Intelligent Integration of Information

Gio Wiederhold

ARPA-SISTO
and
Stanford University
March 15, 1993

Abstract

This paper describes and classifies methods to transform data to information in a three-layer, mediated architecture. The layers can be characterized from the top down as information-consuming applications, mediators which perform intelligent integration of information (I3), and data, knowledge and simulation resources.

The objective of modules in the I3 architecture is to provide end users' apoplications with information obtained through selection, abstraction, fusion, caching, extrapolation, and pruning of data. The data is obtained from many diverse and heterogeneous sources. The I3 objective requires the establishment of a consensual information system architecture, so that many participants and technologies can contribute. An attempt to provide such a range of services within a single, tightly integrated system is unlikely to survive technological or environmental change.

This paper focuses on the computational models needed to support the mediating functions in this architecture and introduces initial applicatiions. The architecture has been motivated in [Wied:92C].

Introduction

The concept of intelligent integration of information rests on the premise that knowledge is needed to integate factual observations into information. Automation becomes important as the volume of accessible data increases. We define information quite narrowly, as something that helps us make choices which will affect the future. Information, to be relevant to a decision maker, must be of modest size. Information embodies the concept of novelty, an essential aspect of information theory [Shan:48]. To gain novelty, various types of processing are required; factual data, as retrieved from an archive, are rarely novel without placing them into a new, current context.

SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0434...$1.50

Accumulating facts can provide larger bases for processing, but does not, by itself, create information. Going beyond accumulation, namely towards integration, initiates the process of matching data and their schemas, i.e., the knowledge about them, and creates information by exposing relationships that were previously unknown. Note that the match-making JOIN operation is the most critical and most processing intensive operation of the relational algebra.

However, for a relational JOIN operation to function, the data have to be perfectly matchable. We have here a conundrum. How can disparate data be guaranteed to match unless they derive from a coherent domain? Of course, if the JOIN operation only combines data that have been previously partitioned by PROJECT operations to normalize complex data into the relational framework, then the coherence condition for having correct joins is established. But now information creation is minimal. To be effective we must focus on multiple, autonomous sources. These will be heterogeneous.

Heterogeneity

Joining heterogeneous data is essential when trying to generate information. Heterogeneity exists at many levels; we review the issues bottom-up.

We will ignore **heterogeneity of hardware and operating systems**, although significant problems still exist at that level. The industrial-strength attention applied to this area is likely to reduce them to a manageable magnitude.

Dealing with **heterogeneity of organizational models** has been an active research area. Early work focused on mappings among network, hierarchical, relational and universal database models. Relatively simple, formal semantic information can help in transforming relational to object-structured data [Bars:92].

We must deal with problems of **heterogeneity in representation**, since it causes problems at the limits. When field lengths differ, the scope of the variables is likely to differ as well; consider 5-digit and 9-digit zipcodes. Numeric representations are less flexible than character strings, for instance the code for *as needed*: `PRN` is difficult to encode numerically and likely to be ignored in a numeric representation for drug-prescription frequency. Efforts directed towards standardizing database schemas have as a goal minimizing these heterogeneities [DISA:92].

Even variables of the same name and representation will have **heterogeneity of scope**, namely different databases will capture different sets and subsets of real-world objects. Recorded entries often partially overlap: `employees` in `personnel`, `payroll`, versus `healthplan` databases. Recent hires and part-time workers will not be in the `healthplan`. Contract personnel is not on the `payroll`. Knowledge about the scope of the database is needed to correctly process such data jointly. A `JOIN` over these data, even for one company, creates an intersection that would not represent the total. If `OUTERJOIN`s are used to combine such data, columns with missing data are created, and the application program has to interpret the incomplete information [Wied:92I].

Frequently there is **heterogeneity of level of abstraction.** Inventories can be aggregated by supplier, by material, or by date for tax purposes. Locations may be identified by zipcode in one domain and by town name in another. Unless atomic data is kept throughout, at the most detailed level of aggregation, this type of heterogeneity will cause partial results when the data must be joined [DeMi:89].

Even more dependent on source context is **heterogeneity of meaning**, namely the assignment of attribute values to real-world attributes. While the `weight` of automobile may be objective, it's `color` is not, and few people use the term the manufacturer's marketing department assigned. Similarly, `course grades` differ among schools and individual faculty members, and are inadequate to truly rank individuals [Cutk:93].

When databases collect more than current snapshots, problems of **heterogeneity of temporal validity** arise. Values have, often implicitly, different temporal ranges: an employee's `salary`, `pension benefits`, a `bonus`, and the value of any `stock options` cannot be simply aggregated. Furthermore, autonomous databases will not be synchronously updated. In a small enterprise, updating is often deferred to Friday morning, to get the payroll right. The fiscal year also differs widely among enterprises. Caching of histories permits going back in time and obtaining a snapshot that is temporally coherent, but a summary for the latest coherent time may be quite stale. Knowledge of enterprise operation is needed to provide the best possible information from diverse sources.

