}$ with $\mathcal{A} \cap textAttr(R) \neq textAttr(R)$, then $\omega_{R,a_+}(\pi_{\mathcal{A}}(R)) \neq \pi_{\mathcal{A}}(\omega_{R,a_+}(R))$.

Given that the result of $\omega_{R,a_+}$ changes under projection and selection, we define the following placement rule that defines bounds on the placement of the operator.

**Definition 8** *(Placement Bounds)* The DrillBeyond operator augmenting a relation $R$ can only be placed with respect to the following conditions:

1. after all operators filtering $R$ such as joins and selections
2. before any projection removing textual attributes of $R$

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

In other words, the DrillBeyond operator $\omega$ is always applied to the minimum number of distinct combinations of textual column values in the tuples of $R$, as these determine the matching process and its result.

**Pushing SQL Query Context** As mentioned in Sect. 3.1, context from the outer SQL query can be used as filter to improve the accuracy and runtime of the inner entity augmentation query, when compared to isolated augmentation. These filters work implicitly to improve the augmentation quality by narrowing the scope of the augmentation operation, and do not require any changes to the augmentation system or its API. However, we can further improve the augmentation by explicitly pushing additional query knowledge to augmentation system. Specifically, we push two types of information: *type information* and *predicates on augmented attributes*.

*Type Information* Though the user can specify a data type such as *text* or *double* for open attributes using SQL syntax, we do not expect those annotations to be provided. However, in many cases it is possible to infer the type of the open attributes from the surrounding query by applying methods of type inference to SQL. As mentioned in section "System Architecture", we integrated a type inference mechanism that determines an open attribute's data type both from the query expressions it takes part in, and from the Web data corpus used. We can pass type information to the augmentation system, which in turn uses this type information to restrict the set of candidate data sources to those matching the open attribute's type. Consider again the query shown in Fig. 8. From the constraint on the *gdp* attribute it can be inferred that it must be of a numeric type. This allows the augmentation system both to reduce its runtime and increase precision by pruning non-numeric candidate sources.

*Predicates on augmented attributes* In addition to the data type of open attributes, we can also push-down the predicates on open attributes themselves. This allows a similar, but more sophisticated, candidate pruning. We assume that users expect some filtering on the database instance level to happen when specifying a predicate, i.e., we assume that some domain-knowledge is encoded in the predicate. With the query shown in Fig. 8, the user clearly intends to filter low *GDP* countries, and has given a relatively large integer number as the specific condition. Though the user query is given only with the very general keyword *GDP*, by also considering the predicate, the augmentation system can improve its data source ranking. In the example, all candidate datasets that give the GDP as a percentage or rank can be ranked lower, because those datasets will not discriminate the entities with respect to the predicate. In other words, given the predicate above, using a dataset that provides *GDP* rank values will lead to all entities being filtered, which is clearly not the user-intent. To improve the augmentation systems ranking, we can therefore check how well the data source's values fit to the predicate that will be applied, i.e., how well the data source discriminates the entities with respect to the predicate. Additionally, we check whether the predicate value and the average of the data source's values are in the same order or magnitude. Again, the intent is measuring whether the data source is fit to evaluate the given predicate.

The improvements in precision and runtime of the EAS when considering predicate and type information are shown in [21].

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

**Cost Model and Initial Placement Strategy** The observations from the previous three sections might suggest that the DrillBeyond operator should be placed as late as possible in the query plan, to maximize the context knowledge available in the intermediate result. However, in addition to quality considerations, there are also performance considerations to be made. As mentioned, the operator is modeled to resemble a join from the perspective of the DBMS. We can therefore reuse the existing join optimization machinery to place the DrillBeyond operator. This however depends on a model for the operator runtime and the operator's output cardinality.

*Output Cardinality* In the most basic case, the operator produces exactly as many tuples as its input relation, as it just adds a single attribute to each tuple. However, we also must consider selectivity of possible predicates on augmented attributes, and the value of $k$, i.e., the number of alternative augmentations that are to be processed. Since at plan-time we do not assume any knowledge about which Web data sources will be used to augment an attribute, correctly estimating selectivity is almost impossible. A well-known solution to processing queries with unknown selectivities is run-time plan adaptation [33]. We therefore initially use the DBMS' default selectivities for different types of predicates, and employ run-time optimizations to compensate when more information is available. The variable $k$ on the other hand is known at plan-time, and could be part of the cost model. However, with our execution strategy, the operator does not produce $k$ tuples for each input tuple, but just a different one in each of $k$ query executions. We therefore do not consider $k$ at this point, but create optimal plans for single execution, and then later optimize their re-execution.

