Exposing the Vagueness of Query Results on Partly Inaccessible Databases

Oliver Haase
Bell Labs Research
Holmdel, NJ 07733-3030, USA
oli@bell-labs.com

Andreas Henrich
Otto-Friedrich University of Bamberg
D-96045 Bamberg, Germany
andreas.henrich@sowi.uni-bamberg.de

ABSTRACT

Query processing on partly inaccessible databases generally does not yield *exact,* but vague result sets. A good notion of vague sets fulfills two aims: It keeps the degree of vagueness of the query result as small as possible, and it clarifies the degree of and the reasons for the vagueness to the end user. The first goal requires a good internal representation, while the second goal requires a good external representation of a vague set. In this paper, we present a novel calculus for expressive vague *sets* that meets both requirements. This is the first approach that is well suited for both internal and external representation of vagueness induced by partial inaccessibility. It consists of a data representation that is capable of holding all the necessary information. Complementary, we have accordingly adapted the usual query language operations. These adaptations are independent of a concrete query language, to make them applicable to most existing query languages. The adapted operations minimize the vagueness of the result, propagate the reasons of uncertainty of the individual vague candidates, and compute an expressive description of the missing elements.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval-Search process, *Selection Process;* E.1 **[Data Structures]:** Distributed Data Structures

General Terms

Algorithms

Keywords

Query Processing, Inaccessibility, Vagueness

1. MOTIVATION

There are many application areas, where a distibuted database has to remain operable even if part of the database is inaccessible due to a network or a machine failure; or because part of the database is stored on a laptop, which is temporarily disconnected from the network. These situations are even considered normal, e.g., in the distribution model of PCTE [11], the IS0 and ECMA standard for an open repository; related situations may arise in mobile environments, as described in [8]. Furthermore, the ever increasing Internet is hosting many distributed databases [6, 7] for which due to the unreliable nature of data sources in the Internet — partial inaccesibility is simply a matter of fact. Another reason for partial inaccessibility in the Internet is network congestion; if a server cannot be accessed within a certain time period, it must be regarded inaccessible.

As a consequence, a distributed database system for environments like these has to take into consideration that the environment may become partitioned into sub-environments that cannot communicate with each other. Nevertheless, the sub-environments have to remain operable, in particular queries to the database have to be processed appropriately. For this purpose, inaccessibility has to be accordingly dealt with during query processing. First, adequate inference rules have to be used that keep the vagueness as small as possible during query processing. Second, the actual vagueness of the query result must be exposed expressively to the user.

In previous papers [3, 5, 4] we have proposed a solution for the first aspect. This solution is based on hybrid representations for vague sets, vague multisets, and vague lists, which address the accessible elements in an enumerating *component* and the inaccessible (or unreachable) elements in a *descriptive* component. The main benefit of this approach is that the descriptive component of a vague collection can be exploited to improve the enumerating component during query processing. Although these hybrid representations — which will be summarized in section 2 — are very beneficial during query processing, they do neither directly state the reasons for the uncertainty of individual elements, nor do they characterize the missing elements.

In order to overcome these shortcomings, we have developed a suitable representation of *expressive vague sets, as* well as appropriate adaptations of the usual query operations. An expressive vague *set* comprises the following information:

1. For each uncertain element of a set the reasons for the uncertainty are maintained in a predicate representing its uncertain properties, i.e. the properties which could not be evaluated unambiguously due to partial inaccessibility of the database.

2. In addition, a so-called *warrant* of apprehension is maintained for an expressive vague *set.* In contrast to the descriptive component of a vague set which describes the properties of all relevant objects including the explicitly enumerated elements, this warrant of *apprehension* describes only the missing elements.

   The **warrant of apprehension** informs the user about the degree of incompleteness of a vague result set — e.g. if the user is told that all missing elements reside on a certain server. In addition, it can be used to search for the missing elements later, when the accessibility situation will have changed.

Since this approach leverages, extends, and complements the basic ideas of the previously proposed representations of vague *sets,* it can best be introduced by shortly summarizing our hybrid representation of vague sets. Hence, we will sketch the idea of vague sets in section 2 before introducing expressive vague *sets* in section 3. Section 4 will consider related approaches, and section 5 concludes the paper.

2. VAGUE SETS

For a comprehensive calculus for vague sets, we need both a suitable representation, as well as appropriate operations.

2.1 Representation

From an abstract perspective, a query evaluated on a partly inaccessible distributed database partitions the database in four types of elements:

1. *Accessible* elements for which we can decide that they surely belong to the result set, because all query conditions can be tested completely based on the accessible parts of the database.
2. *Accessible* elements for which we can decide that they surely do not belong to the result set.
3. *Accessible* elements for which we cannot decide whether or not they belong to the result set, because the evaluation of at least one query condition would require access to the inaccessible part of the database.
4. Inaccessible elements.

Unfortunately, the situation gets even more complex if we take procedural query processing into account. Then, we are faced not only with inaccessible, but also with accessible still unreachable elements. An element i is unreachable, if query evaluation would require the traversal of a path via at least one inaccessible element to get i. Please note, that such situations are conceivable with object-oriented databases when following relationships, as well as with relational databases when evaluating joins.

To deal with this additional source of uncertainty, the hybrid representation of a vague set V comprises:

- An enumerating component which is made up of an *explicit* lower bound $\widehat{V}_\lambda$ and *an explicit upper* bound $\widehat{V}_\upsilon$. The explicit lower bound contains the accessible and reachable elements which surely belong to the result, while the explicit upper bound contains the accessible and reachable elements for which we cannot be sure that they do not belong to the result, e.g. because the evaluation of a condition would require access to the inaccessible part of the database.
- A *descriptive* component δ_V, which defines the properties of all relevant elements including those we could not access. To represent the descriptive component, we use a logical predicate with one free variable, written as $\delta_V \in \mathrm{Pred}_1$. As a further convention, we use the notation $'\delta_V(i)'$ to indicate that i is the (only) free variable in δ_V. The descriptive component states for each element j of the foundation set, if j surely *belongs* to the correct answer (i.e. δ_V evaluates to $\star$ for j: $[\delta_V(j)] = t$), if it *maybe* belongs to the correct answer ($[\delta_V(j)] = u$), or if it surely *does* not *belong* to the correct answer ($[\delta_V(j)] = \mathrm{f}$).

