Vertical Information Management: a Framework to Support High-Level Information Requests

Gregory A. Washburn
greg@cse.ogi.edu

Lois M.L. Delcambre
lmd@cse.ogi.edu

Mark A. Whiting
ma_whiting@pnl.gov

June 16, 1994

Abstract

Vertical information management (VIM) supports decision makers working within various levels of a management hierarchy, who seek information from potentially large, distributed, heterogeneous, and federated information sources. Decision makers are often overwhelmed by the volume of data which may be relevant and collectible, but overly detailed (e.g., from the breadth of open source data). Yet, the collected information must maintain its pedigree to allow access to detail on demand. VIM structures a top-down query refinement and bottom-up information collection process. We propose a VIM framework for specifying, refining, and partitioning a high-level information request which results in the extraction, collection, aggregation, and abstraction of the underlying data. A fundamental assumption of this work is that high-level information requests may involve data that is extracted or derived from underlying information sources, as well as data that is *not* present in the underlying information sources (referred to as "gaps"). We observe that current practice often involves manual processing and negotiation to select relevant information and to fill gaps. This framework includes specification at two levels: an abstract level, independent of the actual information sources used and a representational level to address the representational characteristics of the data within the information sources selected for the query. The top-down query refinement results in the abstract specification and the bottom-up information delivery is captured with the representational specification. This paper presents the VIM framework.

1 Introduction

One of the primary goals for Vertical Information Management is to provide decision makers residing at various levels of the management hierarchy with information that can help them make their decisions. The term was coined as part of the Vertical Integration of Science, Technology and Applications project at the Department of Energy's Pacific Northwest Laboratory. This project examined the information needs of the Department of Energy's Environmental Restoration program and noted the typical problems of diversity, magnitude and quality control (and so on) of the information. In this context, decision makers are faced with questions regarding: the allocation of funds, the assignment of priorities to areas in need of remediation, the evaluation of the success of work in progress, as well as many other high-level issues relevant to cleanup of the DOE complex. Decisions on how to allocate funds, for example, should be based on reliable cost and labor estimates for the various areas under consideration. Similarly, decisions about which sites should receive the highest priority may require information on the risk to human population and the intended land use for the site.

The detailed data supporting these decisions is often the result of costly (in terms of time and money) data collection efforts undertaken at different points in time. A variety of independent databases are used to store this detailed data. Clearly, new systems will continue to be developed and new data may continue to be collected for existing sites, but every effort will be made to benefit from existing data. Another problem, shared by many domains and organizations, is that collected data usually reflects the local, low-level purposes that drove the development of the original information sources. However, decision makers frequently ask questions about a wide range of issues, and such questions may be answered based, in part, on the low-level detailed data that has been collected. Finding the relevant data can become a time-consuming and frequently manual process.

1.1 The problem

The challenges associated with this mission for the Department of Energy as well as for other application areas are highlighted here:

- A decision maker needs high-level information, at a level of abstraction above the stored, detailed data. This person is not necessarily knowledgeable of the available systems, nor of the data they contain.
- The systems that contribute data are independent, and contain data of varying detail, data that was collected for specific purposes, and data that is of varying levels of quality.
- A decision maker may express needs for high-level information that are independent of any particular information source. The requested information is not necessarily stored directly in any of these systems. In this sense, these requests are like queries floating in free space, queries without a schema. There is a negotiation process based on the difference between the query and the available data.
- The information presented to the decision maker needs to be understandable and defensible; that is, the information provider needs to be able to defend or explain how the delivered information was obtained. In this sense, the negotiation is part of the result and must be made explicit.

This is in sharp contrast to the characteristics of traditional database applications. Traditionally, it is assumed that the data of interest to the user can be defined explicitly and completely in the schema of one or more databases. This constitutes more of a "data driven" approach; the available data dictates which questions may be asked. The traditional database answers queries according to the *Closed World Assumption*, whereby the absence of data is interpreted as an answer of "no" or "false." Also, these questions are expressed as queries in a query language against particular schemas.

The current solution to the problems introduced by requests for high-level information relies heavily on expert knowledge based on the isolation of the decision maker from the information collection process. The information expert must conduct a mostly manual effort to:

- identify the relevant underlying systems,
- identify relevant data, and possibly negotiate a modification to the original request,
- map the request onto local queries over these systems,
- identify appropriate routines or algorithms necessary for the summarization process,
- identify requested data that is missing and devise ways to estimate it,

- decide how the results of the local queries and routines used can be combined to provide the requested high level information, and
- report the requested information.

This manual process may take weeks or even months to complete. Without an appropriate mechanism to record the request specification and information gathering process, the large investment of manual effort is lost and is re-invested for future requests. It is often not even possible to re-execute the information delivery plan, e.g., to use more recent data. It is even less likely that an information delivery plan could be re-executed for an even slightly different set of information sources, or slightly different criteria. Also, data delivered through a manual process may not have enough detail about the pedigree or derivation path to allow the results to be defended.

1.2 The solution

In order to avoid losing the heavy investment of time and effort associated with satisfying requests for high-level information, a straightforward solution is proposed: to declaratively specify i) the high-level request, ii) the data gathered from individual systems, and iii) the derivation of the summary information using the gathered data. This research does not support updates as part of an information request; that is, we do not intend to support updates or deletions on the local information systems based on input from the decision maker. But the systems may change locally, and autonomously, according to their intended purpose. Considering a retrieval-only framework also opens up opportunities for data extraction techniques because the mappings from the high-level view to the underlying information sources can be more flexible.

Our solution provides:

- specification of a high-level information request, along with the way the supporting data is collected, and the way the information is to be delivered (to provide a detailed pedigree),
- use of current and future efforts in standardization of: terminology, data representation, and metadata,
- support for re-use of previous specifications to satisfy new high-level information requests,
- mechanisms to systematically handle gaps and incompatibilities in data and,
- mechanisms to make hidden assumptions and context surrounding the high-level information understandable and useful to decision makers e.g., about how missing data was estimated.

Our VIM framework provides a syntax for the initial request and two different, but complementary specifications: request refinement, and information delivery. The syntax is introduced in the next section and the specifications are presented in Section 3. In Section 4, the VIM framework is defined which consists of components to support both the request refinement and the information delivery specification. Languages have been defined for the specification of each component; the associated grammars can be found in Appendix A. VIM constitutes a large body of work, and this paper is intended to introduce this problem and provide an overview of our solution.

```
PROVIDE  <attribute list>

REGARDING  <scope qualification>

WHERE <condition list>
```

Figure 1: Syntax for a high-level information request.

2 Syntax for information requests

A decision maker is not necessarily aware of the underlying information sources, and more importantly, the information needed may not be stored directly in a data source. The request may be initially expressed through conversation, as a written memo, or as a voice mail message; all of these can be imprecise and may result in an ambiguous specification. This is the very nature of high-level information requests. We expect an information expert, working with the decision maker, to express the request using the syntax shown in Figure 1. Its purpose is simply to serve as a starting point for request refinement specification. A request expressed using the high-level information request syntax is broken into three primary clauses: *PROVIDE, REGARDING*, and *WHERE.*

The *attribute list* identifies the data items of primary interest. This is like the attribute list in an SQL select-from-where clause, but it is less precise because there is no known global schema to address. This attribute list may correspond to column headings on a report, or to axes in a graph. These attributes guide the request refinement process; they must ultimately be derived or delivered from available information sources or estimated by an information expert and delivered to the decision maker.

The *scope qualification* specifies the breadth of the request. As an example, a geographical region of interest is often associated with high-level information requests. The delivered information is meaningless without an understanding of the region or area it is characterizing. The scope qualification will influence the choice of information sources and will be used to evaluate the coverage of the information provided.

The *condition list* may specify a wide variety of conditions on the attributes to be delivered. These conditions may be expressed against the actual values of the attributes in the attribute list or they may be conditions against information surrounding the attribute values, i.e., against metadata, such as quality levels above a certain threshold. Note that there are no hard and fast rules for deciding whether a given condition belongs in the *REGARDING* or the *WHERE* clause because it depends on whether or not coverage information is explicitly stored in the information sources.

