The VLDB Journal manuscript No.
(will be inserted by the editor)

Approximate Query Mapping: Accounting for Translation Closeness

Kevin Chen-Chuan Chang[1], **Héctor García-Molina**[2] ⋆

[1] Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA; e-mail: kcchang@cs.uiuc.edu
[2] Computer Science Department, Stanford University, Stanford, CA 94305, USA; e-mail: hector@db.stanford.edu

Received: date / Revised version: date

Abstract In this paper we present a mechanism for approximately translating Boolean query constraints across heterogeneous information sources. Achieving the best translation is challenging because sources support different constraints for formulating queries, and often these constraints cannot be precisely translated. For instance, a query [score > 8] might be "perfectly" translated as [rating > 0.8] at some site, but can only be approximated as [grade = A] at another. Unlike other work, our general framework adopts a customizable "closeness" metric for the translation that combines both precision and recall. Our results show that for query translation we need to handle interdependencies among both query conjuncts as well as disjuncts. As the basis, we identify the essential requirements of a rule system for users to encode the mappings for atomic semantic units. Our algorithm then translates complex queries by rewriting them in terms of the semantic units. We show that, under practical assumptions, our algorithm generates the best approximate translations with respect to the closeness metric of choice. We also present a case study to show how our technique may be applied in practice.

1 Introduction

To enable interoperability, mediator systems [1, 2] must integrate heterogeneous information sources with different data representations and search capabilities. A mediator presents a unified context for uniform information access, and consequently must translate *original* user queries from the unified context to a *target* source for native execution. This translation problem has become more critical now that the wide range of disparate sources are just "one click away" across the Internet. Achieving the best translation is challenging because sources may use different constraints for formulating queries, and often these constraints cannot be precisely translated. This paper presents a framework that finds perfect mappings if possible, or in general the "closest" approximations, taking into account differences in attribute names, operators, and data formats.

Example 1 Consider a mediator that integrates online shopping sites for books, audio, and videos. Specifically, the mediator presents `Media`(name, format, $\cdots$) as a unified view for user querying (*i.e.*, it supports a query interface with the above constraining attributes). Suppose a user wants to find the "VHS" items by some actor "Harrison." Let us consider translating the corresponding constraints $v =$ [format = `vhs`] and $h =$ [name contains `Harrison`].

The mediator will find perfect mappings whenever possible (*e.g.*, it will translate h to [au contains `Harrison`] for source `fatbrain.com`, and it will leave v as is for `amazon.com`). However, in many cases such perfect mappings simply do not exist, because of the the limited query "capabilities" of target sources. For instance, for source *EB* at `www.evenbetter.com`, neither v nor h can be translated precisely. In particular, *EB* does not support v (*i.e.*, selecting only VHS but not DVD) although it can differentiate movies from other media types (such as books).

When perfect mappings do not exist, some schemes focus on finding "minimal-superset" mappings [3], which will return all the potential answers but with as few unwanted answers as possible. In particular, the mediator will map v to [type = `movies`] (*i.e.*, searching the "movies" category) for *EB*, returning VHS as well as DVD items. Unfortunately, for h the only superset mapping at *EB* is $True$ (*i.e.*, returning the entire source database), which is often unacceptable.

However, in many cases, good approximations do exist, and they may be more favorable. For instance, *EB* can approximate h as [star = `"Harrison"`] to match `Harrison` as a last name. (Note that *EB* requires at least a last name for star.) It will miss those answers with `Harrison` as the first name, *e.g.*, `Ford, Harrison`. However, since most users will actually mean last names (e.g., `Harrison, George`) in such a name query, this mapping may be better than $True$.

In fact, even v may need a different approximation, say if [type = `movies`] returns a huge number of DVDs and very few VHS items. Alternatively, mapping [desc contains `vhs`]

⋆ *Present address:* Insert the address here if needed

simply looks for vhs in the textual descriptions. This mapping may return a lot less data than [type = movies], but may perhaps miss a few VHS items (that do not have the term vhs in their description). If the "false negatives" are acceptable, the alternative mapping may be more attractive. □

We can view a query as a Boolean expression of constraints of the *selection* form [attr1 op value] or the *join* form [attr1 op attr2]. (While not discussed here, we stress that our approach can generally handle join constraints as well; see [4].) These constraints constitute the query "vocabulary," and must be transformed to "native" constraints understood by the target source. For example, constraint [score $>$ 8] may have to be mapped to [grade = A]. In this process, attributes have to be mapped (*e.g.*, score to grade), values have to be converted (*e.g.*, score 8 to grade A), and operators have to be transformed (*e.g.*, ">" to "="). Reference [3] provides more details on how we generally model this constraint-mapping problem in the common mediation architecture [1, 2].