*Cost Model* The operator's runtime depends on three components: The first part is incoming tuple processing, the second entity augmentation, and the third is projecting tuples with the augmented attribute (see Algorithm 2). Since we designed the operator as blocking, the first part consists of reading all tuples from lower plan nodes, storing them, and computing all distinct combinations of all textual attributes, which are needed by the entity augmentation system for matching. In the second phase, the distinct combinations are then submitted to the augmentation system. The third phase then consists of iterating all stored tuples, and projecting the new attribute based on the combination of textual attributes found in each tuple.

We can estimate the cost of the operator using a similar model as for a hash join, as phases one and three, hashing the child relation, and then probing it against the augmented hash table correspond to the phases of a hash join. Additionally, there is the cost of phase two, the actual augmentation, which depends not on the number of tuples, but on the number of distinct entries in the augmentation hash table. The processing cost per entry depends on the augmentation algorithms, and is therefore not easy to model in the context of a generic DBMS cost model. However, we can assume that the cost per entry is in a different order of magnitude than the per-tuple cost of the relatively primitive database operations such as comparison or hashing. For our cost model we therefore assume an additional large constant factor $\mathcal{C}$ which is learned from previous executions of our augmentation system (see Sect. 2).

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

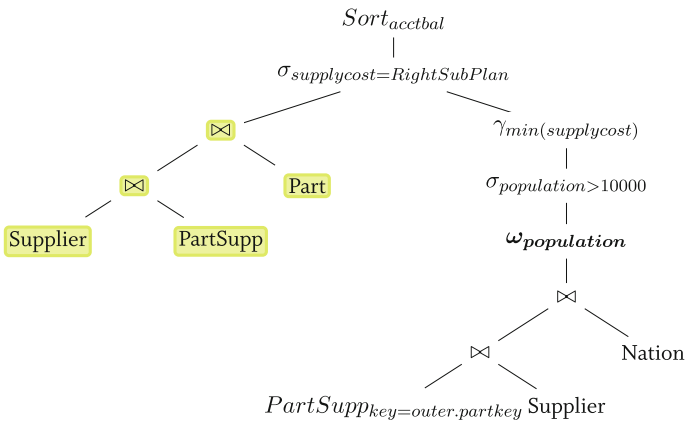



**Fig. 12** Plan invariants

## *3.4 Processing Multi-result Queries*

So far, we have considered the DrillBeyond operator in a single query result setting. However, as discussed in Sect. 3.2, we aim at translating the top-k result returned by the augmentation system into a top-k SQL result. The naïve method, given our operator, is to simply re-execute the query plan $k$ times, and after each execution trigger the projection of a new augmentation result from the operator via the *NextVariant()* API depicted in Algorithm 2. This obviously leads to duplicated work, as only the output of the DrillBeyond operator changes between executions, while the other parts of the query plan operate the same way. Consider the query plan in Fig. 12. Here, only the values of the `gdp` attribute would change between query runs, while the other operations, notably the more expensive joins with the `Customer` and `Orders` relations, would not change. This means that simple re-execution would increase time to compute the query $k$-fold, with most of the effort being inefficient duplicate work. Our first approach to this problem is to identify and then maximize invariant parts of the multiple executions, preventing their re-execution by materializing intermediate results. As a first observation, note that all tuple flows below the DrillBeyond operator can actually never change between executions: in the example shown in Fig. 12, these operators are highlighted. The cost of those operations can be minimized by materializing the input to the DrillBeyond operator. This elementary optimization is already included in the basic operator implementation shown in Algorithm 2, in the form of the tuple store created by the operator. However, the changing augmentation output may influence the result of other operators further up the query plan. In the example, the aggregation of the `gdp` attribute will change its output in each iteration based on the augmentation values provided by the DrillBeyond operator. In addition, even the aggregation of the regular attribute `totalprice` is influenced by the changing `gdp` values: Since the selectivity of the predicate on `gdp` may vary

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

using different augmentations, the set of `Orders`-tuples that has to be aggregated may vary in different executions as well.