The syntax allows basic components of a request to be identified and made explicit. It is expected that an information expert working with the decision maker will be able to partition the request to fit the proposed syntax. For the purpose of this work, it is assumed that requests will be delivered in this syntax.

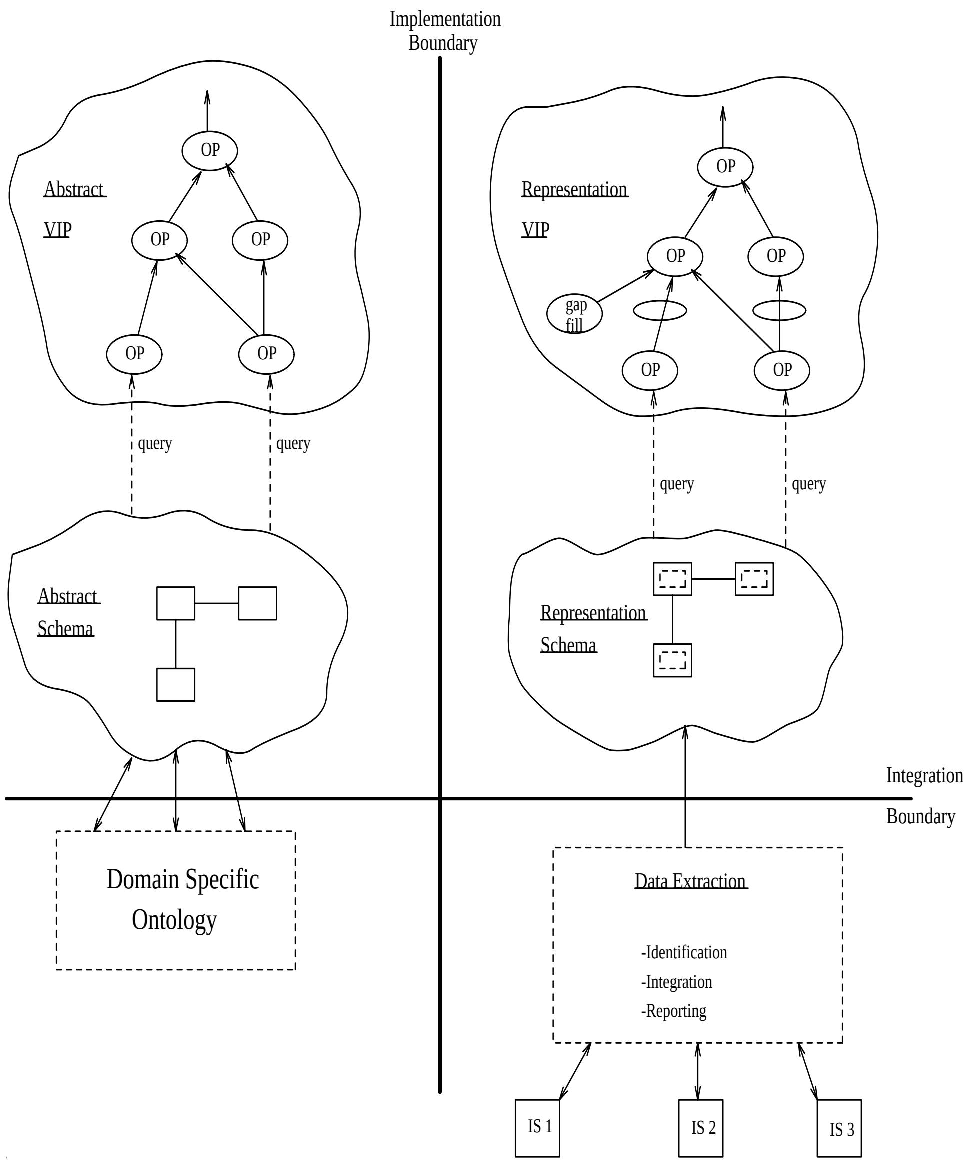

Figure 2: The VIM Framework

3 Abstract and representational specification

In this section, the specifications supported by the VIM framework are presented. A more detailed description of VIM is presented in the next section. The two levels of specification correspond to the left and right sides of Figure 2. The request refinement specification involves refining, restating, and partitioning the initial request using top-down decomposition. The purpose of the decomposition is to transform the attributes of interest in the initial request into successively simpler and more concrete components. This may be the result of recognizing that an attribute in the attribute list is really the aggregation (e.g., average, min, max) of several simpler components, or the result of an algorithm applied to several components, or simply a compound object constructed from its components (e.g., set(component 1, component2)). The decomposition terminates when the identified components correspond to information typically found in database systems particular to the application area. An information expert familiar with the application domain would be responsible for directing the decomposition. This decomposition is carried out independent of the representational details (e.g., type, format, etc.) of the data contained in available information sources and the coverage of the information.

Request refinement has identified a view of data, in the *abstract schema*, (referred to as “VIP data”) that can be expected to contribute to forming the high-level information. Secondly, it has identified the primary conceptual steps for constructing the high-level information from the VIP data, in the *abstract vertical information plan* (AVIP). These elements of the specification are shown on the left side of Figure 2.

Once the initial request has been refined, the construction of the corresponding high-level information object can be specified. This is shown as the information delivery specification on the right side of Figure 2. The information delivery specification adds detail to the request refinement specification by:

- identifying the particular information sources that contribute the VIP data,
- specifying the representation for the VIP data identified in request refinement,
- identifying additional processing steps based on the representational characteristics (e.g., to convert from one representation to another), the availability of the expected VIP data, and additional conditions expressed in the condition list of the initial high-level request, and
- including metadata and annotation with the results of each of these steps in the derivation process.

4 The VIM framework

In this section, we discuss Figure 2 in more detail.

4.1 Representation-independent components

Primary processing steps for constructing the high-level information from the VIP data are identified during the decomposition of the initial high-level information request. These steps are represented in the *abstract vertical information plan* (AVIP). The AVIP is represented as a directed acyclic graph with the initial high-level request at the root, and the processing steps at the internal nodes. The leaf nodes correspond to the requests for VIP data. For the purposes of request refinement specification, the AVIP is constructed in a top-down fashion (thus, it is constructed in the reverse direction of the arrows shown for the AVIP in Figure 2). But it represents processing steps to be executed in a bottom-up fashion where VIP data is extracted and then aggregated, filtered, processed, etc. to produce the high-level information object in response to the original request. Each node is an operator that receives data from below and delivers data upward.

The *abstract schema* captures the view of VIP data identified as the result of the request refinement process. It is a conceptual view of the necessary data, independent of particular information systems. Data items are identified by names that have associated definitions in a corresponding *domain-specific ontology*. The abstract schema is at the level of an entity-relationship diagram, where entities are defined by attributes and relationships.

The most significant feature of the abstract schema is that it is expressed without representations or types for attributes. As an example,

```
interface Sample{
      attribute sample-name:  name;
      attribute sampling-date:  date;
      attribute sample-location:  location;
          <Format, Precision>
      ...
}
```

defines the *location* attribute of the *Sample* entity as a *sample-location* but the normal data type specification, e.g., char[20], is not present. The semantics of this attribute is indicated by the definition of the domain *sample-location* in the application ontology. This is directly analogous to the specification of the domain name for an attribute in the relational model. We view this as a key feature of the this research that directly enables the reuse of a request refinement specification, e.g., over different underlying information sources. The grammar for the Abstract Object Definition Language is given in Appendix A; the syntax is derived from the Object Data Management Group's Object Definition Language [4] with the type specification removed.

4.2 Representation-dependent components

The representation-dependent components are specified when the information sources to be used to answer a particular request are identified. The manner in which information sources are identified is outside of the scope of this research. It is based, in part, on the *REGARDING* and perhaps the *WHERE* clause of the initial request. The possibility for automated assistance is dependent on the presence of relevant metadata about the underlying information sources, e.g., scope.

The specification of information delivery involves two primary tasks: representing the data extracted from underlying information systems, and the construction of the high-level information object. The components responsible for these tasks are shown on the right side of Figure 2.

