Information Integration for Collaborative Engineering

Michael N. Huhns
Microelectronics and Computer Technology Corporation
Information Systems Division
3500 West Balcones Center Drive
Austin, TX, U.S.A. 78759-6509
(512) 338-3651 or huhns@mcc.com

Abstract

The goal of our research is to develop methods for interoperating among separately developed information resources to enable them to be accessed and modified coherently. We are providing the computational infrastructure, from network communications to consistency maintenance, for collaborative engineering. In this paper, we describe our techniques for achieving interoperation and consistency at the semantic level. The techniques function at both compile time and run time. At compile time, a common ontology, provided by the Cyc knowledge base, is used to develop semantic mappings among resources. The mappings are based on schema-level models of each resource. At run time, a distributed truth maintenance system is used to maintain semantic consistency of data. The truth maintenance system is executed by knowledge-based mediating agents–one for each user, resource, and application–that actively monitor integrity constraints. In addition, the agents help locate and provide access to the most appropriate information for each user or application.

1 Introduction

World-wide production of manufactured goods is currently being affected by five related factors: 1) there are pressures for a shorter time-to-market, forcing a need for all aspects of product engineering—from conceptualization through delivery and maintenance—to be considered simultaneously, 2) there are changes in the artifacts of production, in that many products that used to be standardized are being specially designed for each customer, and more complicated products are being attempted, 3) there are increasing data, knowledge, and experience being accumulated about all aspects of product engineering, which can be used to aid future production processes, 4) there are now a plethora of tools for aiding product engineering, including tools for simulation, visualization, layout, test, aesthetics, compliance with standards, and manufacturability, and 5) the scope of the problem has increased to the point that teams of engineers are typically required. The overall trend in each of the five factors has been towards increasing the complexity of the engineering task. This in turn has placed additional demands on the computational aids for engineering, with the foremost demands being for interoperability and coherent access to all the relevant information available.

The purpose of the Carnot Project is to provide for the integration of a variety of data and processing resources, first within an enterprise, and then, across enterprises. The Carnot Project is developing the infrastructure for the concept of an enterprise-wide information space that allows easy access for application developers and engineers to the variety of different kinds of data and information that exist within large enterprises, multinational corporations, regional companies, etc. The project also addresses the problems of trying to coordinate interactions among companies. See Figure 1. This is the context for what is seen as a global information system, spanning businesses and various other kinds of providers of services.

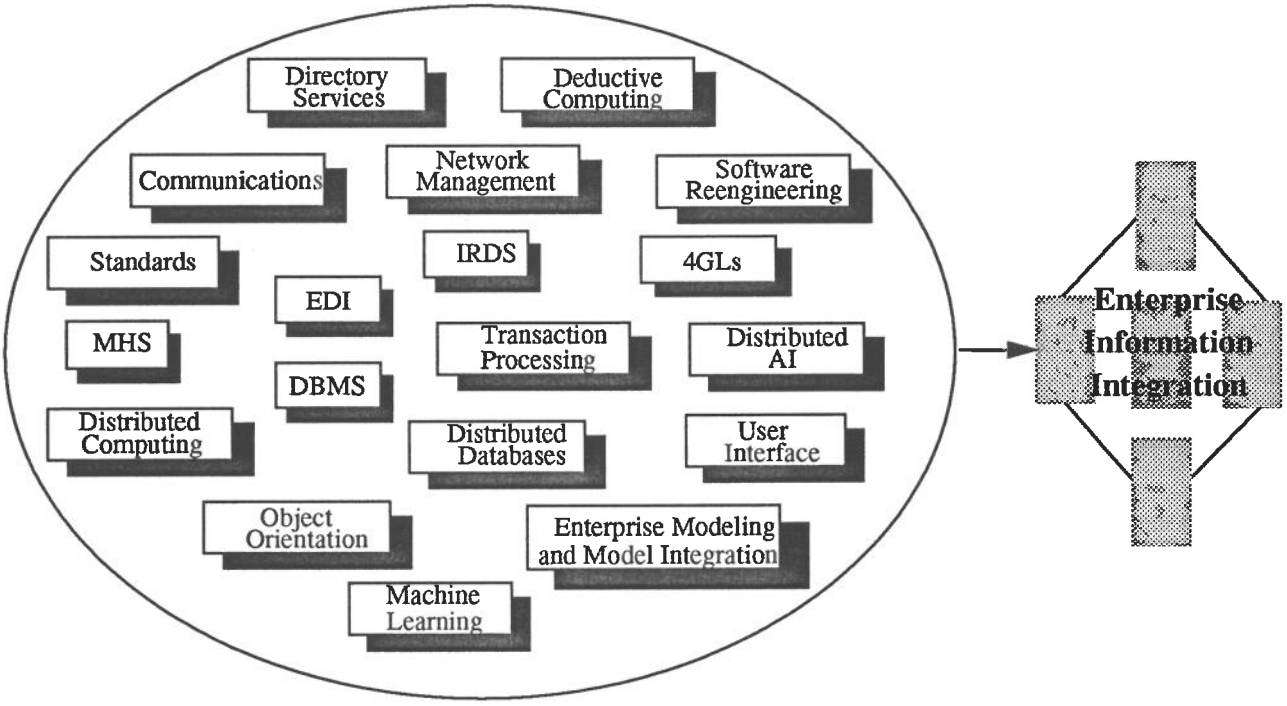

Figure 1: Technologies being used and integrated by Carnot to provide for enterprise-wide information spaces