In [21] we introduced a series of optimization strategies for these problems including invariant caching, augmentation operator splitting, selection pull-up, projection pull-up partial selection and run-time reoptimization. We also implemented the described system in PostgreSQL and evaluated it on modified TPC-H queries [21]. The evaluation shows the effectiveness of various optimizations in minimizing the runtime overhead of producing multiple SQL query results based on alternative augmentations. Finally, we showed that pushing SQL query context into the EA system can improve quality and performance of the EA processing compared to standalone processing.

The tight integration of augmentation and relational query processing and its various optimizations provided by DrillBeyond, enables the use of ad-hoc data search and integration in new contexts, and greatly increase the practicality of the entity augmentation methods.

## *3.5 Related Work*

To the best of our knowledge, there is no previous work that allows the user to augment a relational database with external data at query-time by simply adding arbitrary new attributes to an SQL query. However, there are considerable amounts of works that can be related to the various individual aspects of DrillBeyond.

*Self-service BI and Mashup Tools* A work closely related in spirit to ours is [2]. In this paper, the authors discuss their vision for self-service BI, in which an existing data cube can be semi-automatically extended with so-called *situational data* in the course of an interactive OLAP session. However, in contrast to the work at hand, it is a vision paper, and does therefore not present a concrete system design. In [40], a vision for ad-hoc BI based on data extracted from the textual Web content is presented. They envision using cloud-computing and parallelized information extraction methods to answer OLAP queries over the general Web, e.g., by enabling ad-hoc extracting of product data, reviews and customer sentiments from the general Web. Further, there is a considerable amount of work aimed at bridging traditional OLAP and semantic Web technologies to enable novel, agile forms of business intelligence. [1] gives a comprehensive survey of these efforts to utilize the semantic Web to acquire and integrate relevant external data with internal warehouse data. For example, [25] propose a RDFS vocabulary for expressing multidimensional data cubes, so-called *Web Cubes* over semantic Web data, and show how traditional OLAP operations can be performed over combined internal and Web cubes.

One of the central properties of our proposed *Open World Queries* is their declarative nature: users specify their information need simply by an attribute name, i.e., a keyword query. While this approach is sufficient in many use cases, more complex Web data integration problems may require more sophisticated, programmatic

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

specifications. One class of approaches to this problem, called *Mashup tools*, aims at allowing users with only basic programming knowledge to easily compose various data sources to higher level services [16]. Many of these tools have been proposed, which differ in their level of abstraction, community features, user interfaces, or support for discovery of data sources and mashups [28]. Similarly to our work, most research on mashups uses public Web data sources because of their general availability. However, mashups have also been recognized as a lightweight form of data integration for the enterprise context, which is exemplified by systems such as IBM's Damia [51]. In [54], a framework for mashup construction with strong focus on data integration is proposed. It adds more complex data transformation operators, such as a *fuse* operator for automatic object matching, and an aggregation operator to reduce multiple matches that result from a fuse operation to a concise representation. Furthermore, it introduced entity search strategies that minimize the number of queries that need to be issued to data sources such as entity search engines or Deep Web databases. These strategies where explored in more detail in [23], where different query generators are combined with query ranking and selection strategies to form an adaptive querying process, which also utilizes initial query results to optimize the choice of further queries. In [24], this feature set is further extended to include efficient, pipelined execution of such query processes, enabling stream-based processing and quick presentation of initial integration results. By combining approaches such as the ones outlined here, mashup tools can become a powerful alternative to traditional data integration processes. While they are more expressive than *Open World Queries*, they are only applicable in situations where the sources to be integrated are known, and a user with programming knowledge is available to define the mashup and necessary source adapters.

*Hybrid DB/IR Systems* A closer integration between the worlds of database and IR research has long been a goal of both communities [4, 11, 55]. There are several classes of DB/IR integration, and multiple system architectures for achieving them. First, there are works that aim at including IR capabilities, such as full text search into database engines, e.g., to support keyword queries on textual attributes of a database. Such systems have already been available in commercial RDBMS for some time [17, 31, 41].