After we first studied query translation [3] in our earlier work, and implemented that machinery, we soon realized that *approximate translation* is critical for "real-world" applications. Our earlier work focused on minimal-superset mappings as the "correct" translations, because the *exact* results can be recovered by post-processing their supersets. As just illustrated, in many cases only approximations exist, and they might be more practical than the strictly correct ones. (Analogously, a concurrent system with strict serializability may result in undesirable low concurrency.) In fact, in our case study of a "real-world" scenario (as Section 7 will discuss), we informally estimated that 70% of the translations must rely on approximation.

Furthermore, different mediation applications need different "correctness" or *closeness* criteria for mappings. It is thus essential for a translation system to flexibly support a wide range of closeness metrics. This paper presents such a framework, where the best approximate translations can be found under any reasonable metric that is "monotonic" (as Section 4 will explain). In particular, the framework supports minimal-superset, maximal-subset (when extra-answers must not be returned), and other "hybrid" criteria in between. Our customizable criteria allow one to quantify "false positives" and "false negatives" that are expected to occur in a translation, in an analogous fashion to how the conventional IR parameters of precision and recall quantify "errors" that occur in executing a single query.

Our results show that, under such flexible metrics, one must cope with *interdependencies* among both query conjuncts and disjuncts (Section 5). It is thus critical to note that query mapping is not simply a matter of translating each constraint separately. Some interrelated constraints can form a "semantic unit" that must be handled together. This discovery is surprising since our previous study [3] showed that query disjunctions can be translated separately, significantly simplifying the translation process. Now, in an approximate translation scenario, interrelation depends on the particular closeness metric, as we next illustrate.

Example 2 Let us continue our movie search example. Suppose that the user is looking for both VHS and DVD formats with the query $Q = v$:[format = vhs] $\vee$ d:[format = dvd]. Let us denote the closest mapping (for some closeness metric) of query Q as $\mathcal{S}(Q)$.

First, suppose the mediator adopts the minimal-superset metric, under which it will generate $\mathcal{S}(v)$ and $\mathcal{S}(d)$ both as [type = movies] (Example 1). In this case, to translate Q, the mediator can separately map the disjuncts, *i.e.*, $\mathcal{S}(Q) = \mathcal{S}(v) \vee \mathcal{S}(d) =$ [type = movies], which indeed precisely translates Q, *i.e.*, $Q \equiv \mathcal{S}(Q)$.

To contrast, assume next that the mediator is concerned about large result sizes, so as illustrated earlier, uses the mappings $\mathcal{S}(v)$ = [desc contains vhs] and $\mathcal{S}(d)$ = [desc contains dvd]. (That is, given the mediator's closeness metric, these are the best approximate translations.) Now $\mathcal{S}(v) \vee \mathcal{S}(d)$ = [desc contains vhs] $\vee$ [desc contains dvd]. This mapping is not as good as [type = movies], which in our example exactly gets all VHS and DVD titles. Thus, for the closeness metric in use, translating v and d separately leads to a suboptimal mapping, and hence disjunction Q is not "separable." □

Query translation must rely on human expertise to define what constraints may be interrelated, and how to translate basic semantic units. For instance, in Example 2 we need a rule for translating the single-constraint pattern [format = F] such as v and d. But do we need a rule for composite queries, *e.g.*, $(v \vee d)$? What kind of queries must constitute such "semantic units"? In this paper we will answer these questions, identifying the essential requirements for a translation rule system.

Based on rules, our challenge is then to translate arbitrary queries as Boolean expressions of constraints (we currently do not handle negation). Our approach is to *divide-and-conquer*. We present Algorithm *NFB* to "decompose" an original query into its semantic units, which can then be translated by the given rules. Note that there are many decompositions, but not all of them will lead to the closest mapping. In our running example, suppose that we are given translation rules for the semantic units (h), (v), (d), and $(v \vee d)$, and we wish to translate query $hv \vee hd$. (Note that we omit the $\wedge$ operator for notational simplicity.) We can decompose the query as $(h)(v) \vee (h)(d)$, or with some rewriting, as $(h)(v \vee d)$. On which expression should we apply the rules to obtain the best mapping? Is the best solution unique? How is the optimality of translation guaranteed? Again, we will answer these questions in this paper.