  As can be seen, due to partial inaccessibility of the database the descriptive component must be evaluated three-valued.

  This predicate can be used to check for concrete elements, whether they belong to the result, or not.

Formally, a vague set over the foundation set T *can* be defined as follows:

Definition 1 (vague sets). The *triple* $V = (\widehat{V}_\lambda, \widehat{V}_\upsilon, \delta_V(i))$ *with* $\widehat{V}_\lambda \subseteq T$, $\widehat{V}_\upsilon \subseteq T$, $\delta_V \in Pred_1$ is a vague set over *the* foundation *set* T *($V \in \widetilde{\mathbf{Set}}(T)$), iff*

$$\widehat{V}_\lambda \subseteq \widehat{V}_\upsilon$$
$$\text{A} \quad \textit{for all } j \in \widehat{V}_\lambda : [\delta_V(j)] = t$$
$$\text{A} \quad \textit{for all } j \in \widehat{V}_\upsilon \setminus \widehat{V}_\lambda : [\delta_V(j)] = u$$

An exact *set* $S \subseteq T$ is contained in *V,* iff all elements of $\widehat{V}_\lambda$ are *also* in S and δ_V evaluates *at* least *to* $'u'$ *for* all elements in S.

To demonstrate the adequacy of this representation, an example is useful:

Example 1. *Assume the distributed* database *depicted in* **figure 1.** *This* might *be* an *object-oriented* database *with a link concept,* **or a** *hypertext system consisting of HTML* pages or *XML* documents. Now, *consider the* query *q:*

> Select all groups of all departments at the university in "Bamberg" which have at least one member playing golf.

evaluated on *this* partly inaccessible *database. Obviously, the* correct **answer** *to this* query would *be the set* containing groups g_1, g_3, and g_5.

Since server 3 is inaccessible, we cannot *compute this result.* Instead, normal *inside-out* evaluation *starts with* university u_1, navigates *to the (accessible)* departments d_1 and d_3, and further *to their respective* groups g_1, g_2, g_5 and g_6. Because g_1 has *the* member m_1 playing *golf,* it *surely* belongs *to the* query result; because m_3, which is *the only* member *of* g_2, has *the hobby* "fishing", g_2 surely does not *belong to the* result; and because g_5 and g_6 *both* have *inaccessible* members, *but* no accessible members playing *golf,* we cannot *decide whether or* **not** *they belong to the query result. Due to the* inaccessibility *of server* 3 *the* link from u_1 *to* d_2 cannot *be* traversed. *As* an *effect,* groups g_3 and g_4 cannot *be* reached, although *they* **are** accessible. Moreover, *the* query **processor** cannot know, how many *unreachable* candidates exist.

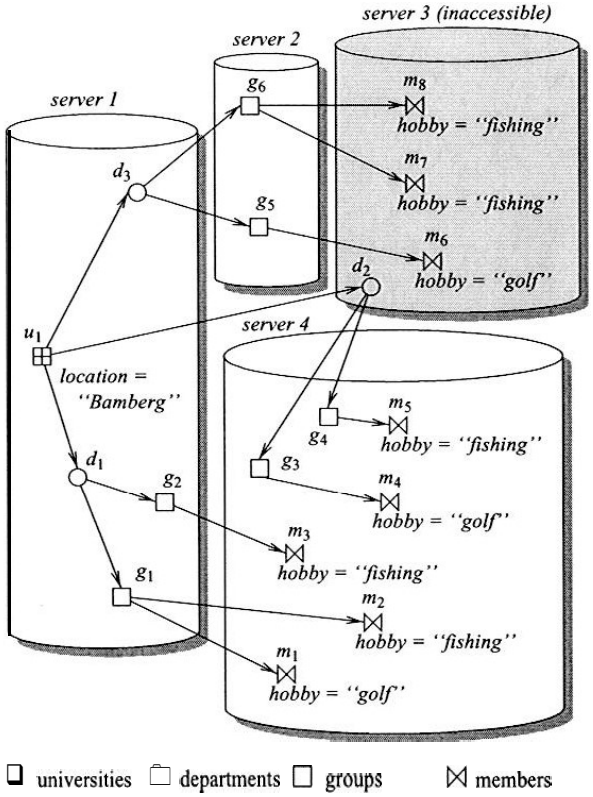

Figure 1: An example database

To sum up, the explicit lower bound $\hat{V}_\lambda$ of the vogue result set V comprises only g_1, while the explicit upper bound $\hat{V}_v$ consists of g_1, g_5, and g_6.

The descriptive component $\delta_V(g)$ of the "ague result V requires (1) that a candidate g must be of type group, (2) that it must hove on incoming link from a department d, (3) that d must have an incoming link *from* a university *u, (4) that the value of the attribute* location *of u must be "Bamberg", (5) that g must have an outgoing link to a member m, and (6) that the value of the attribute* hobby *of m must be "golf".*

As will be seen in the following section, the descriptive component of this hybrid representation can be exploited during query evaluation to keep the number of accessible, but unreachable elements as small as possible.

2.2 Query Operations

The adaptations of the usual query operations to vague sets state the rules for the step-wise computation of the enumerating and the descriptive component of a vague result set. Here, we do not present the individual query operations; the interested reader is referred to [5]. Instead, we show by the example of vague set intersection the key technique for the minimal propagation of vagueness during query evaluation. The same principle will be incorporated into the treatment of *expressive* vague sets, as will be seen in section 3.

The computation of the result X of the intersection $V \cap_3 W$ of two vague sets *V* and *W consists* of two parts: the computation of the enumerating component, and the cornputation of the descriptive component. Please note that we use the index "3", if we refer to operations on vague sets. If an operator is written without index, we always refer to the exact, completely accessible case [1].