The I3 architecture places all resolution of heterogeneity into one mediator layer, although the range of the tasks is likely to require sublayers as well as domain-specific partitioning [Wied:92C].

Uncertainty and Risk

Since heterogeneity can rarely be completely resolved, uncertainty is introduced into the decision-making process. In daily life we always encounter heterogeneities and deal with them, in general, successfully. While uncertainties arise, they typically become negligible at the level of abstraction we operate on when making decisions. Formally, we use models of our world to abstract the data appropriately, and we can live with the remaining risk. Risk can also be reduced by devoting more resources to a task. Uncertainty in matching flight arrival and departure times are mitigated by adequate expected layover times.

Automation of uncertainty calculations requires much further work [Cohe:87]. While there are a number of uncertainty algebras, there is today little understanding of their domain applicability. If the uncertainty is at the limits of the representation and scopes, then the fuzzy model seems best. If, say, during the mediating transforms, the uncertainties are distributed throughout the range of values, then probability-based models may be more realistic.

In any case, the user's applications must be informed about resulting uncertainties, since the risk-taking is always assigned to the application and its owner.

Models

Essential to information processing is abstraction. Abstraction aggregates source data to higher levels, where heterogeneities of detail matter less. Abstractions are also needed to aggregate detail prior to joining it with data at a different level of abstraction. An example of the former is aggregation of an `employee's hours-worked` to prepare a payroll. The second case occurs when aggregating the `hours-worked` on projects to permit comparison with the project's production goals and cost estimates.

Aggregation implies the existence of an aggregation hierarchy, another aspect of knowledge about data. Simple, low-level hierarchies are often encapsulated into

object representations. This does mean that object integration interacts with the difficult, higher level of abstraction more rapidly than base data kept in relational form, where many joins just recapture the (unnormalized) object representation.

We need multiple models over most primitive data: the `payroll` example above implied a different aggregation than the time charged to projects. The distinct models represent distinct specialization of knowledge, here that of the payroll department versus that of the accounting department. Note that treating an employee as an *object* would not serve the projects' accounting model, since its breakdown is at a finer level of aggregation.

These observations reinforce the theme of this paper: knowledge is needed to generate integrated information. However, the examples should also dispell the notion that knowledge-based processing is awkward, costly, unpredictable, or worst of all, *sophisticated*. It is done everywhere, and the hardest choices to make are in the representation of the knowledge, since here the trade-offs of efficiency and flexibility are made. Hard-coded programs are at one side of the spectrum, rules represented in a high-level language at the other side, and intermediate representation include tables and compiled codes [Wied:92I].

Models for projecting the future.

Information which leads to action typically involves extrapolation of the current state into the future, since that is when the actions take effect. Whether a decision is about a long-term investment, say buying a house, or a short-term decision, say buying a meal, there exists a model of what the future will be like with or without the action.

Evaluation of the action requires processing of the history, of the current state, and the planned action or action sequence versus the known model. Often there will be many alternative actions, so that there will be multiple possible futures for a single historical model.

Qualitative differences.

When we use databases to retrieve information by selections and joins we obtain all the valid answers. That list is likely longer than the decision maker can evaluate. The list could be limited by only showing, say, just the first 10 choices. In a PROLOG model only the first valid result is presented. What a decision maker needs is a small number of *good* solutions, since there will be criteria external to the computer's knowledge. A simple ranking and showing the *best* solutions is often inadequate. We should present only present *qualitatively different* choices to the decision maker.

A poor qualitative difference in travel scheduling is:

```
Plan FF1:
UA59:  dep Wash.Dulles 17:10, arr LAX 19:49.
Plan FF2:
AA75:  dep Wash.Dulles 18:00, arr LAX 20:24.
```

A good example, where *good* is context dependent

```
Plan BP1:
UA59:  dep Wash.Dulles 17:10, arr LAX 19:49.
Plan BP2:
UA199:  dep Wash.Dulles 9:25, arr LAX 11:52.
```

Knowing what good means to a user requires again having a model. The 'busy person' model used here distinguishes working versus sitting in a plane during the workday.

The case labeled 'bad' may be 'good' in a passenger's model where `UA` frequent flyer miles are better than `AA`'s or where airline food, indistinguishable in the BP model, differs.

In any case, the model which defines the criteria is likely to form a hierarchy describing the user's, or more precisely, the user's application's preferences. What is qualitatively different depends on the user's view of the world. We have another instance where knowledge determines the appropriate reduction of data to information.

Hierarchies

The models are typically simple hierarchies, since they focus on a very specific task decomposition. For the meal purchase the alternative of meat versus fowl may be the primary qualitative difference, the choice of a wine is secondary. For buying a house, the three major considerations are location, perhaps followed by price and size. Any such hierarchy should be

a) visible to the user and the mediator
b) adjustable by the user

While we have focused on near-trivial examples to illustrate the points being made, the issues of layered management and long-term maintenance is critical in large information systems. Here conflicting demands due to the variety of required tasks abound. Modularity is essential.