Of course, the other direction of system integration is also possible: extending information retrieval systems with more database-like features. The authors of [49] proposed a method for processing semi-structured keyword queries over large, Web-extracted knowledge bases is presented. One novel aspect is the query language, which only adds a minimal amount of structure compared to pure keyword queries, but still allows users to formulate precise information needs without having to understand the large, heterogeneous schema of the underlying knowledge base. The other aspect is the disambiguation of the keywords in the query with respect to the concepts and relations in the knowledge base. It may be worthwhile to apply their method for graph-based query disambiguation to DrillBeyond.

Similar work has been done in the area of XML databases, where a large class of work is aimed at enabling ranked retrieval over collections of XML documents. A

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

recent overview over this extensive field is given in [53]. The problems this research faces are related to those faced by DrillBeyond: underspecified, but structured queries have to be mapped to a large heterogeneous collection of datasets.

In [6], a hybrid DB/IR query type called context sensitive prefix search is introduced. Given a set of documents forming a *context*, and a word prefix, this query type aims at retrieving documents containing words with the given prefix, conditioned on the words also occurring in the context document set. The authors show how, by introducing further structure through special markup keywords, this abstract operator can be applied to implement many database and expressive information retrieval operations, including joins and aggregation.

Other works aim at processing structured queries based on Web information extraction, e.g., [9], or even at decomposing structured queries to keyword queries that can be posed to an information retrieval system [39]. The idea in this case is to enrich the structured query result with relevant documents.

In general, the large amount of related work in DB/IR hybrid technologies shows the necessity of fusing the two paradigms for many applications. By integrating entity augmentation directly into relational query processing, DrillBeyond also aims at supporting scenarios that require a combination of structured and semi-structured data.

## 4 Summary and Future Work

In the era of Big Data, the number and variety of data sources is increasing every day. However, not all of this new data is available in well-structured databases or warehouses. Rather, data is collected at a rate that often precludes traditional integration with ETL processes and global schemata. Instead, heterogeneous collections of individual datasets are becoming more prevalent, both inside enterprises in the form of *data lakes*, and in public spaces such as Web data sources. This new wealth of data, though not integrated, has enormous potential for generating value in exploratory or ad-hoc analysis processes, which are becoming more common with increasingly agile data management practices. However, in today's database management systems there is a lack of support for ad-hoc data integration of such heterogeneous data sources. Instead, integration of new sources into existing data management landscapes is a laborious process that has to be performed ahead-of-time, i.e., before queries on the combined data can be issued.

In this chapter, we introduced a combined database and information retrieval system that enables users to query a database as well as a heterogeneous data repository in a seamless and integrated way with standard SQL. Relevant sources are automatically retrieved and integrated at query processing-time, without further input from the user. The ambiguity resulting from the coarse query specification, as well as the uncertainty introduced by relying on automatically integrated data is compensated by returning a ranked list of possible results, instead of a single deterministic result as in a regular SQL query. This allows the user to choose the best alternative for the problem at hand.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

To achieve that, we introduced a novel method for *Top-k Entity Augmentation* (Sect. 2) which is able to construct a top-k list of consistent integration results from a large corpus of heterogeneous data sources. This technique forms the basis for our *DrillBeyond* system (Sect. 3), which provides hybrid augmentation/relational query processing capabilities. This enables the use of ad-hoc data integration for exploratory data analysis queries, and improves both performance and quality when compared to using separate systems for the two tasks.

In conclusion, we introduced novel, automatic data augmentation methods that harness the large variety of data sources, while requiring minimal user effort and incorporated those methods into traditional relational DBMS, to enable their efficient use in analytical query contexts. To conclude this chapter, we will finally sketch opportunities to extend and build on our contributions in future work.

## *4.1 Future Work*

In the current work, we applied the *Top-k Entity Augmentation* to Web tables only. However, we argue that underlying top-k consistent set covering is a general technique that can be applied to many different forms of data sources. For example, there we discussed related work on augmentation based on information extraction from general Web page text. Applying our approach to generate minimal but diverse covers based on Web pages instead of Web tables would be a promising approach to increase coverage. Furthermore, our approach does not yet consider correlations between sources as a factor of trust. Approaches from data fusion literature that detect and utilize such source correlations could be combined with our set covering approach to increase the precision of the generated covers.