In summary, we make the following main contributions for approximate query translation:

- We propose a general *framework*, and we define the notion of translation closeness. Our framework can adopt different closeness metrics for different applications.
- We present an *algorithm* for systematically finding the best translation with respect to a given closeness metric. Algorithm *NFB* will find a *unique* best-mapping in the practical cases when a safe decomposition exists.

– We develop fundamental theorems on the separability of query components and safeness of decompositions. These results are critical for the development of any algorithm that attempts approximate query translation.
– We study how to estimate the precision and recall parameters of a translation, and we show that reasonable formulas do exist for such estimation.

We briefly discuss related work in Section 2, and then start in Section 4 by defining closeness criteria that combine precision and recall. Section 5 studies a basic assumption on compositional monotonicity and our results on compositional separability. In Section 6 we present our framework and Algorithm *NFB*. Section 7 discusses a case study to show how our approach may be applied in practice. Section 8 concludes with simple formulas for estimating precision and recall of translated query compositions. Interested readers may also refer to Appendix for the safeness formalism and proof of our results.

2 Related Work

Information integration has been an active research area [1, 2, 5, 6]; however, we believe that our focus on the *constraint mapping* problem is unique. Many integration systems have dealt with source (syntactic) capabilities, *e.g.*, Information Manifold [7, 8], TSIMMIS [9, 10], Infomaster [11, 12], Garlic [13, 14], DISCO [15], and others [16–18]. These efforts have mainly focused on generating query plans that observe the native *grammar* restrictions (*e.g.*, allowing conjunctions of two constraints, or disallowing disjunctions).

Our work complements these efforts by addressing the semantic mapping of constraints, or analogously the translation of *vocabulary* (of native constraints). In particular, the output of our semantic mapping (which uses only the constraint vocabulary understood by the target source) can be the input to the capability mapping that others have analyzed. Section 3 discusses our particular focus and how our approach can be generally applied in a common mediation architecture.

There has also been much work on data translation and schema integration. The main focus of these related efforts (such as [19–24]) is to unify data representations across mismatched domains by transforming data to a unified context, where queries can be performed. In contrast, our complementary goal is to map queries to the native domain where data reside. We believe our approach is especially well suited for autonomous sources containing large volumes of data, such as found on the Web (where it is not economical or feasible to transform all data). In addition, note that in our constraint mapping problem we must consider both data conversion and query capability mapping (as Section 1 discussed). Furthermore, we consider translation errors and closeness, which as far as we know are not considered in traditional schema and data translation work.

Surprisingly, although approximation is critical for query mapping (Section 1), we have seen virtually no translation efforts that stress this notion. However, approximation has been studied for query processing: First, some work aims to reduce processing cost through approximation. For instance, references [25, 26] study the approximate fixpoints of Datalog predicates, and [27] uses approximate predicates as filters for expensive ones. Second, several researchers have explored accelerated but approximated query answering [28–31] to reduce response time. Third, reference [32] develops a framework for representing approximate complex-objects and supporting queries over them. Finally, CoBase [33] explored query relaxation for approximate answering.

We define our translation metrics based on the precision and recall parameters. Both classic notions have been commonly used for quantifying respectively false-positives and false-negatives, most notably for information retrieval [34, 35]. In addition, single-valued measures for IR effectiveness have also been proposed, such as the well-known E-measure [34] (see Section 4).

Finally, the approximate translation discussed in this paper was motivated by our previous work [3]. As Section 1 mentioned, our earlier model of "exact" mappings significantly simplifies the translation process, but unfortunately it cannot accommodate general closeness metrics. In contrast, this paper specifically explores the notion of *approximation*, and deals with mappings under any reasonable closeness metrics (that are monotonic).

3 The Constraint Mapping Problem

We describe the *constraint mapping* problem in a common mediation architecture [1, 2] for integrating autonomous and heterogeneous *sources*. In such systems, *wrappers* unify the source data models, and *mediators* interact with the wrappers to process queries transparently. Our discussion assumes a simple relational view of data. Specifically, wrappers present each source as a set of *source relations*. We believe our framework is not sensitive to the data model; *e.g.*, in reference [24] we discuss the translation of hierarchical data.