1.1 Aspects of Enterprise-Wide Information Spaces

1.1.1 Heterogeneity

Today's corporate computing environments are heterogeneous, containing many independent information resources of different types, such as a database management system with its databases, an expert system with its knowledge base, an information repository, or an application program. Unfortunately, the resources are often mutually incompatible in syntax and semantics, not only due to the different types, but also due to mismatches in underlying hardware and operating systems, in data structures, and in corporate usage. Information resources attempt to model some portion of the real world, and necessarily introduce simplifications and inaccuracies in this attempt that result in incompatibilities.

The incompatibilities must be resolved [11], because 1) applications that span several of the resources must operate correctly, 2) a coherent picture of the enterprise is often needed for decision making and efficient business operations, and 3) applications must interoperate across a global enterprise. (Strategic business applications that require intercorporate linkage, e.g., linking buyers with suppliers, or intracorporate integration, e.g., producing composite information from engineering and manufacturing views of a product, are becoming increasingly prevalent.) One resolution would be to construct a homogeneous environment, but this is impractical. Instead, we utilize enterprise modeling and model integration to yield the appearance and effect of homogeneity.

1.1.2 Decentralization

Because enterprises are geographically dispersed, their resources are typically decentralized. Rather than masking this decentralization with an appearance of centralization, the Carnot Project attempts to take advantage of it to achieve improvements in performance, reliability, and availability.

1.1.3 Consistency

With its distributed facilities, the Carnot Project is addressing the problem of maintaining kinds of consistency among the various data resources that are being updated by multiple users in a system. In a geographically-dispersed, decentralized, enterprise-wide system, the definition of consistency is something that varies greatly from organization to organization, and from application to application. It is not feasible in all cases to take the rigid view of traditional database systems, i.e., that everything that is going to be updated will be updated at once, and if there is any failure during that process, then none of it will be updated. In practical situations, it is often the case that some database has been updated by a clerk in one location, the clerk has sent a memo to that effect to some other department, the memo has been lost, and for some period of time there actually is an inconsistency between data sets or databases at two different locations. Those inconsistencies are often tolerated, and in some cases it is found that companies spend a considerable amount of effort trying to remove those inconsistencies manually.

Carnot is providing facilities that allow users to express the kinds of consistency requirements that an organization ideally would like to impose, and is building tools and mechanisms that support maintenance of those consistency constraints in an automated fashion without manual intervention or manual auditing of database contents. The result is flexible and nonbrittle transactions.

1.1.4 Standards and Open Systems

There are many projects dealing with the notions of standards and open systems, involving international standards bodies and ad hoc groups of vendors and users. Our strategy in the Carnot Project is to build upon the ongoing standards work throughout ISO and CCITT so that we provide added value that intercepts where we expect commercial products to be in approximately three to five years.

For example, the Carnot Project has adopted the OSI communication protocols, because TCP/IP does not go above the OSI model's transport layer. The OSI protocols provide additional functionality above that. The work that has been done with the ISO Development Environment (ISODE) for OSI protocols has layered the upper portions of the OSI protocol stack on the Internet facility as well as a variety of other communication facilities. The Carnot approach to heterogeneity of communications protocols is to use OSI protocols at the upper layers, with support for bridging at the transport layer and below.

1.1.5 Capabilities Beyond Connectivity

The achievement of an enterprise-wide information space requires a considerable amount of work beyond that involved in just interconnecting the resources. One has to provide ways of being able to understand what the various resources are, and what their intended purpose is within the environment. For example, it is important to know what the fields in a database relation mean, beyond the fact that there is an integer in the third column. Carnot deals with the meaning of concepts as opposed to just simply recording the content in terms of data structure or some simple English-language label.

There has been quite a bit of progress made in terms of providing for interconnectivity, but there is a considerable amount of work yet to be accomplished in terms of making the interconnectivity usable to business and people. Connectivity by itself, while necessary, is not really sufficient, so the Carnot Project has the challenge of providing functionality that is necessary on top of the connectivity.

1.2 Carnot Architecture

The Carnot architecture, shown in Figure 2, consists of five layers of services: communication services, support services, distribution services, semantic services, and access services.

1.2.1 Communication Services

The communication services provide the user with a uniform method of interconnecting heterogeneous equipment and resources. While these services provide the interconnection, it is the groups of services above them that allow for their effective use. The communication services are built to run either on a GOSIP compliant OSI stack or on top of TCP/IP.