With respect to the *DrillBeyond* system, a possible avenue for future work would be to investigate which parts of the concept could be adapted to modern analytical RDBMS architectures to increase efficiency. In our current work, we integrate entity augmentation with a classical, single-node row store DBMS. However, in many contemporary scenarios, analytical queries are executed on highly parallel, distributed column stores. Investigating how our proposed architecture and optimizations apply to these systems would increase our method's practical applicability. Furthermore, DrillBeyond so far only allows the usage of additional attributes, i.e., it allows only horizontal table augmentation. However, methods for vertical augmentation, or in other words, the ad-hoc integration of further tuples of an existing relation, have also been discussed in related work [29, 48]. These approaches could be integrated with relational query processing as well. Similarly, the materialization of completely new relations from Web Data using just a schema description has also been studied in isolation, but could be integrated with general query processing as well. Furthermore, although DrillBeyond does support joins over open attributes, we did not study the optimization of such joins. In summary, in future work the idea of Open World SQL queries could be generalized from additional attributes to all aspects of SQL and relation query processing.

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies".
Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

## References

1. A. Abello, O. Romero, T. Bach Pedersen, R. Berlanga, V. Nebot, M. Aramburu, A. Simitsis, Using semantic web technologies for exploratory olap: a survey. IEEE Trans. Knowl. Data Eng. **27**(2), 571–588 (2015)
2. A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J.N. Mazón, F. Naumann, T.B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis, G. Vossen, Fusion cubes: towards self-service business intelligence. Int. J. Data Wareh. Mining (IJDWM) (2012). (accepted)
3. R. Agrawal, S. Gollapudi, A. Halverson, S. Ieong, Diversifying search results. In: Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM '09 (ACM, New York, 2009), pp. 5–14
4. S. Amer-Yahia, P. Case, T. Rölleke, J. Shanmugasundaram, G. Weikum, Report on the db/ir panel at sigmod 2005. ACM SIGMOD Rec. **34**(4), 71–74 (2005)
5. P. André, J. Teevan, S.T. Dumais, From x-rays to silly putty via Uranus: serendipity and its role in web search. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (ACM, New York, 2009), pp. 2033–2036
6. H. Bast, I. Weber, The complete search engine: Interactive, efficient, and towards IR and db integration, in *CIDR 2007: 3rd Biennial Conference on Innovative Data Systems Research*, ed. by G. Weikum (VLDB Endowment, Asilomar, CA, USA, 2007), pp. 88–95
7. J. Bautista, J. Pereira, A grasp algorithm to solve the unicost set covering problem. Comput. Oper. Res. **34**(10), 3162–3173 (2007)
8. M.J. Cafarella, J. Madhavan, A. Halevy, Web-scale extraction of structured data. SIGMOD Rec. **37**(4), 55–61 (2009)
9. M.J. Cafarella, C. Re, D. Suciu, O. Etzioni, M. Banko, Structured querying of web text, in *3rd Biennial Conference on Innovative Data Systems Research (CIDR)* (Asilomar, California, USA, 2007)
10. J. Carbonell, J. Goldstein, The use of MMR, diversity-based reranking for reordering documents and producing summaries, in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98* (ACM, New York, NY, USA, 1998), pp. 335–336
11. S. Chaudhuri, R. Ramakrishnan, G. Weikum, Integrating DB and IR technologies: what is the sound of one hand clapping, in *CIDR* (2005), pp. 1–12
12. J. Cohen, B. Dolan, M. Dunlap, J.M. Hellerstein, C. Welton, Mad skills: new analysis practices for big data. Proc. VLDB Endow. **2**, 1481–1492 (2009)
13. N. Dalvi, A. Machanavajjhala, B. Pang, An analysis of structured data on the web. Proc. VLDB Endow. **5**(7), 680–691 (2012)
14. E. Demidova, P. Fankhauser, X. Zhou, W. Nejdl, Divq: Diversification for keyword search over structured databases, in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10* (ACM, New York, NY, USA, 2010), pp. 331–338
15. G.W. DePuy, R.J. Moraga, G.E. Whitehouse, Meta-raps: a simple and effective approach for solving the traveling salesman problem. Transp. Res. Part E Logist. Transp. Rev. **41**(2), 115–130 (2005)
16. G. Di Lorenzo, H. Hacid, Hy Paik, B. Benatallah, Data integration in mashups. SIGMOD Rec. **38**(1), 59–66 (2009)
17. P. Dixon, Basics of oracle text retrieval. IEEE Data Eng. Bull. **24**(4), 11–14 (2001)
18. X.L. Dong, B. Saha, D. Srivastava, Less is more: selecting sources wisely for integration, in *Proceedings of the 39th international conference on Very Large Data Bases, PVLDB'13, VLDB Endowment* (2013), pp. 37–48
19. J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, W. Lehner, Building the dresden web table corpus: a classification approach, in *2nd IEEE/ACM International Symposium on Big Data Computing, BDC* (2015)
20. J. Eberius, M. Thiele, K. Braunschweig, W. Lehner, DrillBeyond: enabling business analysts to explore the web of open data, in *PVLDB* (2012)