A mediator exports integrated *mediator views* (as query interface) for users to formulate queries. Thus, a *user query* $\mathcal{U}$ over some views $\mathtt{V_i}$ has the form (in an SQL-like expression) **select** ... **from** $\mathtt{V_1}, \ldots, \mathtt{V_h}$ **where** C, or algebraically $\mathcal{U} = \sigma_C(\mathtt{V_1} \times \cdots \times \mathtt{V_h})$, where C is a Boolean expression of *constraints*. (The projection operation is omitted as it is irrelevant to our discussion.) Note again that we do not consider negation in this paper. A constraint is either a *selection* condition [$\mathtt{V_i}$.attr1 op value] or a *join* condition [$\mathtt{V_i}$.attr1 op $\mathtt{V_j}$.attr2], where attr1 and attr2 are attributes of view $\mathtt{V_i}$ and $\mathtt{V_j}$ respectively. For simplicity, we may write a selection constraint as [attr1 op value] when the containing view of attr1 is clear from the context (such as in Example 1 where we considered only one integrated view).

A mediator view is typically an SPJ query over some source relations plus possibly some *data conversion* functions. For instance, view (title, ln, fn, review) might be a join of relation (title, review) from source T_1, (title, author) from T_2, and a function NameLnFn(author, ln, fn) for converting

author to last and first names. We can model such a function as a *conceptual relation* with the tuples [author, ln, fn] that "satisfy" the function. Note that in general a view can be a union of SPJ components; *e.g.*, a `book` view can be a union of two relations from two bookstore sources. In this case, we can process each component separately and union the results as is typically done.

For source execution, the mediator must rewrite a user query in terms of the source relations. Thus, with view expansion, $\mathcal{U}$ will be rewritten to the following form in Eq. 1, where $\mathbf{R}_i$ is the cross-product of all the source relation instances that a particular source T_i contributes to any queried views, and $\mathbf{X}$ is the cross product of the relevant conceptual relations. We specifically refer to the selection condition Q as a *constraint query*: In most cases Q is simply the user-query condition C, but in addition Q can also include the constraints used in the view definitions.

$$\mathcal{U} = \sigma_Q(\mathbf{R}_1 \times \cdots \times \mathbf{R}_n \times \mathbf{X}) \quad (1)$$

Intuitively, the *constraint mapping* problem is to *push* as much as possible the constraint query to the sources. That is, the mapping translates Q from the mediator's *original context* to the *target context* at each source. Note that the constraints in Q are generally not readily executable across different contexts. First, there exists *schema* difference between the views and the sources: The conversion functions in $\mathbf{X}$ can present new attributes (*e.g.*, ln and fn that replace author) or change data representations. Second, there exists *capability* differences: Unless the mediator only allows the least common denominator of what the sources support, the constraints can be beyond the capabilities of some sources.

Thus, constraint mapping will find the mapping of Q for each source T_i, denoted $\mathcal{S}_i(Q)$, to retrieve the relevant subset of $\mathbf{R}_i$. Because of different source capabilities, a perfect mapping such that $Q \equiv \mathcal{S}_i(Q)$ often does not exist. Suppose for now that $\mathcal{S}_i(Q)$ may return extra answers (*i.e.*, it has false-positives) but may not miss any answers (*i.e.*, it has no false-negatives). The mediator then combines these source results, passes them through the conversion functions, and post-processes with a *filter* query $\mathcal{H}$ (to remove false-positives) consisting of the residue conditions not fully pushed to the sources, *i.e.*,

$$\mathcal{U} = \sigma_{\mathcal{H}}[\sigma_{\mathcal{S}_1(Q)}(\mathbf{R}_1) \times \cdots \times \sigma_{\mathcal{S}_n(Q)}(\mathbf{R}_n) \times \mathbf{X}]. \quad (2)$$

Comparing Eq. 1 and Eq. 2, we obtain the essential property for a *correct* translation:

$$Q = \mathcal{H} \wedge \mathcal{S}_1(Q) \wedge \cdots \wedge \mathcal{S}_n(Q) \quad (3)$$

We next illustrate this translation problem with Example 3, which considers a mediator that integrates two sources.