Let us start with the descriptive component δ_X. Since each element of X must be an element of V *and* an element of W, δ_X is built as the logical conjunction of δ_V and δ_W:

$$\delta_X(i) = S''(i) \wedge \delta_W(i)$$

For the calculation of the enumerating component of X, the situation is a bit more complicated. If we *did not use a* descriptive component in our representation, we would compute the explicit *lower* bound $\hat{X}_\lambda$ of X as the intersection of $\hat{V}_\lambda$ and $\hat{W}_\lambda$, because only elements that are certainly contained in both input sets are also certainly contained in the result. For the explicit upper bound $\hat{X}_v$ of X, the following consideration is crucial (still assuming we did not use a descriptive component): Each element of $\hat{V}_v$ could be a (unreachable) candidate of $\hat{W}_v$ — and consequently of $\hat{X}_v$ —, and vice versa. Hence, we would have to build the **union** of $\hat{V}_v$ and $\hat{W}_v$ **to construct** $\hat{X}_v$.

The situation *with* descriptive components is different: Here, we can apply δ_V to each element of the explicit upper bound $\hat{W}_v$ of *W*, to check if the element actually is a candidate for $\hat{V}$. *(or even V_A)*, and vice versa. Those elements *i* with $[\delta_V(i)] = f$ can explicitly be excluded — in contrast to the situation *without* descriptive component. Obviously, a vague set with less wrong candidates is more precise than one with more wrong candidates.

The above considerations lead to the following rules for the computation of the enumerating part of X:

$$\begin{aligned}\hat{X}_\lambda &= (\hat{V}_\lambda \cup \{i \in \hat{W}_v \mid [\delta_V(i)] = t\}) \\ &\quad \cap (\hat{W}_\lambda \cup \{i \in \hat{V}_v \mid [\delta_W(i)] = t\}) \\ &= \{i \in \hat{V}_v \cup \hat{W}_v \mid [\delta_V(i)] = t\} \wedge [\delta_W(i)] = t\})\end{aligned}$$

$$\begin{aligned}X_v &= (\hat{V}_v \cup \{i \in W_v \mid [\delta_V(i)] \neq f\}) \\ &\quad \cap (\hat{W}_\lambda \cup \{i \in \hat{V}_v \cap [\delta_W(i)] \neq f\}) \\ &= \{i \in \hat{V}_v \cup \hat{W}_v \cap [\delta_V(i)] \neq f\} \wedge [\delta_W(i)] \neq f\})\end{aligned}$$

Example *2. To demonstrate the usefulness of the above definition,* let *us consider the* **intersection** *$X = V \cap_3 W$ where V is the vogue result set from example 1 with $\hat{V}_v = \{g_1, g_5, g_6\}$), and W is a vague set whose explicit upper bound contains all accessible group objects, i.e. $\hat{W}_v = \{g_1, g_2, g_3, g_4, g_5, g_6\}$. Without a* descriptive *component — knowing only that V is incomplete — we would have to use $\hat{X}_v = \{g_1, g_2, g_3, g_4, g_5, g_6\}$. The application of the descriptive component of V allows us to restrict $\hat{X}_v$ to $\{g_1, g_3, g_5, g_6\}$, because g_2 and g_4 do not have a member playing "golf".*

As we have seen in this section, the hybrid representation of vague sets (together with the adaptations of the usual query operations) is well suited to minimize the vagueness

[1] To avoid unnecessarily complex syntax, we use the index "3" only to indicate the vague adaptation of a query operator. Logical junctions are noted without this index, because their three-valued evaluation should be obvious in the respective context.

during query processing. On the other hand, this representation lacks an expressive explanation of the degree of and the reasons for the vagueness induced by partial inaccessibility. In order to overcome this shortcoming, the next section presents our approach for expressive vague sets.

3. EXPRESSIVE VAGUE SETS

The representation of vague sets presented in the previous section was intended to optimize the quality of the query result on a partly inaccessible database. However, the end user should and can be given an expressive *explanation* of the induced vagueness. This explanation consists of two parts:

1. For each uncertain candidate in the explicit upper bound of the expressive vague set, we indicate the reason, why we could not include the candidate in the explicit lower bound. We call this information the uncertain properties of a candidate.

   To become more concrete, all uncertain candidates i of a (expressive) vague set V ($i \in V_v \setminus V_\lambda$) have in common that due to partial inaccessibility some query condition could not be evaluated. In terms of vague sets as presented in section 2, this means that we cannot decide whether the two-valued evaluation of the descriptive component for i would yield t or f.

   The descriptive component states exactly the properties a candidate i must fulfill to qualify as an element of the correct query result. However, in the general case, not all sub-predicates of the descriptive component represent uncertain properties of i; in contrast, those sub-predicates which are evaluable even in the case of partial inaccessibility, are for sure. If we replace these evaluable sub-predicates of the descriptive component by their respective truth values and simplify the resulting predicate, we will get a simplified predicate which states exactly the uncertain properties of i. Or to phrase it differently: If i fulfills this new predicate, it belongs to the desired query result. We will denote these uncertain properties with UP_V in the following.

2. We describe the missing elements. The description of these elements is called the *warrant of apprehension.* In contrast to the descriptive component of a vague set which describes the properties of *all* relevant elements including the explicitly enumerated elements, this *warrant* of apprehension describes only the missing elements.

   For a concrete query a good description of the characteristics of the missing elements might be something like *"there are* some elements missing in *the* result and *these* elements *either reside* on server s_x, or *they* can *be* reached from an element in *the set* Y via a path traversing server s_x".

   The warrant of apprehension informs the user about the degree of incompleteness of a vague result set — e.g. if the user is told that all missing elements reside on a certain server. In addition, it can be used to search for the missing elements later, when the accessibility situation will have changed. We will denote the warrant of apprehension with $\$_V$ in the following.