The abstract schema provides the structure for the *representation-based* schema. Entities, attributes, and relationships all have the names they were given in the abstract schema. The representation-based schema is a specification for the actual data that is to be acquired from the underlying information sources. This specification provides the additional representational characteristics (e.g., type) for the data, as well as metadata describing various aspects of the VIP data (as identified in the abstract schema) such as quality, coverage, units and other contextual aspects.

The data specified by the representation-based schema is incorporated into the construction process via queries specified within the *representation-based vertical information plan* (RVIP). These leaf-level queries in the RVIP correspond to the leaf-level queries in the AVIP. The RVIP is a directed acyclic graph, with the final step for the construction of the high-level information object at the root. The internal nodes are operations corresponding to the internal nodes of the AVIP along with additional operations (nodes) that accommodate the representational detail and the discrepancy between the expected VIP data and the acquired VIP data. These additional operators generally perform representational conversions as well as estimation and other techniques to fill gaps in the available data.

4.3 Context

There is a need for metadata, also referred to as *context*, to be included with specifications developed as part of the VIM framework. The metadata is used to make assumptions and other aspects surrounding the primary data explicit and open to inspection. The metadata might specify the units a value is reported in, or the instrument a measurement was taken with, for example. The format for expressing this metadata is based on a representation developed by Sciore, et al., where the context for data is defined as an attribute list associated with the data item [15]. The field names represent the metadata to be captured. These fields may be assigned single values, or they may contain a textual description. Each field itself, may have context associated with it. This results in a potentially nested list of fields assigned to values to explain assumptions surrounding the primary data item. An example of context that may be associated with a volume measurement is:
<**Units** = ``gallons'', **Temp** = ``25.5''<**Units** = ``Celsius''>>

This denotes that the particular volume measurement was reported in gallons and the temperature was 25.5 degrees Celsius.

It is not the intent of this work to identify all possible types of context and define the appropriate fields. Instead, we use the above representation for context with the expectation that particular applications of VIM, for particular application areas, can define and use context as needed. Metadata can be introduced at every step in the VIM process, using the context syntax.

4.4 Data extraction

The VIP data identified in the abstract schema must be extracted from actual underlying information systems to populate the representation-based schema. This is illustrated by the component labeled *Data Extraction* in Figure 2. The details of identifying the relevant underlying data, extracting it, converting between its various representations, and forming the classes corresponding to entities in the abstract schema are out of the scope of this work partly because of the extensive work in this area, generally referred to as schema integration or schema mapping (see the section on related work) e.g., for federated databases. We limit our discussion to a definition of the responsibilities of this task. In some sense, data extraction is treated like a black box; the inputs and outputs are specified, and the the primary issues that must be resolved inside the box are listed here.

4.4.1 Inputs to data extraction

The inputs for the data extraction task include:

- Representation-based schema: including a representation chosen for all VIP data as well as the optional context for entities and attributes. In this sense, a complete representation-based schema is used as input to the data extraction task; the goal is to populate this schema with underlying data and metadata.
- Metadata for the underlying databases: this is in the form of schema definitions and additional local context associated with each information source. The local context allows a data source to post information about itself; making implicit properties of the source explicit and open to inspection. This may include statements regarding the scope and/or coverage of data contained in a system, for example.
- REGARDING clause: The region or characterization expressed in the high-level information request must be accounted for when choosing particular information sources to use for VIP data. Language for concrete specification of scope at the request level as well as at the information source level is not part of the contribution for this work. This is an informal specification at best.
- WHERE clause: The conditions expressed in the WHERE clause of the information request can be transformed into conditions on the VIP data. The data extracted from the underlying sources must meet these translated conditions. Ultimately these clauses may appear in (local) queries used for data extraction, perhaps modified as necessary for the particular information source.

4.4.2 Responsibilities of data extraction

Data extraction is responsible for connecting the representational specification to actual underlying data. This responsibility involves several tasks, illustrated in Figure 3. These tasks may require significant manual effort, or they may benefit from techniques developed as part of the research in the area of federated heterogeneous databases. Regardless of the degree of automation, the following must be accounted for:

- Identification of relevant information sources: Decisions as to which information sources will contribute relevant information must be made. These decisions may be based on the REGARDING clause of the initial request, the representation-based schema specification, the definitions of underlying schemas, or the local context for information sources.
- Mapping: Entities and their associated attributes from the representation-based schema must be identified in the schemas of the underlying information sources. The participant information sources must establish semantic equivalences betweeen their entities, attributes, and relationships and the standard terms defined in the domain-specific ontology. This may be done as needed, for a specific VIM request, or ahead of time (perhaps for an earlier VIM request).
- Extraction: The data from underlying sources must be lifted using the defined mapping. In this process, entities with missing attributes will be discovered and reported to the representation schema using the

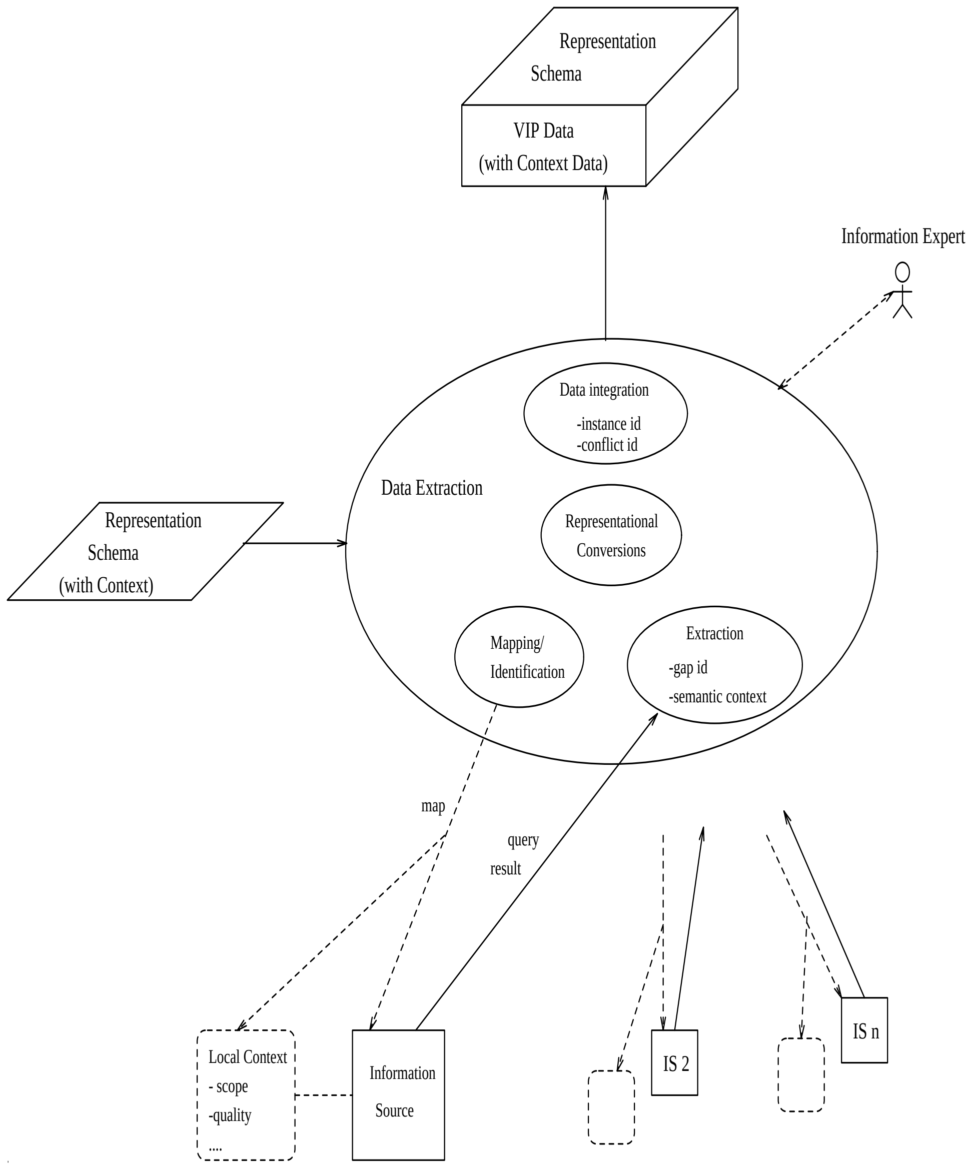

Figure 3: Responsibilities of Data Extraction.

appropriate fields of the context associated with the attribute. Data will also be lifted that corresponds to the other fields in the context specified for the attribute. This may involve data from the local context for the individual information source, or data that is stored directly in the source.