Final edited form was published in A bert Y. Zomaya, Sherif Sakr, Hgg., 2017. "Handbook of Big Data Technologies". Cham: Springer, S. 365–407. ISBN 978-3-319-49339-8.
https://doi.org/10.1145/2236584.2236587

21. J. Eberius, M. Thiele, K. Braunschweig, W. Lehner, Drillbeyond: processing multi-result open world SQL queries, in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management, SSDBM '15* (ACM, New York, NY, USA, 2015), pp. 16:1–16:12
22. J. Eberius, M. Thiele, K. Braunschweig, W. Lehner, Top-k entity augmentation using consistent set covering, in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management, SSDBM '15* (ACM, New York, NY, USA, 2015), pp. 8:1–8:12
23. S. Endrullis, A. Thor, E. Rahm, Entity search strategies for mashup applications, in *2012 IEEE 28th International Conference on Data Engineering (ICDE)* (IEEE, New Jersey, 2012), pp. 66–77
24. S. Endrullis, A. Thor, E. Rahm, Wetsuit: an efficient mashup tool for searching and fusing web entities. Proceedings of the VLDB Endowment **5**(12), 1970–1973 (2012)
25. L. Etcheverry, A. Vaisman, Enhancing olap analysis with web cubes, in *The Semantic Web: Research and Applications*, vol. 7295, Lecture Notes in Computer Science, ed. by E. Simperl, P. Cimiano, A. Polleres, O. Corcho, V. Presutti (Springer, Berlin Heidelberg, 2012), pp. 469–483
26. T.A. Feo, M.G. Resende, Greedy randomized adaptive search procedures. J. Glob. Optim. **6**(2), 109–133 (1995)
27. F. Glover, Tabu search-part i. ORSA J. Comput. **1**(3), 190–206 (1989)
28. L. Grammel, M.A. Storey, A survey of mashup development environments, in *The Smart Internet, Lecture Notes in Computer Science* vol. 6400 (2010), pp. 137–151
29. R. Gupta, S. Sarawagi, Answering table augmentation queries from unstructured lists on the web. Proc. VLDB Endow. **2**(1), 289–300 (2009)
30. A. Halevy, A. Rajaraman, J. Ordille, Data integration: the teenage years, in *Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06, VLDB Endowment* (2006), pp. 9–16
31. J.R. Hamilton, T.K. Nayak, Microsoft SQL server full-text search. IEEE Data Eng. Bull. **24**(4), 7–10 (2001)
32. M. Hasan, A. Mueen, V. Tsotras, E. Keogh, Diversifying query results on semi-structured data, in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12* (ACM, New York, NY, USA, 2012), pp. 2099–2103
33. Z.G. Ives, D. Florescu, M. Friedman, A. Levy, D.S. Weld, An adaptive query execution system for data integration, in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, SIGMOD '99* (ACM, New York, NY, USA, 1999), pp. 299–310
34. R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations* (1972)
35. G. Lan, G.W. DePuy, G.E. Whitehouse, An effective and simple heuristic for the set covering problem. Eur. J. Oper. Res. **176**(3), 1387–1403 (2007)
36. D. Laney, 3d data management: controlling data volume, velocity and variety. META Group Res. Note **6**, 70 (2001)
37. O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, C. Bizer, The mannheim search join engine, in *Web Semantics: Science, Services and Agents on the World Wide Web* (2015)
38. X. Li, X.L. Dong, K. Lyons, W. Meng, D. Srivastava, Truth finding on the deep web: is the problem solved? in *Proceedings of the 39th International Conference on Very Large Data Bases, PVLDB'13, VLDB Endowment* (2013), pp. 97–108
39. J. Liu, X. Dong, A.Y. Halevy, Answering structured queries on unstructured data, in *WebDB*, vol. 6 (Citeseer, 2006), pp. 25–30
40. A. Löser, F. Hueske, V. Markl, Situational business intelligence, in *Business Intelligence for the Real-Time Enterprise*, vol. 27, Lecture Notes in Business Information Processing, ed. by M. Castellanos, U. Dayal, T. Sellis (Springer, Berlin, 2009), pp. 1–11
41. A. Maier, D.E. Simmen, DB2 optimization in support of full text search. IEEE Data Eng. Bull. **24**(4), 3–6 (2001)
42. G. Marchionini, Exploratory search: From finding to understanding. Commun. ACM **49**(4), 41–46 (2006)