While the uncertain properties of the uncertain candidates can be derived from the descriptive component of a vague set (though this would be unnecessarily inefficient), the warrant of apprehension is generally not derivable from a vague set. This part of the vagueness explanation requires an extended data representation.

3.1 Representation

In this section we propose the formal representation of expressive vague sets.

Analogous to a vague set, *an* expressive vague set contains an *explicit* lower *bound* and an *explicit* upper bound. The descriptive component is replaced by the warrant of apprehension which describes only the missing elements and therefore clarifies the degree of incompleteness of the result. More precisely, the warrant *of* apprehension is a predicate which would — in case of complete accessibility yield t for an element which belongs to the result but is missing in the enumerating component. For the elements in the enumerating component and for those elements which should not be in the result it would yield f. Due to inaccessibility, it can also yield u if a condition cannot be evaluated.

As an additional component, an expressive vague set also contains one predicate per explicit element $i \in V_v$ indicating its uncertain properties. To keep the technique as simple as possible, not only the uncertain candidates ($i \in \widehat{V}_v \setminus \widehat{V}_\lambda$) but also the sure elements ($i \in \widehat{V}_\lambda$) get such a predicate assigned. Of course, the uncertain properties of a sure element $i \in \widehat{V}_\lambda$ is simply the truth value t.

In order to better distinguish expressive vague sets from vague sets, we will use caligraphic letters A, . . . , 2 for expressive vague sets in the following.

Definition 2 (expressive *vague* sets). *The quadruple* $v = (\widehat{V}_\lambda, \widehat{V}_v, \$_V, UP_V)$ *with* $\widehat{V}_\lambda \subseteq T$, $\widehat{V}_v \subseteq T$, $\$_V \in Pred_1$, $UP_V : \widehat{V}_v \to Pred_1$ *is* an expressive vague set over *the* foundation *set* T ($V \in \widetilde{\mathbf{Set}}_e(T)$), *iff* [2]

$$\widehat{V}_\lambda \subseteq \widehat{V}_v$$

$$\mathbf{A} \quad \text{for all } i \in \widehat{V}_\lambda : [\,UP_V(i)] = t$$

$$A \quad \textit{for all } i \in \widehat{V}_v \setminus \widehat{V}_\lambda : [UP_V(i)] = u$$

$$\mathbf{A} \quad \text{for all } i \in \widehat{V}_v : [\$_V(i)] = f$$

In this definition, the mapping UP_V maps each element of $\widehat{V}_v$ to a predicate representing its uncertain properties.

Now, that we have defined our notion of an expressive vague set, we have to show how a query on a partly inaccessible database can be transformed into an equivalent expressive vague set.

3.2 Query Operations

Before we can discuss the individual query operations, we have to do some preparatory work:

The computation of the uncertain properties of an uncertain candidate of an expressive vague set requires the partial evaluation of a logical predicate. This partial evalu-

[2] Note that $\mathrm{UP}_V(i)$ yields a predicate with a free variable. As a consequence, $[\mathrm{UP}_V(i)]$ stands for the evaluation of this predicate for the element i.

ation eliminates the evaluable sub-predicates of a complex predicate. Instead of the complete evaluation of a predicate which yields one of the truth values 't', 'f', or 'u', the partial evaluation yields a new, simplified predicate. Informally speaking, partial evaluation means to insert the respective truth values t and f for the evaluable sub-predicates, and to simplify the resulting predicate.

Only the partial evaluation of an existence- or all-quantified predicate is a bit more complicated, because in our context, the foundation set $\mathcal{V}$ in '$\exists k \in \mathcal{V}$: p(k)' or '$\forall k \in \mathcal{V}$: p(k)' will itself be an expressive vague set, since $\mathcal{V}$ is usually the vague result of a subquery evaluated on the partly inaccessible database. Consequently, we have to deal with a predicate p of the form '$p(i) = \exists k \in \mathcal{V} : q(k,i)$' or 'p(i) = Vk $\in V$: $q(k,i)$' where $\mathcal{V}$ is the result of a vague subquery and $q(k,$ i) stands for a predicate testing the desired relationship between k and i (e.g. some type of join-condition).

Let us discuss the consequences for existence-quantified predicates; for all-quantified predicates, similar considerations hold true:

If the foundation set is an *exact set,* i.e. $V = \{l_1, \ldots, l_n\}$, then partial evaluation of '$\exists k \in V$: $q(k,$ i)' will mean (1) to transform the predicate to a disjunction '$\bigvee_{k \in \{k_1,\ldots,k_n\}} q(k, i)$', and (2) to partly evaluate this disjunction through insertion of a different l_i per k_i, and final simplification[3]. Formally, this will lead to $[p(j)]^p \stackrel{\text{def}}{=} [\bigvee_{l \in V} q(l,j)]^p$.

If the foundation set is a complete expressive vague set V, i.e. with vague candidates but no missing elements, then we will have to indicate the circumstances in which the partial evaluation of q for an element l_i is really relevant for the partial evaluation of the complete quantified predicate; this is the case if the uncertain properties of the respective element evaluate to true, i.e. if the candidate really belongs to V. Consequently, we have to conjoin the individual partial evaluations of q with the uncertain properties of the respective element l_i. This technique *"weights"* the influence of the individual sub-parts to the complete partial evaluation. As a consequence, for $p(i) = \exists k \in \mathcal{V} : q(k,i)$ we can define the partial evaluation for the element j as $[p(j)]^p \stackrel{\text{def}}{=} [\bigvee_{l \in \hat{V}_v} \text{UP}_V(l) \wedge q(l,j)]^p$.

Finally, if the foundation set is *an incomplete* expressive vague set $\mathcal{V}$, we will have to take into consideration the contribution of the missing elements of V. Therefore, we have to complete the partial evaluation of p(i) = $\exists k \in \mathcal{V}$: $q(k, i)$ to $[p(j)]^p \stackrel{\text{def}}{=} [\bigvee_{l \in \hat{V}_v} \text{UP}_V(l) \wedge q(l,j)]^p \vee (\exists k : \$_V(k) \wedge q(k,i))$.