- Conversion: The lifted data must be converted to the representation chosen for the representation-based schema, if necessary. Data in the underlying systems may be of different types, in a variety of units, based on different scales, or any of a number of different representations.
- Integration: The converted, extracted data, must now be integrated. This is more complicated than performing a simple union of all data sets corresponding to the mapping and extraction for a given entity. Integration must be performed at the data level to identify corresponding instances (the instance identification problem [17]), and to identify conflicts among their attribute values. The way these conflicts get resolved is mandated by their use in the corresponding RVIP.

A subset of the responsibilities identified here for data extraction correspond to classical schema integration for answering queries in a federated environment. Classical schema integration attempts to combine all data from the participant databases to form a complete, conceptual global schema. Queries issued against the global schema are then translated into a set of local queries according to the mapping identified during schema integration [2, 7].

4.4.3 Outputs from data extraction

Data extraction results in the production of the following:

- Populated representation-based schema: Each entity, attribute, and relationship must be extracted using one or more associated queries that can be issued to deliver data corresponding to the representation-based schema specification. The production of these queries is part of ongoing work [14].
- Context: Metadata is associated with each attribute in the representation-based schema that reflects whether the corresponding data was stored directly in an underlying information source or not. It is possible for data extraction to report an estimate, or to make a substitution in the absence of underlying data that meets the exact specification in the representation-based schema. It is left to the RVIP to deal with this alternate information; this is accomplished by the insertion of the appropriate operators into the RVIP.
- Identification of conflicting instances: There may be overlapping coverage between underlying data sources, and an instance of an entity stored in one source may appear as another entity in a different source. This is not a problem if both sources agree on the attribute values of interest for that instance, then the two entities can be consolidated into one representation. This is not always the case. When the values disagree, a *conflict* has been detected and must be resolved. Data extraction is only responsible for reporting these conflicts, how they are resolved is handled in the particular RVIP that uses the representation-based schema.

5 An example

An example that demonstrates each of the pieces of VIM is presented here. The example is based on an actual, ongoing high-level information request satisfaction effort, but the names of locations, and systems involved, as well as data values reported, are fictitious. This example is based on a high-level information request related to environmental remediation. A more detailed discussion of this example can be found in [18].

5.1 The scenario

The Region of interest is one of several that have been contaminated with nuclear waste. Of particular interest is a major River that passes through the Region, near the contaminated areas and provides both drinking water to downstream communities and irrigation water to surrounding farmlands. Toxic waste has been buried in tanks underground, stored in tanks above ground, as well as in various other containers. Over time, some of these containers may begin to leak and contaminate the surrounding areas. There is concern about the risk this contamination presents to humans. In particular, there is concern about contaminants that may reach the river.

A request has been issued to determine the threat to humans from contamination in and near the River. The metric to be used in analyzing the threat to human health is public risk incurred through the various receptor pathways (e.g., groundwater, surface water, air). Risk is a technically complex determination and a politically sensitive issue; a full discussion of this metric is beyond the scope of this paper. For the discussion below, the calculated risk numbers represent incidences of cancer for an on-site receptor. The organization managing the Region has been charged with satisfying this request because it has the most direct and reliable information about the Region. The scientists working within this organization are extremely knowledgeable of details associated with the particular sites. They understand many of the assumptions surrounding the data collection and analysis efforts.

These results will be accessible to the public and will be inspected by various environmental special interest groups. The risk numbers will be challenged by these groups looking to increase the political pressure on the managing interests of the Region. Thus, we need results that can be defended along several fronts: political, human safety, and clarity of results.

The example is presented in a narrative fashion in order to highlight the steps involved in VIM. The corresponding real-world request has been performed manually, except for the actual queries on underlying data sources.

5.2 The syntax expression for the request

The initial high-level information request has been elaborated with a little more detail after discussion among the people in charge of satisfying the request. The attributes of interest are the human risk numbers and their associated uncertainty percentage. The sites of interest are the ground water, surface water, and the soil sites within 500 feet of the River between river mile X and river mile Y. This is the section of the river running through the Region. The decision maker wants to make sure that the reported results are on the conservative

PROVIDE:	human risk numbers, %uncertainty
REGARDING:	surface water, groundwater and soil sites in the Region within a 500ft band of the River, between river mile X and river mile Y
WHERE:	human risk threshold is low

Figure 4: An example high-level information request in a standard syntax.

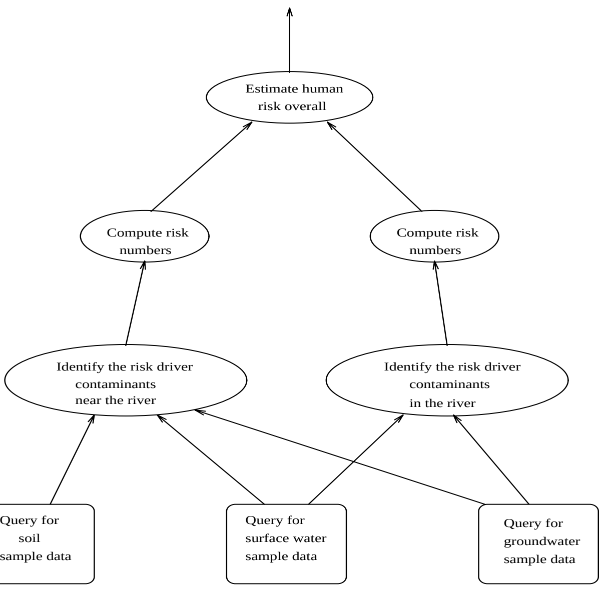
Estimate human risk overall
Compute risk numbers
Compute risk numbers
Identify the risk driver contaminants near the river
Identify the risk driver contaminants in the river
Query for soil sample data
Query for surface water sample data
Query for groundwater sample data

Figure 5: AVIP for assessing the risk a polluted river presents to humans.

side; that is, no potential contaminant of concern goes unreported. This is captured in the *WHERE* clause of the request shown using the syntax for high-level information requests in Figure 4.

5.3 The corresponding AVIP

The overall human risk can be broken into the risk associated with contamination found directly in the river and the risk associated with the contamination of the land near the river. (See Figure 5.) At this point, the decision maker requests that the data collection activities be centered on contaminants of concern that are the risk drivers. That is, information will be provided on those contaminants that contribute an individual risk level higher than a certain threshold. This assures only the most relevant contaminants are reported and speeds up the entire data analysis process. An AVIP specification language has been defined and the grammar is presented in Appendix B. A complete specification of the AVIP can be represented as an expression in this language.

The necessary base data has now been identified; queries for data on soil, surface water, and ground water sites can be represented. Not shown in the figure is the detail of these queries. Notice that the soil sample

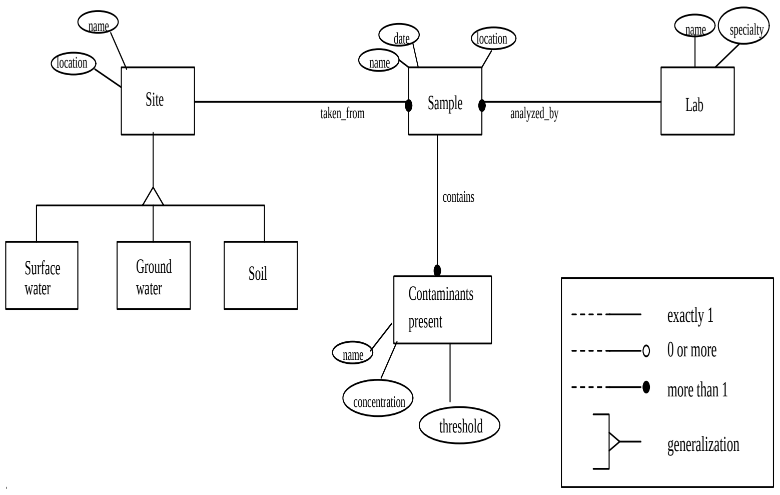

Figure 6: Identification of expected VIP data using an abstract schema.

data only contributes to the list of contaminants near the river, and not to the list of contaminants found in the river. Conceptually, they require concentration levels associated with particular contaminants found in samples taken at particular locations from the sites of the above-mentioned types. To actually specify these queries, the abstract schema must be developed. These queries direct the formation of the abstract schema.