43. R. Martí, M.G. Resende, C.C. Ribeiro, Multi-start methods for combinatorial optimization. Eur. J. Oper. Res. **226**(1), 1–8 (2013)
44. H. Mohanty, P. Bhuyan, D. Chenthati, Big Data: A Primer (Springer, India, 2015)
45. J. Morcos, Z. Abedjan, I.F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, Dataxformer: an interactive data transformation tool, in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (ACM, New Jersey, 2015), pp. 883–888
46. T.T. Nguyen, Q.V.H. Nguyen, M. Weidlich, K. Aberer, Result selection and summarization for web table search, in *2015 IEEE 31st International Conference on Data Engineering (ICDE)* (2015), pp. 231–242
47. D.E. O'Leary, Embedding ai and crowdsourcing in the big data lake. IEEE Intell. Syst. **29**(5), 70–73 (2014)
48. R. Pimplikar, S. Sarawagi, Answering table queries on the web using column keywords, in *Proceedings of the 36th Int'l Conference on Very Large Databases (VLDB)* (2012)
49. J. Pound, I.F. Ilyas, G. Weddell, Expressive and flexible access to web-extracted data: a keyword-based structured query language, in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD '10* (ACM, New York, NY, USA, 2010), pp. 423–434
50. S. Sarawagi, S. Chakrabarti, Open-domain quantity queries on web tables: Annotation, response, and consensus models, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14* (ACM, New York, NY, USA, 2014), pp. 711–720
51. D.E. Simmen, M. Altinel, V. Markl, S. Padmanabhan, A. Singh, Damia: data mashups for intranet applications, in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08* (ACM, New York, NY, USA, 2008)
52. F.M. Suchanek, G. Kasneci, G. Weikum, Yago: a core of semantic knowledge, in *Proceedings of the 16th International Conference on World Wide Web, WWW '07* (ACM, New York, NY, USA, 2007), pp. 697–706
53. M.A. Tahraoui, K. Pinel-Sauvagnat, C. Laitang, M. Boughanem, H. Kheddouci, L. Ning, A survey on tree matching and XML retrieval. Comput. Sci. Rev. **8**, 1–23 (2013)
54. A. Thor, D. Aumueller, E. Rahm, Data integration support for mashups, in *Workshops at the Twenty-Second AAAI Conference on Artificial Intelligence* (2007)
55. G. Weikum, DB and IR: both sides now, in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data* (ACM, New Jersey, 2007), pp. 25–30
56. M. Yakout, K. Ganjam, K. Chakrabarti, S. Chaudhuri, Infogather: entity augmentation and attribute discovery by holistic matching with web tables, in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, SIGMOD '12* (ACM, New York, NY, USA, 2012), pp. 97–108
57. M. Zhang, K. Chakrabarti, Infogather+: semantic matching and annotation of numeric and time-varying attributes in web tables, in *Proceedings of the 2013 International Conference on Management of data, SIGMOD '13* (ACM, New York, NY, USA, 2013), pp. 145–156
58. C.N. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in *Proceedings of the 14th International Conference on World Wide Web, WWW '05* (ACM, New York, NY, USA, 2005), pp. 22–32