This leads to the following formal definition of partial evaluation.

Definition 3 (partial evaluation). *Let $p(i_1, \ldots, i_n)$ be an n-ary predicate ($p \in Pred_n$); $j_1, \ldots, j_n$ elements of the* foundation *set T. Further, let i be the* vector $(i_1, \ldots, i_n)$, and j *the vector* $(j_1, \ldots, j_n)$.

The partial evaluation $[p(j)]^p$ *of p for j is defined* inductively over *the* structure of p:

. p(i) *atomic:*

$$[p(j)]^p \stackrel{def}{=} \begin{cases} t \text{ , if } [p(j)] = t \\ f \text{ , if } [p(j)] = f \\ p(i) \text{ , if } [p(j)] = u \end{cases}$$

- $p(i) = q(i)\ A$ r(i):

$$[p(j)]^p \stackrel{def}{=} \begin{cases} f, \text{if } [q(j)]^p = f \text{ or } [r(j)]^p = f \\ [r(j)]^p, \text{if } [q(j)]^p = t \\ [q(j)]^p, \text{if } [r(j)]^p = t \\ [r(j)]^p \text{ A } [q(j)]^p, \text{otherwise} \end{cases}$$

- $p(i) = q(i)$ V r(i):

$$[p(j)]^p \stackrel{def}{=} \begin{cases} t, \text{ if } [q(j)]^p = t \text{ or } [r(j)]^p = t \\ [r(j)]^p, \text{if } [q(j)]^p = f \\ [q(j)]^p, \text{if } [r(j)]^p = f \\ [r(j)]^p \text{ V } [q(j)]^p, \text{otherwise} \end{cases}$$

- $p(i) = \neg q(i)$:

$$[p(j)]^p \stackrel{def}{=} \begin{cases} f, \text{if } [q(j)]^p = t \\ t, \text{if } [q(j)]^p = f \\ \neg [q(j)]^p, \text{otherwise} \end{cases}$$

- $p(i) = \exists k \in \mathcal{V} : q(k,i)$:

$$[p(j)]^p \stackrel{def}{=} [\bigvee_{l \in \hat{V}_v} UP_V(l) \wedge q(l,j)]^p \vee (\exists k : \$_V(k) \wedge q(k,i))$$

- $p(i) = \forall k \in \mathcal{V} : q(k,i)$:

$$[p(j)]^p \stackrel{def}{=} [\bigwedge_{l \in \hat{V}_v} UP_V(l) \wedge q(l,j)]^p \wedge (\forall k : \$_V(k) \wedge q(k,i))$$

---

[3] Please note, that i and k stand for formal variables, while j and l stand for the concrete elements for which the partial evaluation is performed.

As we will see in the following, we will use the partial evaluation of certain predicates to compute the uncertain properties of the vague candidates of an expressive vague set.

3.2.1 Start Sets

Queries in all types of query languages start from one or several independent initial sets of elements called **start sets** in this paper. In SQL, e.g., these start sets are defined in the **from-clause** of a query. From an abstract point of view, a start set consists of all elements in the database which fulfill a certain predicate. In practice, this predicate is often related to the element's type, like, e.g., in SQL.

With this understanding in mind, we can define an operator $\textcircled{\$}_3$ which returns a start set computed on a partly inaccessible database.

Let T denote the (infinite) set comprising all potential elements for the underlying datamodel, and let 0 denote the set of all elements actually contained in the database at the time of query evaluation including the elements residing on the inaccessible parts. Then the signature of the operator $\textcircled{\$}_3$ which receives a database and a predicate and yields an expressive vague start set would be $\textcircled{\$}_3 : T \times \text{Pred}_1 \rightarrow \widetilde{\text{Set}}_e(T)$. However, since the first operand of $\textcircled{\$}_3$ would always be 0, we omit it in the following and get $\textcircled{\$}_3 : \text{Pred}_1 \rightarrow \widetilde{\text{Set}}_e(T)$. A typical instantiation would be

$\textcircled{S}_3(\text{type}(i)$ = group) denoting that the start set is to include all elements of the entire database with type group.

Based on these notations, we define the rules for the computation of an expressive vague start set. These rules are not completely independent of the underlying datamodel and distribution model. However, we can still formulate helpful, widely applicable rules. Necessary adaptations to a concrete datamodel must be made on a case-by-case basis.

The explicit lower bound $\hat{V_\lambda}$ of an expressive vague start set $\mathcal{V} = \textcircled{S}_3(p)$ comprises all accessible elements of the database, for which the predicate p is evaluable and evaluates to t. Analogously, the explicit upper bound $\hat{V}_\upsilon$ comprises all accessible elements of the database, for which the predicate p either evaluates to t, or is not evaluable. In the following, let $\{s_1, \ldots, s_n\}$ denote the set of inaccessible segments. A segment is intented to be a very abstract concept; in a real database, this can, e.g., be a particular server, a logical cluster of servers, or a relational table. Additionally, let the function *loc* denote the segment an element is residing on. Then, $\hat{V}_\lambda$ and $\hat{V}_\upsilon$ can be computed as follows:

$$\begin{aligned} V_\lambda &= \{j \in 0 \mid [\text{loc}(j) \notin \{s_1, \ldots, s_n\} \text{ A } \mathbf{P(j)}] = t\} \\ \hat{V}_\upsilon &= \{j \in 0 \mid [\text{loc}(j) \notin \{s_1, \ldots, s_n\}_{\text{A}} \mathbf{P(j)}]_{\text{-}} = u\} \end{aligned}$$

For type extensions, such as the above example @(type(i) = group), the lower and the upper explicit bounds typically both contain the same set of elements. This is true if the assumption holds that whenever au element is accessible also its type information is accessible. If this is not the case, $\hat{X}_\upsilon$ will additionally contain those elements for which the type information cannot be accessed. Although this situation might seem rather academic for type information, it is more realistic for start sets using other predicates.