5.4 The corresponding abstract schema

This abstract schema identifies the VIP data expected to be present in underlying systems to be used to satisfy the high-level request. The basic view of this data is shown in Figure 6. In general, a sample is taken from a site and is analyzed by a laboratory. The analysis results in the detection of contaminants with associated concentration levels for that sample. A site may be a surface water site, a ground water site, or a soil site and is associated with many samples taken from it. An attribute of particular interest to the AVIP is *location*. The desire for conservative risk numbers in the initial high-level request is reflected in the inclusion of the *threshold* attribute associated with the detected contaminant. An abstract schema can be expressed using the grammar for the language presented in Appendix A.

5.5 The representation-based schema

The corresponding representation-based schema has the structure of the abstract schema shown in Figure 6, but with types and optional context defined for each of the attributes. For example, the type of the *sample.location* attribute is designated to be a pair of floating point numbers from the domain of "sample_location," defined in the domain specific ontology. It's associated context is the following:

<**status**, **format**= "lat-lon", **precision**>.

The *status* field will be assigned a value during data extraction when it can be assessed whether or not the data item corresponding to *sample.location* has been lifted directly from a source, or has been estimated. The *format* field dictates that all retrieved location data will be in latitude-longitude coordinates. The *precision* field will be given a value during data extraction. In other words, all sample data will have location in

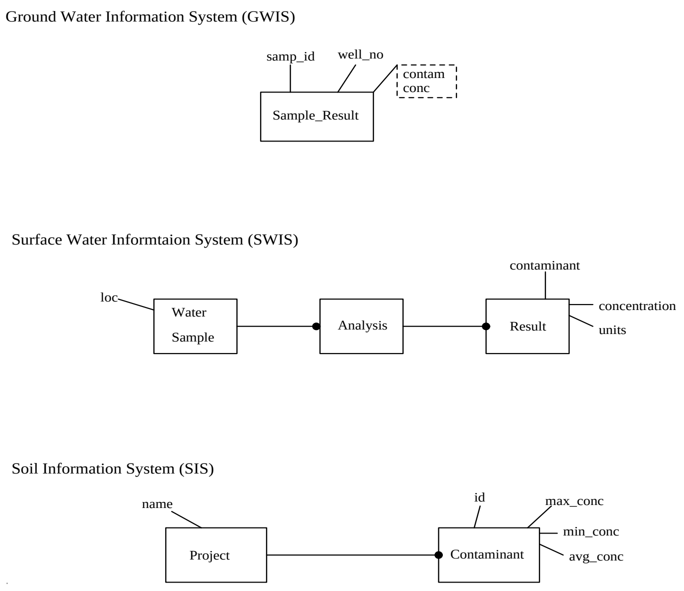

Figure 7: Schemas for particular underlying information sources.

latitude-longitude, but with varying degrees of quality and precision. Appendix C presents the grammar for our representation-based schema definition language.

5.6 Overview of the data extraction

Now that the representation schema has been specified, the next task is to use data extraction to populate it. For this example, consider three information systems, one that tracks ground water data (GWIS), one for surface water (SWIS), and one for soil (SIS).

For the purpose of this example, we will only discuss the attempts to populate the *Sample* entity present in the representation-based schema (based on the abstract schema shown in Figure 6.) Note: names appearing in the representation-based schema are in *italics*, and the names from underlying sources are in quotes. In particular, we focus on the attribute *sample.location*. Each of the underlying systems has different representations for the information specified in the representation-based schema.

- GWIS: The GWIS schema represents *Sample* as part of the relation "Sample_Result," see Figure 7. The attribute *location* appears as "well-no", identifying which well the sample was taken from. According to the semantic context specified in the representation schema, location needs to be reported as lat-lon coordinates. Since there is no direct mapping between a well number and latitude-longitude coordinates, the well number is used in place of the coordinates. This is recorded in the context associated with *location*.
- SWIS: *Sample* corresponds to "Water Sample" as shown in Figure 7. *Sample.location* matches with "loc" represented using latitude and longitude coordinates. There is no recorded *precision*, but it is estimated that the locations are accurate to within 100 feet and this is indicated in the *precision* context field.
- SIS: The system tracking soil data doesn't include the notion of samples explicitly. It simply identifies

the maximum, minimum, and average concentrations of contaminants found as the result of particular soil characterization projects. For this system, *Site* and its related *Sample* maps onto "Project" in the sense that the region the project covers is a site, and there is no distinction between individual samples. The location associated with a sample is represented as the location of the project. Location isn't stored explicitly with the project description, so the gap context for *location* indicates that a gap occurred and it was filled with "Project.name." It is up to the information expert to determine a general location based on the project name. Note that this provides more information than simply reporting "N/A" or a null value.

The data integration task involves identifying redundancy within the extracted data. This problem is simplified for this example since each contributing system covers a disjoint set of sample types and hence disjoint samples. There is potential for overlap however within SIS. If two projects have intersecting sampling regions, it is possible that the maximum concentration for a contaminant was found within the intersection of the regions for both projects. Complete data integration would involve identifying this overlap, recognizing conflicting values, and reporting this with the delivered data.

5.7 The corresponding RVIP

Now that representations for the VIP data have been set, underlying information sources have been identified, mappings have been defined, and gaps have been identified and/or partially filled, the construction of the high-level information object can be specified. The corresponding RVIP for the original AVIP is shown in Figure 8.

Accommodations must be made for the lack of precise location data in order to support the computations for risk based on contaminants in the River and near the River. Additional factors must be taken into consideration as well, such as the structure and type of the data flowing on the arcs of the RVIP, the gaps in the semantic context (namely in missing precision and %error data), and operators for the propagation of uncertainty. These were not included in this example for the sake of simplicity.

Operators have been introduced that were not present in the AVIP (Figure 5) to accommodate for these discrepancies, as indicated by ellipses drawn with a dotted line. In particular, the lack of concrete location data resulted in the introduction of operators to "make do" with the information that was available. Constructor operators were also introduced to form the complex object containing all concentrations for contaminants found in samples of the various kinds. This RVIP specifies the steps taken to derive the human risk numbers requested by the decision maker. Appendix D presents the grammar for the RVIP specification language which can be used to express an RVIP in greater detail.

In summary, the AVIP was developed independently of the underlying information sources and resulted in the identification of expected VIP data. This was captured through specification of the abstract schema. The connection to underlying data was made by specification of the representation-based schema and the subsequent data extraction effort to populate it. Finally, construction of the requested high-level information was spelled out using the RVIP. Additional operators were introduced to handle the discrepancies identified during data extraction and to accommodate for the REGARDING and WHERE clause from the initial request.