A Demonstration

Before we can formalize the organization of the mediating modules, we have to gain experience with them. Examples of mediation are found in PACT [Cutk:92] and [Wied:92D], but we are now embarked on an industrial scale demonstration. The project involves the Lockheed F-22 Advanced Tactical Fighter, for which an Integrated Weapons Systems DataBase (IWSDB) is to

be provided. The IWSDB is to span technical specifications, design, manufacturing, and operational logistics. Work on the F-22 production models began last year (1992) although the first fully engineered plane will not fly until 1995 or 1996. Given the lifespan of such aircraft, we estimate that the IWSDB must operate to the year 2040.

To demonstrate the viability of a mediated architecture, mediators are being built for some early manufacturability validation tasks. For instance, one mediator is to support analysis of variation of tolerance in a composite structure. The application uses a conventional tool, the SAS statistical analysis package. Needed data are stored in several databases, as DEC VAX MIP for specifications, IBM CATIA for design drawings, and simple files for the logs from pre-production manufacturing and assembly tests. From all these sources data must be selected, converted, abstracted, and merged, prior to forwarding tables to the application.

Some later mediators will support browsing, since already more than 50 databases have been identified, and this number will grow greatly as more subcontractors are brought into the production process.

General tolerance management, tracking of engineering changes, assigning test and failure reports are all tasks requiring mediated services. The long term objective of this integration is to close the loops in the manufacturing processes, creating the infrastructure for responsive design, flexible manufacturing, and high product utilization.

To compose these mediated systems a flexible information transport mechanism is being established, based on KQML, tested earlier in PACT. KQML assumes a distributed, asynchronous environment and permits the transport of information in rule, object, equation, and even tuple form. The sender and receiver have, of course, to agree on the representation and the scope or *shared ontology*.

Conclusion

We have illustrated aspects of processing in a mediated architecture. The architecture is intended to envelop a number of technologies, ranging from extension to database services to knowledge-driven inferential schemes. When appropriate, mediation will also employ mathematically-based algorithms.

These processing functions can be classified into four categories that depend on each other

abstraction ⇔ merging ⇔ extrapolation ⇔ caching

These types of functions will be composed in a variety of ways. The actual selection and composition must be based on understanding the needs of the information-consuming application and the capabilities of the underlying data sources. Developing formal models, and representing them in a processable manner should allow rapid creation, adaptation, and reuse of mediating programs. By building early prototypes, we can learn new concepts of integration, and develop a scientific approach to integration technology.

Acknowledgement

The concepts of mediation have appeared under many names in a variety of systems. This work was specifically supported by ARPA, and we thank the many people involved for their contributions. Carl Friedlander of ISX commented on the draft.

References

We only cite references to immediate prior work. More references can be found in these citations.

[Bars:91] T. Barsalou, N. Siambela, A.M. Keller, and G. Wiederhold: "Updating Relational Databases through Object-based Views"; *ACM-SIGMOD 91*, Boulder CO, May 1991, pp.248–257.

[Cohe:87] P.R. Cohen, G. Shafer, and P.P. Shenoy: "Modifiable Combining Functions"; *Uncertainty in Artificial Intelligence*; Proc.of an AAAI Workshop, Jul.1987.

[Cutk:93] M.R. Cutkovsky, R.S. Engelmore, R.E. Fikes, M.R. Genesereth, T.R. Gruber, W.S. Mark, J.M. Tenenbaum, and J.C. Weber:"PACT: An Experiment in Integrating Concurrent Engineering Systems"; *IEEE Computer*, Jan.1993. pp.28–37.

[DeMi:89] L. DeMichiel: "Performing Operations over Mismatched Domains"; *IEEE Transactions on Knowledge and Data Engineering*, Vol.1 No.4, Dec. 1989, pp.485–493.

[DISA:92] *Dept. of Defense, Data Administration Strategic Plan*; Defense Information Systems Agency, Center for Inf. Mgmt, July 1992.

[Shan:48] C.E. Shannon and W. Weaver: *The Mathematical Theory of Computation*; reprinted by The Un.Illinois Press, 1962.

[Wied:92C] Gio Wiederhold: "Mediators in the Architecture of Future Information Systems"; IEEE Computer, March 1992, pp 38-49.

[Wied:92D] Gio Wiederhold: "Model-free Optimization"; *Darpa Software Technology Conference 1992*, Meridien Corp., Arlington VA, pp.83–96.

[Wied:92I] Gio Wiederhold: "The Roles of Artificial Intelligence in Information Systems"; *Journal of Intelligent Systems*; Vol.1 No.1, 1992, pp.35–56.