This leads us to the definition of the uncertain properties for the elements in $\hat{V}_\upsilon \setminus \hat{V}_\lambda$. For an expressive vague start set $V = \textcircled{S}_3(p)$, au element i is an *uncertain candidate,* i.e. $i \in \hat{V}_\upsilon \setminus \hat{V}_\lambda$, if part of the predicate p cannot be evaluated. Hence, the uncertain properties are exactly the partial evaluation of the predicate p. Please note, that p might be a complex predicate, only parts of which are unevaluable for a particular element:

$$\text{UP}_V(j) = [p(j)]^p$$

Finally, for the definition of the warrant of apprehension we can distinguish two cases: (1) The start set has been computed *completely,* and the query processor is aware of it. This is the case if either the entire database is accessible, or the inaccessible part cannot possibly host any relevant elements. (2) The start set has not been computed completely, or the query processor cannot be sure if there are any elements missing.

If some elements are (or could be) missing, they have to fulfill the predicate p, while they are residing on an inaccessible segment:

$$\$_V(i) = \begin{cases} f, \text{ if } V \text{ is complete} \\ p(i) \text{ A } \text{loc}(i) \in \{s_1, \ldots, s_n\}, \text{ otherwise} \end{cases}$$

It is worth mentioning that especially the inclusion of the locality information $\text{loc}(i) \in \{s_1, \ldots, s_n\}$ enhances the expressiveness of the warrant of apprehension.

In so-called *semantic,* navigational datamodels, or in highly distributed, e.g. Web-based databases, queries often start with one single *root* object, instead of an entire set of objects. The rules above also apply to start sets containing only one particular element, as shows the next example:

Example 3. Assume, we have *accessed the object* u_1 *(representing* Bamberg *University) of* our example *database given* in figure 1 *by* some *type of* navigational operation, and we *want to formulate* a *query relative to this object as its* starting *point. In this* case, *the start set* N *of the* query would have *the* representation $\hat{N}_\lambda = \{u_1\}; \hat{N}_\upsilon = \{u_1\}; UP_N(u_1) = t; \$(i) = f$. *We will use this start set* as *the* basis for our running example.

3.2.2 *Navigation*

The next operation, *set-wise* navigation, is highly vulnerable to partial inaccessibility, as the destination elements can reside on the inaccessible part of the database. We will spend this operation special consideration.

Navigation is based on the traversal of a link in an object-oriented database or in a hypertext environment. The operation starts at a given source set, traverses a specified set of links per source element, and returns the union of all destination elements. It can be considered a special kind of relational semi-join $R_1 \ltimes_\Theta R_2$, where the join predicate Θ is materialized as links. Semi-join in turn can be expressed as the concatenation of normal join and projection to the attributes of R_2.

What is important in our context is, that the join predicate, e.g. equality of the identically named attributes for a natural semi-join, must be materialized and stored together with the respective originating elements. Then, in order to find the peers of the source set, we do not need to scan the entire set of potential destination elements, but we can just follow physical links, pointing from a source element o_1 to a destination element o_2, denoted as $o_1 \rightarrow o_2$. This is a crucial feature of most highly distributed datamodels. In addition, we assume that links are typed (with the link type corresponding to the semi-join predicate Θ), and we use the notation $o_1 \xrightarrow{lt} o_2$ if a link of type lt points from o_1 to o_2. As a final syntactical convention, the set of all destination elements reachable via links of type lt that originate at a certain element o is denoted by $(o \xrightarrow{lt})$.

In the two-valued, completely accessible case, navigation $\ltimes$ (S, *It)* gets a source set S and a link type lt as input, and has the semantics $\ltimes(S, It) = \bigcup_{i \in S}(i \xrightarrow{lt})$.

In case of partial inaccessibility, three-valued navigation $\mathcal{X} = \ltimes_3(\mathcal{V}, It)$ gets an expressive vague set V and a link type lt as input, and produces as output an expressive vague set $\mathcal{X}$, whose computation will be discussed in the following. For this discussion, it is important to note that the set $(i \xrightarrow{lt})$ of destination elements, reachable from i via links of type lt, generally cannot be computed completely, because some destination elements might reside on the inaccessible part of the database. Similar to the enumerating component of a vague set, we denote those elements that are accessible with $\widehat{(i \xrightarrow{lt})}$.

For the computation of the explicit lower bound $\hat{X}_\lambda$, we compute the set of all accessible destination elements of each element of the explicit lower bound of $\mathcal{V}$, and we build the

union of these sets. The computation of the explicit upper bound $\hat{X}_v$ is done analogously.

$$\hat{X}_\lambda = \bigcup_{i \in \hat{V}_\lambda} \widehat{(i \xrightarrow{lt})} \quad ; \qquad \hat{X}_v = \bigcup_{i \in \hat{V}_v} \widehat{(i \xrightarrow{lt})}$$

For each uncertain candidate j of $\mathcal{X}$, there are two possibilities that j really belongs to X: Either (a) one of the source elements i with $i \xrightarrow{lt}$ j — that can only be uncertain candidates in $\mathcal{V}$, otherwise j would be in $\hat{X}_\lambda$ — really belongs to $\mathcal{V}$, or (b) there is an element missing in $\mathcal{V}$ that refers to j.

This consideration leads to the rule for the computation of the uncertain properties of the elements of X:

$$\mathrm{UP}_X(j) = \underbrace{\left(\bigvee_{\{i \in \hat{V}_v \mid j \in \widehat{(i \xrightarrow{lt})}\}} \mathrm{UP}_V(i) \right)}_{\text{covers case (a)}} \vee \underbrace{\left(\exists i : \$_V(i) \wedge j \in \widehat{(i \xrightarrow{lt})}\right)}_{\text{covers case (b)}}$$

Missing elements in $\mathcal{X}$ are all those elements for which there is an origin element in the enumerating component of V but that reside on the inaccessible part of the database, or for which all origin elements in V are missing themselves:

$$\$_X(j) = (\exists i \in \hat{V} : i \xrightarrow{lt} j \wedge \mathrm{loc}(j) \in \{s_1, \ldots, s_n\}) \vee (\exists i : \$_V(i) \wedge i \xrightarrow{lt} j \wedge \forall k \in \hat{V}_v : j \notin (k \xrightarrow{lt}))$$

We illustrate the rules for the computation of vague navigation by means of an example:

Example 4.