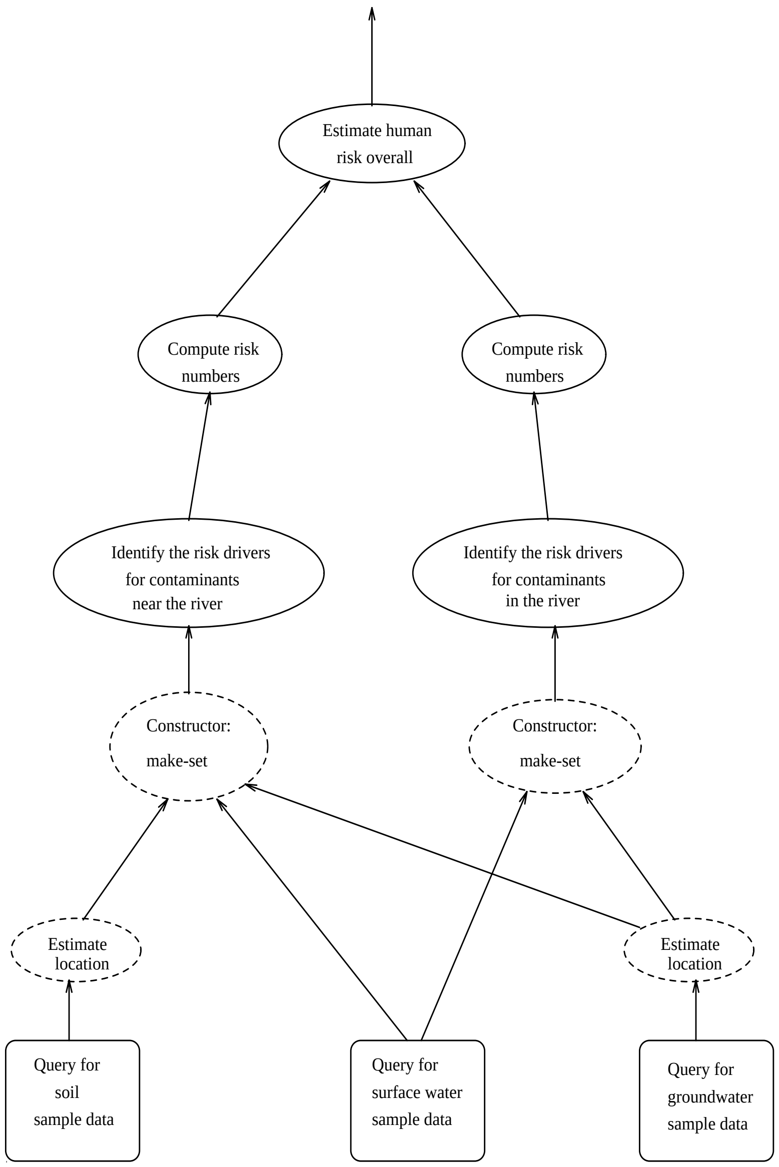
Estimate human risk overall
Compute risk numbers
Compute risk numbers
Identify the risk drivers for contaminants near the river
Identify the risk drivers for contaminants in the river
Constructor: make-set
Constructor: make-set
Estimate location
Estimate location
Query for soil sample data
Query for surface water sample data
Query for groundwater sample data

Figure 8: An RVIP for assessing human risk.

6 Related work

6.1 Federated databases

It can be argued that a federated database system is another approach for providing decision makers with information using data from multiple underlying sources. A federated database system, as discussed in [16], is a collection of cooperating, independent databases that may or may not be based on different data models.

A query can be issued against the global schema and query processing maps this query onto individual queries on the participating systems and the result is returned. The maintenance of the global schema in the face of changes to the underlying systems, or the removal or addition of systems to the federation, can be a formidable task. Generally speaking, federated systems do not deal with missing data from underlying systems. Also, the idea of using incomplete data as part of the query result is not well established in federated database research.

A variation on the construction of a complete global schema is to allow views to be defined by the users for particular applications and then construct a "partial global schema" to support the defined views [8, 3, 6]. The approach taken by Motro relies on the creation of a superview and its mapping onto the underlying databases as a *virtual database* [10]. In some sense, the representation-based schema for VIM is a superview of the particular underlying databases. However, the representation-based schema is not defined in a bottom up fashion from the schemas of the particular underlying databases. Rather, its structure is defined by the corresponding abstract schema which may be defined before particular underlying systems have been chosen and may result in gaps.

Schema integration claims to support ad hoc queries over all data contained in (or declared visible through export schemas for) the individual local databases [2, 7, 9, 13]. Our approach on the other hand, is concerned with specification and delivery of information for particular information requests. In this sense, we are concerned only with integrating "enough" of the underlying databases as is required to satisfy the request (as specified in the representation-based schema), not for a global schema. Integration efforts based on the global schema approach often intend to support updates to the underlying databases as well [9]. This is not our intent; VIM data extraction is concerned with read-only access to the underlying information sources.

One primary purpose for the introduction of VIM is to specify the derivation of information that is not directly stored in any underlying system. Decision makers need information that is an abstraction, or summarization of data contained in underlying systems. This requires the ability to use operators for estimation, simulation, etc. that transform specific data into information useful to a decision maker. Equally important, is the ability to understand what this high-level information means, how it was derived, and what it was derived from. The specifications developed as part of VIM provide this information. This distinguishes VIM from traditional work on federated databases.

Another important distinction is the attention given to "gaps" in the underlying information sources. In many cases, the global schema approach only integrates the data that perfectly matches the specification of the global schema. This is not the case with the population effort for the representation-based schema. It is expected that mismatches and inconsistencies will occur in the underlying information sources, and rather than ignoring the mismatched data, the mismatches are identified and taken into account so that the data can be used in the RVIP. VIM is also concerned with supporting requests for information made independent of the particular information sources.

As a result of the high degree of semantic disparity and separation of the information request from the underlying data, there is a large amount of effort, usually manual, involved in the process of finding the correct underlying data and manipulating it to produce the requested high-level information. This makes the incorporation of the appropriate metadata for annotation of these steps a crucial element in our approach. For all of these reasons, a federated system cannot replace the VIM framework.

6.2 The ARPA Intelligent Integration of Information Program

The Advanced Research Projects Agency has funded a large scale effort, involving computer scientists from the University of Colorado, Stanford, George Mason, and others, as well as participants from industry to develop an architecture for the intelligent integration of information.

> The goal of the ARPA program in the Intelligent Integration of Information (I3) is the development of the technology needed for a broad community of information producers and consumers to access, manipulate, and integrate information residing in multiple heterogeneous distributed information sources. [1]

A reference architecture has been proposed and is still under development. It is currently organized as services provided at three layers [1]:

- The Wrapping and Translation Layer: This is the bottom layer of the architecture and services act as a wrapper around particular information sources. That is, the resource is "wrapped" with a representation that is consistent with other components of the architecture.
- The Mediation Layer: This middle layer provides services for a customer. These services add value to the underlying data, they are usually designed manually, and acquire data via wrapper invocations. Mediators are statically defined over a collection of resource wrappers.
- The Facilitation Layer: These services may establish links to resources dynamically, and they may call on mediation services to perform value added tasks. These services are aimed at providing flexible links to underlying resources which include mediators and wrapped resources.

The goals for the I3 project are on a larger scale than those for our work. Many of the tasks we recognize as manual steps are the goals for automation in the ARPA project. VIM is dedicated to knowledge capture for the largely manual negotiation process that takes place in response to high-level information requests. In this sense, VIM works to enable the development of intelligent agents for mediators to automate aspects of this manual process.

VIM can be viewed as a form of mediator in the I3 architecture. The mediator is responsible for representing the results of queries in a canonical format; this requires that a mediator resolve the semantic heterogeneity of data items extracted from the underlying wrapped resources. There is an entire body of work on defining mediators [19] e.g., using MSL [11], potential canonical data representations [12], their role in the ARPA I3 program [1], and their use for integrating heterogeneous information in general [5, 11].

There are at least two key differences between the mediator based approach and the framework for VIM. Mediators are specified particular to a set of information sources; there is no notion of an abstract specification provided. VIM is dedicated to reporting gaps in underlying data, to making pedigree an integral part of the the final information product, and to documenting the overall negotiation process that took place.

7 Conclusion

An important result of the research has been the characterization of details behind the need for high-level information based on detailed data stored in independent information systems. The following characteristics of the problem have been identified:

- There is a strong need to capture and record the manual effort inherently involved with specifying and satisfying requests for high-level information,
- There are several subproblems: i) expressing the initial request in a useful manner, ii) finding the relevant underlying data, and iii) combining and manipulating it in meaningful ways,
- The requested data is not necessarily stored in any particular information source,
- The available data may not be perfect; there may be missing data, and
- There are many barriers to automation resulting from the overall complexity of the problem.