Example 3 Consider a mediator for two sources. Suppose that source T_1 provides relation `paper`(ti, au) for paper titles and authors and `aubib`(name, bib) for author names and their bibliography. The second source T_2 has `prof`(ln, fn, dept) for professor last, first names, and departments.

To illustrate, suppose that the mediator exports a faculty view `fac`(ln, fn, bib, dept) integrated from `aubib` and `prof`, and a publication view `pub`(ti, ln, fn) from `paper`(ti, au). In particular, the `fac` view joins `aubib` and `prof` through a conceptual relation (a conversion function) `NameLnFn`(name, ln, fn) with some view-definition conditions joining the name related attributes.

Suppose that a user is looking for the papers written by some CS faculty interested in data mining. The constraint query Q includes both selection and join constraints as the following. (Note that we omit the view-definition conditions in Q, as they will be processed in the mediator rather than the sources.)

$$Q = a{:}[\texttt{fac}.\text{ln} = \texttt{pub}.\text{ln}] \wedge\ b{:}[\texttt{fac}.\text{fn} = \texttt{pub}.\text{fn}] \wedge$$
$$c{:}[\texttt{fac}.\text{bib contains } \texttt{data}(\text{near})\texttt{mining}] \wedge$$
$$d{:}[\texttt{fac}.\text{dept} = \texttt{cs}].$$

Let's first consider the mapping for source T_1, *i.e.*, for relations `paper` and `aubib`. The join conditions $a \wedge b$ together map to x_1 : [`paper`.au = `aubib`.name]. If T_1 does not support the proximity operator near, rather than dropping constraint c, we can relax it to (x_2:[`aubib`.bib contains `data`] $\wedge$ x_3:[`aubib`.bib contains `mining`]), which requires only the occurrences of keywords. Lastly, constraint d maps to $True$ (it can only be processed in T_2). Thus, $\mathcal{S}_1(Q) = x_1 \wedge x_2 \wedge x_3$.

We next perform the mapping for source T_2, which contributes relation `prof`. All the constraints except d map to $True$. Suppose that T_2 uses department code 230 for CS, thus $\mathcal{S}_2(Q) =$ [`prof`.dept = `230`].

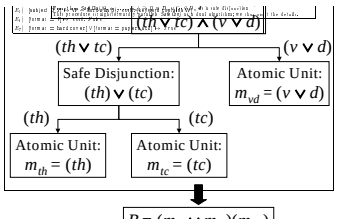
Finally, the filter query $\mathcal{H}$ is simply the constraint c (*i.e.*, $\mathcal{H} = c$), the only constraint that is not fully realized at the underlying sources. Thus, $Q = \mathcal{H} \wedge \mathcal{S}_1(Q) \wedge \mathcal{S}_2(Q)$. □

For an *exact* query processing that cannot miss any potential answers, $\mathcal{S}_u(Q)$ as the *closet mapping* of Q must logically subsume Q, *i.e.*, $\mathcal{S}_u(Q)$ will retrieve a superset of what Q does as just illustrated. However, in many other applications, a mediator may be willing to tolerate false-negatives as well as false-positives, *e.g.*, to explore more cost-effective native queries. Therefore, while post-filtering can remove false-positives, the result will be a subset of what Q would return were it supported. In other words, Eq. 3 becomes

$$Q \supseteq \mathcal{H} \wedge \mathcal{S}_1(Q) \wedge \cdots \wedge \mathcal{S}_n(Q) \quad (4)$$

This paper specifically addresses the constraint mapping problem, *i.e.*, translating Q into $\mathcal{S}_u(Q)$ which best approximates Q. The derivation of filter queries $\mathcal{H}$ is thus beyond the scope of this paper. A filter query can simply be the original query, *i.e.*, $\mathcal{H} = Q$, or it can consist of essentially only those constraints whose translated versions may retrieve false positives. Reference [36] discusses how to derive filters with the least processing cost.

As we can perform the mappings for different sources separately (as illustrated), we will focus on a particular source T_u as the translation *target* and discuss the requirements for


(th ∨ tc) ∧ (v ∨ d)
(th ∨ tc)
(v ∨ d)
Safe Disjunction:
(th) ∨ (tc)
Atomic Unit:
m_vd = (v ∨ d)
(th)
(tc)
Atomic Unit:
m_th = (th)
Atomic Unit:
m_tc = (tc)