Coming back to our *query* "Select all groups of all departments at the university in 'Bamberg' which have . . ." *from* example 1, we can now navigate in *two steps* from *the object* u_1 representing *the* university in *"Bamberg" to the related* group *objects. As* a *first step* we consider *the query* $\mathcal{O} = \rtimes(\{u_1\}, dep)$, i.e. $\mathcal{O} = \rtimes(\mathcal{N}, dep)$ with respect *to* example 3, searching for all departments *which* can *be* reached from u_1. *We get:*

$$\hat{O}_\lambda = \{d_1, d_3\};\ 6, = \{d_1, d_3\}$$
$$UP_O(d_1) = t;\ UP_O(d_3) = t$$
$$\$_O(j) = u_1 \xrightarrow{dep} j \ A\ loc(j) = server\ 3$$

When we extend our example query further *to* $\mathcal{P} = \rtimes(\rtimes(\{u_i\}, dep), group)$ *we get:*

$$\hat{P}_\lambda = \{g_1, g_2, g_5, g_6\};\ \hat{P}_v = \{g_1, g_2, g_5, g_6\}$$
$$UP_P(g_1) = t;\ UP_P(g_2) = t$$
$$UP_P(g_5) = t;\ UP_P(g_6) = t$$
$$\$_P(j) = (\exists i \in \{d_1, d_3\} : i \xrightarrow{group} j \ \mathrm{A}\ loc(j) = \text{server 3}) \vee (\exists i : u_1 \xrightarrow{dep} i \ \mathrm{A}\ loc(i) = server\ 3 \wedge i \xrightarrow{group} \mathrm{j}\ \mathrm{A}\ \mathrm{Vk} \in \{d_1, d_3\} : j \notin (k \xrightarrow{group}))$$

Please note, that $\$_P(j)$ might not seem to be an easily readable description of the missing elements. However, $\$_P(j)$ contains all conceivable information about the missing elements, and can be automatically transformed into a human-readable description such as: "Elements with *the* following properties *might be* missing: (I) Group *objects* residing on server 3 which are related *to the* department *objects* d_1 or d_3. (2) Group objects that can *be* reached from a department *object* on server 3 *which is linked to* u_1 . " Additionally, graphical representations of the corresponding relationships could further illustrate the formal warrant of apprehension.

3.2.3 *Selection*

Another important query language operator is selection σ. Its adaptation to expressive vague sets maps an expressive vague input set and a selection predicate to an expressive vague result set, i.e. $\sigma_3 : \widetilde{\mathbf{Set}_e}(T) \times \mathrm{Pred}_1 \rightarrow \widetilde{\mathbf{Set}_e}(T)$. In order to name the individual elements of the vague input set — which are checked against the selection predicate — we bind them to an element variable. Thus, vague selection is denoted as $\mathcal{X} = \sigma_3(e : \mathcal{V}, p(e))$.

The computation of the expressive vague result set $\mathcal{X}$ is straightforward. We start with the enumerating component: For each explicit element $\mathrm{j} \in \hat{V}_v$ we check p, in order to decide whether or not j belongs to the result.

$$\hat{X}_\lambda = \{j \in \hat{V}_\lambda \ \ [p(j)] = t\}$$
$$\hat{X}_v = \{j \in \hat{V}_v \, I \, [p(j)] \neq f\}$$

The warrant of apprehension of the result is the logical conjunction of the warrant of apprehension of the expressive vague input set and the predicate p, because a missing element must obey both conditions.

$$\$_X(i) = \$_V(i) \wedge p(i)$$

Finally, the uncertain properties of the explicit elements of $\mathcal{X}$ are the partial evaluation of the conjunction of the uncertain properties of the input and the predicate p.

$$\mathrm{UP}_X(j) = [\mathrm{UP}_V(j) \wedge p(j)]^p$$

Example 5. *As* an example *for* selection we reconsider our example *query* "Select all groups of all departments at the university in 'Bamberg' which have at least one member playing golf" :

$$\mathcal{Q} = \sigma(\ g : \rtimes(\rtimes(\{u_1\}, dep), group), \exists m \in \rtimes(\{g\}, \mathrm{member}) : hobby(m) = golf\ )$$

For the expressive vague result *set* $\mathcal{Q}$ *we get:*

$$\hat{Q}_\lambda = \{g_1\},\ \hat{Q}_v = \{g_1, g_5, g_6\},\ UP_Q(g_1) = t$$
$$UP_Q(g_5) = \exists m : g_5 \xrightarrow{member} m \ \mathrm{A}\ loc(m) = \text{servers} \wedge hobby(m) = golf$$
$$UP_Q(g_6) = \exists m : g_6 \xrightarrow{member} \mathrm{m}\ \mathrm{A}\ loc(m) = \text{servers} \wedge hobby(m) = golf$$
$$\$_Q(j) = ((\exists i \in \{d_1, d_3\} : i \xrightarrow{group} \mathrm{j}\ \mathrm{A}\ loc(j) = \text{server 3}) \vee (\exists i : u_1 \xrightarrow{dep} \mathrm{i}\ \mathrm{A}\ loc(i) = server\ 3 \wedge i \xrightarrow{group} \mathrm{j}\ \mathrm{A}\ \mathrm{Vk} \in \{d_1, d_3\} : \mathrm{j} \notin (k \xrightarrow{group}))) \wedge \exists m : \mathrm{j} \xrightarrow{member} m\ \mathrm{A}\ hobby(m) = golf$$

3.2.4 *Binary Set Operations*

As one example for a binary set operation, we consider the expressive Vague Set Intersection $\mathcal{X} = \mathcal{V} \cap_3 W$. Expressive

Vague Set Union $\mathcal{X} = V \cup_3 W$ and expressive Vague Set Difference $\mathcal{X} = V \setminus_3 W$ can be treated analogously.