Based on the recognition of the problem, and its subproblems, components of a framework for its solution were identified and defined. The framework for vertical information management has the following components:

- An *abstract vertical information plan* for identification of the primary, representation-independent steps involved in the high-level information object construction,
- An *abstract schema* for representation-independent specification of data to be used in the high-level information object construction,
- A *domain-specific ontology* used as the semantic base for terms appearing in the abstract schema; entity names and domains of attributes are defined with respect to a particular application domain,
- A *representation-based schema* for specification of data extracted from particular information sources based on the specification provided by the abstract schema,
- *Data extraction* for mapping the representation-based schema onto the schemas of underlying information sources, extracting data, identifying gaps, and integrating the instances to populate the representation-based schema,
- A *representation-based vertical information plan* for specification of the high-level information object construction based on the representational characteristics of data specified in the representation-based schema,

The above components provide the following capabilities:

- Specification of request refinement using the abstract vertical information plan, and the abstract schema,
- Specification of information delivery using the representation-based schema and the representation-based vertical information plan,
- Support for contextual metadata within each component of the framework; each step in the derivation of high-level information can be annotated, and context provided for the data involved.
- Successful retrieval despite missing or incomplete data in the underlying information sources, and
- Support for intervention by the information expert within the information derivation process to supply estimates or reconcile discrepancies in the underlying data.

The abstract schema is a reusable component and it is a primary contribution of this work. It is a semantic specification of the data used to produce the high-level information, independent of the representation of the data. The independence from representation, its use as one of the initial components of the specification process, and the fact that the schema is limited to one particular application area combine to make the abstract schema reusable. The abstract schema is a stand alone specification, based on the domain-specific ontology, that can be mapped onto a variety of information sources within a particular application domain. The complexity of the mapping required may vary, but the abstract schema itself, is fixed. It is even possible that several requests for high-level information can be supported by a single abstract schema specification.

The abstract vertical information plan can also be reused for the same request issued against a different set of underlying information systems. The representational details of the data are not visible in this specification; it is the refinement of the initial request. So if the request needs to be reissued, the abstract vertical information plan may also be reused.

The development of a prototype for VIM is underway in conjunction with the Department of Energy's Pacific Northwest Laboratories to test the extent to which the abstract specifications are reusable. Of particular interest is the versatility of the abstract schema over varying collections of underlying information sources.

The representation-based schema provides an interface between the problems associated with integration of the underlying data, and the construction of the high-level information object. The results of extracting underlying data, converting it to a chosen representation, and integrating it are captured in the representation-based schema. The construction of the high-level information object benefits from this effort, but is free from the details associated with populating the representation-based schema. The relationship between the construction of the high-level information object, and the underlying information systems contributing data, is localized in the representation-based schema. This greatly simplifies the overall process of satisfying high-level information requests. In particular, this separates the detail of the data extraction process from the manipulations required to answer the high-level information request.

8 Future Work

A logical direction for future work is to develop mechanisms that facilitate the browsing and inspection of specifications that have been developed. A decision maker needs to be able to pinpoint how and where specific conditions appearing in the initial request have been accounted for in an existing specification for a delivered high-level information object. This may require the ability to issue "queries" against the components of a specification, or to query against the structure of the delivered high-level information object.

References

[1] Reference architecture. a draft developed at the ARPA Intelligent Integration of Information Workshop, November 1994, available at http://www.cs.colorado.edu/ dbgroup/i3-ref-arch.html.

[2] C. Batini, M. Lenzerini, and S.B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4), December 1986.

[3] Elisa Bertino. Integration of heterogeneous data repositories using object-oriented views. In *Proceedings of the IEEE Data Engineering Conference (DEC)*, 1991.

[4] R. G. G. Cattell, editor. *The Object Database Standard: ODMG 93.* Morgan Kaufmann Publishers, USA, 1994.

[5] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The tsimmis project: Integration of heterogeneous information sources. In *Proceedings of the 100th IPSJ Anniversary meeting*, 1994.

[6] Bogdan Czejdo and Malcolm Taylor. Integration of database systems using an object-oriented approach. In *Proceedings of the IEEE Data Engineering Conference (DEC)*, 1991.

[7] Bipin Desai and Richard Pollock. Mdas: Multiple schema integration approach. *IEEE Bulletin of the Technical Committee on Data Engineering*, 13(2), June 1990.

[8] M. Kaul, K. Drosten, and E.J. Neuhold. View system: Integrating heterogeneous information bases by object-oriented views. In *Proceedings of the IEEE Data Engineering Conference (DEC)*, 1990.

[9] James A. Larson, Shamkant B. Navathe, and Ramez Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4), April 1989.

[10] Amihai Motro. Superviews: Virtual integration of multiple databases. *IEEE Transactions on Software Engineering*, 13(7), July 1987.

[11] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A mediation system based on declarative specifications (extended version). submitted for publication 1995, available at http://www-db.stanford.edu/ yannis/yannis-papers.html.

[12] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogenous information sources. to appear in ICDE 95.

[13] M.A. Qutaishat, N.J. Fiddian, and W.A. Gray. Association merging in a schema meta-integration system for a heterogeneous object-oriented database environment. In *Lecture Notes in Computer Science: Advanced Database Systems.* Springer-Verlag, 1992.

[14] Radhika Reddy. Schema mapping for federated database retrievals. unpublished memorandum, DISC, Oregon Graduate Institute.

[15] Edward Sciore, Michael Siegel, and Arnon Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2), June 1994.

[16] Amit PPP. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3), September 1990.

[17] Richard Wang and Stuart Madnick. The inter-database instance identification problem in integrating autonomous systems. In *Proceedings of the IEEE Data Engineering Conference (DEC)*, 1989.

[18] Greg Washburn. *Vertical Information Management: A Framework to Support High-Level Information Requests in the Context of Autonomous Information Systems.* PhD thesis, University of Southwestern Louisiana, 1995.

[19] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3), 1992.

A Grammar for Abstract-ODL (AODL)

The BNF grammar for the representation of the abstract schema is presented here. This grammar allows for the specification of schemas at the name level only. This is a modification of the grammar for the Object Definition Language developed by the Object Database Management Group (ODMG). The primary modifications are the removal of type information and the association of domain names with attributes. The original grammar for ODL can be found in [4].

The meta-symbols used in the BNF notation are as follows:

- ::= denotes equivalence of the left hand side to the right hand side of the statement,
- | denotes an alternative selection,
- <symbol> denotes a nonterminal in the grammar,
- {symbol} denotes 0 or more occurrences of symbol,
- [symbol] denotes an optional symbol,
- **symbol** denotes a terminal of the grammar.

Note: some of these meta-symbols are also used as terminals in the language, attempts have been made to be make them boldface. However, it may be difficult to distinguish between a meta-symbol and a terminal at some points in the grammar (for example, < > is used to enclose the context).

```
<a_schema> ::= <a_class_definition> |<a_class_definition>
                                     <a_schema>

<a_class_definition> ::= interface {<a_class_name>[:
                                   <a_superclass_list>]
                           [<a_property_list>]
                         }

<a_superclass_list> ::= <a_superclass>|
                        <a_superclass>,<a_superclass_list>
<a_superclass> ::= <a_class_name>
<a_class_name> ::= <string>

<a_property_list> ::= <a_property_spec>;
                        |<a_property_spec>;<a_property_list>
<a_property_spec> ::= <a_attribute_spec>|<a_relationship_spec>
```

```
<a_attribute_spec> ::= attribute <domain_name>:
                                 <a_attribute_name>
                           context <context>

<domain_name> ::= <string>
<context> ::= <identifier [= " literal"] [<context>]
              {, identifier [= " literal"] [<context>]}>

<a_relationship_spec> ::= relationship
                          <target_of_path><traversal_path_name_1>
                          inverse <inverse_traversal_path>
                          [ {order_by <a_attribute_list>} ]

<traversal_path_name_1> ::= <string>
<target_of_path> ::= <target_class>
                    |<a_collection_type> <target_class>
<target_class> ::= <a_class_name>

<inverse_traversal_path> ::= <target_class>::<traversal_path_name_2>
<traversal_path_name_2> ::= <string>
<a_attribute_list> ::= <a_attribute_name> |
                        <a_attribute_name>,<a_attribute_list>
<a_collection_type> ::= set | bag | list | array
```

B Grammar for Abstract VIP Specification Language (AVIPSL)

The BNF grammar for the specification of an abstract vertical information plan is presented here. The expression of queries is based on the grammar presented for the Object Query Language developed as part of the ODMG standard by the Object Management Group. [4] The use of meta-symbols in the grammar is consistent with those presented in Appendix A.

```
<AVIP> ::= <op_spec> (<input_list>) [<context>]
<input_list> ::= <input> , <input_list> | <input>
<input> ::= <AVIP> | <query>

<op_spec> ::= [<output>] <op_name>
<query> ::= [<output>] <query_expr>

<output> ::= ident :  <arc_struct> [<context>]

<arc_struct> ::= struct(<field_list>) [<context>]
<field_list> ::= <field> {, <field>}
<field> ::= <ident> {<arc_struct> [context]}
<field> ::= <domain_name>:  <ident> [{<arc_struct> [context]}]

<arc_struct> ::= set(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= bag(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= list(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= array(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= <domain_name>:  <ident>

<ident> ::= string
<literal> ::= string
```

```
<domain_name> ::= string

<basic> ::= true | false | nil | <literal>

<context> ::= <identifier [= "<basic>"] [<context>]
              {, identifier [= "<basic>"] [<context>]}>

<query_expr> ::= <arc_struct> | <basic>
<query_expr> ::= for all identifier in <query_expr> :  <query_expr>
<query_expr> ::= exists identifier in <query_expr> :  <query_expr>
<query_expr> ::= <query_expr> in <query_expr>
<query_expr> ::= select [distinct] <query_expr>
                 from identifier in <query_expr>
                 {, identifier in <query_expr>}
                 [where <query_expr>]
<query_expr> ::= sortidentifier in <query_expr>
                     by <query_expr> {, <query_expr>}

<query_expr> ::= count (<query_expr>)
<query_expr> ::= sum (<query_expr>)
<query_expr> ::= min (<query_expr>)
<query_expr> ::= max (<query_expr>)
<query_expr> ::= avg (<query_expr>)

<query_expr> ::= group identifier in <query_expr>
                         by (identifier:  <query_expr>
                           {, identifier:  <query_expr>})
                         [with (identifier:<query_expr>
                            {, identifier:<query_expr>})]

<query_expr> ::= <query_expr> intersect <query_expr>
<query_expr> ::= <query_expr> union <query_expr>
<query_expr> ::= <query_expr> except <query_expr>

<query_expr> ::= <query_expr>.attribute_name
<query_expr> ::= <query_expr>.relationship_name

<query_expr> ::= <query_expr> = <query_expr>
<query_expr> ::= <query_expr> != <query_expr>
<query_expr> ::= <query_expr> < <query_expr>
<query_expr> ::= <query_expr> <= <query_expr>
<query_expr> ::= <query_expr> > <query_expr>
<query_expr> ::= <query_expr> >= <query_expr>

<query_expr> ::= not <query_expr>
<query_expr> ::= <query_expr> and <query_expr>
<query_expr> ::= <query_expr> or <query_expr>
```

C Grammar for Representation-ODL (RODL)

The BNF grammar for the representation of the representation-based schema is presented here. This grammar allows for the specification of a schema with its chosen representation. This is a modification of the grammar for the Object Definition Language developed by the Object Database Management Group (ODMG). The primary modifications are to include context information with the individual attributes. The original grammar for ODL can be found in [4]. The use of meta-symbols is consistent with those presented in Appendix A.

```
<r_schema> ::= <class_definition> |<class_definition> <r_schema>
```

```
<class_definition> ::= interface {<class_name>[:
                                <superclass_list>]
                          [<property_list>]
                        }


<superclass_list> ::= <superclass>|<superclass>,<superclass_list>
<superclass> ::= <class_name>
<class_name> ::= <string>

<property_list> ::= <property_spec>;
                    |<property_spec><property_list>
<property_spec> ::= <attribute_spec>|<relationship_spec>

<attribute_spec> ::= attribute<type>[ [<size>] ]:
                             <attribute_name>
                             semantic_context <s_context>
                             gap_context <g_context>

<type> ::= <atomic_literal>|<structured_literal>
          |<collection of objects or literals>
<atomic_literal> ::= integer | float | boolean
                    | character
<size> ::= <integer>

<s_context> ::= <context>
<g_context> ::= < ``<gap_status>'' [<context>],
                    Source = ``literal'' [<context>]>
<gap_status> ::= no gap | constant | estimate | hole
               |substitute <sub_context>
<sub_context> ::= <context>
<context> ::= <identifier [= " literal"] [<context>]
             {, identifier [= " literal"] [<context>]}>

<relationship_spec> ::= relationship
                        <target_of_path><traversal_path_name_1>
                        inverse <inverse_traversal_path>
                        [ {order_by <attribute_list>} ]

<traversal_path_name_1> ::= <string>
<target_of_path> ::= <target_class>
                    |<collection_type> <target_class>
<target_class> ::= <class_name>
<collection_type> ::= set | bag | list | array

<inverse_traversal_path> ::= <target_class>::<traversal_path_name_2>
<traversal_path_name_2> ::= <string>
<attribute_list> ::= <attribute_name> |
                    <attribute_name>,<attribute_list>
```

D Grammar for the RVIP Specification Language

The BNF grammar for the specification of a representation-based vertical information plan is presented here. The grammar is similar to the one presented in Appendix B. The primary difference is the use of types for the attribute specifications as opposed to domain names. The use of meta-symbols is consistent with that outlined in Appendix A.

```
<RVIP> ::= <op_spec> (<input_list>) [<context>]
```

```
<input_list> ::= <input> , <input_list> | <input>
<input> ::= <RVIP> | <query>

<op_spec> ::= [<output>] <op_name>
<query> ::= [<output>] <query_expr>

<output> ::= ident :  <arc_struct> [<context>]

<arc_struct> ::= struct(<field_list>) [<context>]
<field_list> ::= <field> {, <field>}
<field> ::= <ident> {<arc_struct> [context]}
<field> ::= <type>:  <ident> [{<arc_struct> [context]}]

<arc_struct> ::= set(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= bag(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= list(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= array(<arc_struct> [<context>]
                 {,<arc_struct> [<context>]}
<arc_struct> ::= <type>[[size]]:  <ident>

<ident> ::= string
<literal> ::= string
<type> ::= <atomic_literal> | <structured_literal> |
           <collection of objects or literals>
<atomic_literal> ::= integer | float | boolean
                   | character

<basic> ::= true | false | nil | <literal>

<context> ::= <identifier [= " <basic>"] [<context>]
              {, identifier [= " <basic>"] [<context>]}>

<query_expr> ::= <arc_struct> | <basic>
<query_expr> ::= for all identifier in <query_expr> :  <query_expr>
<query_expr> ::= exists identifier in <query_expr> :  <query_expr>
<query_expr> ::= <query_expr> in <query_expr>
<query_expr> ::= select [distinct] <query_expr>
                 from identifier in <query_expr>
                 {, identifier in <query_expr>}
                 [where <query_expr>]
<query_expr> ::= sort identifier in <query_expr>
                                   by <query_expr> {, <query_expr>}
<query_expr> ::= count (<query_expr>)
<query_expr> ::= sum (<query_expr>)
<query_expr> ::= min (<query_expr>)
<query_expr> ::= max (<query_expr>)
<query_expr> ::= avg (<query_expr>)

<query_expr> ::= group identifier in <query_expr>
                         by (identifier:  <query_expr>
                           {, identifier:  <query_expr>})
                         [with (identifier:<query_expr>
                            {, identifier:<query_expr>})]

<query_expr> ::= <query_expr> intersect <query_expr>
```

```
<query_expr> ::= <query_expr> union <query_expr>
<query_expr> ::= <query_expr> except <query_expr>

<query_expr> ::= <query_expr>.attribute_name
<query_expr> ::= <query_expr>.relationship_name

<query_expr> ::= <query_expr> = <query_expr>
<query_expr> ::= <query_expr> != <query_expr>
<query_expr> ::= <query_expr> < <query_expr>
<query_expr> ::= <query_expr> <= <query_expr>
<query_expr> ::= <query_expr> > <query_expr>
<query_expr> ::= <query_expr> >= <query_expr>

<query_expr> ::= not <query_expr>
<query_expr> ::= <query_expr> and <query_expr>
<query_expr> ::= <query_expr> or <query_expr>
```