The computation of the enumerating component is very similar to the rules for vague sets, as stated in section 2.2. The only exception is, that we use the warrant of apprehension instead of the descriptive component, in order to check whether an element of the enumerating component of one input set also belongs to the other one.

$$\begin{aligned} X_\lambda &= (V_\lambda \cup \{i \in W_v \ [\$_V(i)] = t\}) \\ &\quad \cap (W_\lambda \cup \{i \in V_v \mid [\$_W(i)] = t\}) \\ \hat{X}_v &= (\hat{V}_v \cup \{i \in \hat{W}_v \ [\$_V(i)] \neq f\}) \\ &\quad \cap (W_\lambda \cup \{i \in V_v \mid [\$_W(i)] \neq f\}) \end{aligned}$$

Each missing element of $\mathcal{X}$ must be both missing in V and in W, otherwise it is either already in the enumerating component of $\mathcal{X}$, or it does not belong to $\mathcal{X}$:

$$\$x(i) = \$_V(i) \text{ A } \$_W(i)$$

The uncertain properties of an element of $\mathcal{X}$ is the logical conjunction of its uncertain properties for V and its uncertain properties for W. This is because the element must be in both input sets in order to qualify for the result. For elements that have not been in the enumerating component of one of the input sets, we compute the uncertain properties for this input set through the partial evaluation of the respective warrant of apprehension.

$$\mathrm{UP}_X(j) = \begin{cases} \mathrm{UP}_V(j) \text{ A } \mathrm{UP}_W(j), & \text{if } j \in \hat{V}_v \text{ and } j \in \hat{W}_v \\ \mathrm{UP}_V(j) \text{ A } [\$_W(j)]^p, & \text{if } j \in \hat{V}_v \text{ and } j \notin \hat{W}_v \\ [\$_V(j)]^p \text{ A } \mathrm{UP}_W(j), & \text{if } j \notin \hat{V}_v \text{ and } j \in \hat{W}_v \end{cases}$$

4. RELATED WORK

The articles [1, 9, 12, 13, 10] are concerned with vague sets resulting from vague attribute values - sometimes called *rough sets*. A rough set approximates an exact set by a lower bound (containing all elements surely contained in the desired set) and an upper bound (containing all elements potentially contained in the desired set). The most important difference between these notions of vague sets and our approach is that all papers mentioned above pre-suppose the accessibility of all relevant data. Consequently, these representations do not need a descriptive component. The work presented in [2] deals with partial inaccessibility as follows: A partial answer consists of an incremental *query*, which materializes the evaluable sub-queries. This incremental query can be issued as soon as the inaccessible data become available. In addition, the user can formulate so-called parachute queries which extract some information from the partial answer. Our own previous work [3, 5, 4] was focusing on good internal representations of vague collections, but neglected to also represent and compute the degree of and the reason for the vagueness induced by partial inaccessibility.

5. CONCLUSION

In this paper we have presented our notion of expressive vague sets, which is a significant enhancement of the vague sets we had proposed in previous papers. While vague sets are well suited to keep the vagueness of a query result evaluated on a partly inaccessible database as small as possible during query processing, expressive vague sets additionally compute the reasons for and the degree of the actually induced vagueness. This information explains to the end user why some result elements must be considered vague candidates, and it describes themissing elements.

After having implemented our concept of vague sets into an algebraic query language for the PCTE datamodel, we are now implementing it into XQuery, the WWW Consortium's proposal for an XML query language. XML based databases are a promising application domain for two reasons: XML is experiencing more and more acceptance as a common representation format, and queries on XML document bases distributed over the unrelyable Internet are most likely to benefit from our apporach.

Another interesting open issues for further research is the transformation of an expressive vague set to a natural language like representation.

6. REFERENCES

[1] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. In *Proc. IGMOD 1987, San* Franzisco, CA, 1987.

[2] P. Bonnet and A. Tomasic. Partial answers for unavailable data sources. In *Proc. FQAS'98, LNiCS* 1495. Springer, 1998.

[3] 0. Haase and A. Henrich. Error propagation in distributed databases. In *Proc. ACM CIKM'95,* 1995.

[4] 0. Haase and A. Henrich. A closed approach to vague collections in partly inaccessible distributed databases. In *Proc. ADBIS'99, LNiCS* 1691, Maribor, Slovenia, 1999. Springer.

[5] 0. Haase and A. Henrich. A hybrid representation of vague collections for distributed object management systems. *IEEE Transactions* on *Knowledge* and Data Engineering, 12(3):448–467, May/June 2000.

[6] 0. Haase, A. Schrader, K. Geihs, and R. Janz. Corba directories for virtual home environments. In *Proc. SoftCOM'99,* Split - Rijeka, Croatia, Trieste - Venice, Italy, 1999.

[7] 0. Haase, A. Schrader, K. Geihs, and R. Janz. Mobility support with corba directories. In *Proc. CNDS'00, San* Diego, USA, 2000.

[8] A. Heuer and A. Lubinski. Database access in mobile environments. In *Proc. ICDE'96, LNiCS* 1134, Ziirich, 1996. Springer.

[9] J. Morrissey. Imprecise information and uncertainty in information systems. *ACM Transactions* On Information *Systems,* 8(2):159–180, 1990.

[10] Z. Pawlak. Rough Sets. *International* Journal *of Computer* and *Information Sciences,* 11(5):341–356, 1982.

[11] Portable Common Tool Environment - Abstract Specification / C Bindings / Ada Bindings. IS0 IS 13719-1/-2/-3, 1994.

[12] E. Wong. A statistical approach to incomplete information in database systems. *ACM Transactions* on Database *Systems,* 7(3):470–488, 1982.

[13] L. Y. Yuan and D.-A. Chiang. A sound and complete query evaluation algorithm for relational databases with disjunctive information. In *Proc. PODS'89,* Philadelphia, Pa., USA, 1989.