1.2.2 Support Services

The support services provide a collection of basic applications that make use of the interconnected resources supported by the communication services, and can be thought of as network-wide utilities available to applications and other higher level services. The support services consist of implementations of existing and emerging standards, such as the SQL Access Group's Remote Data Access (RDA), and CCITT's X.400 message handling and X.500 directory services. Work is progressing on additional support services, such as Information Resource Dictionary System (IRDS) interfaces, EDI, security via the ISO Upper Layers Security Model, and Network Management via SNMP, CMIS, and CMIP.

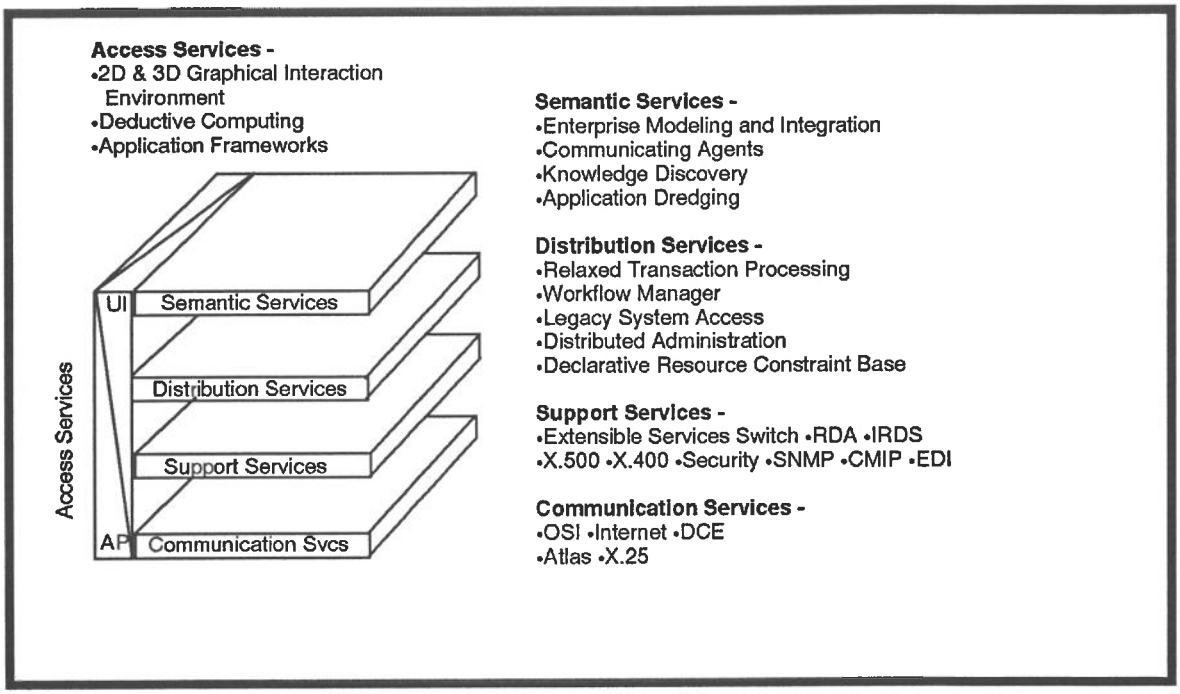

Figure 2: Layered architecture of the Carnot system

1.2.3 Distribution Services

The distribution services provide the basic facilities needed to coordinate the execution of work across a variety of heterogeneous resources, including databases, files, and other applications used in running a business. It is with these services that the Carnot system begins to add unique value to the standards efforts.

The main element in the distribution services is a distributed shell environment called the Extensible Services Switch (ESS), which coordinates the control flow and information flow among components of the various Carnot services. ESS is based on the MIT actor model. Distributed transaction/query managers build ESS scripts that reflect current business realities and accumulated corporate folklore. These managers build the scripts through interactions with directory services, repository managers, and deductive knowledge in Carnot's declarative resource constraint base. The declarative resource constraint base is essentially a collection of predicates that expresses business rules, interresource dependencies, consistency requirements, and contingency strategies throughout the enterprise. The declarative constraint base may adapt to a changing environment without modifying the application programs that manage the transactions; new applications can easily be added to the environment.

1.2.4 Semantic Services

The semantic services provide a global or enterprise-wide view of all the resources integrated within a Carnot system, rather than the view that occurs at the distribution services layer,

which essentially treats each resource as accessible but separate. The semantic services have a view of the distributed environment in which a common language can be used for communicating among resources. No attempt is made to make everything appear as a single centralized system, but rather as a common semantic framework.

This framework includes 1) MCC's Cyc knowledge base, whose ontology and reasoning mechanisms are used as a backdrop for representing sets of concepts that may be found in various databases and other resources; 2) MCC's Reasoning Architecture software, which provides mechanisms for independent, communicating agents, which in turn can encapsulate large groups of expertise as separate active entities within the Carnot system; and 3) schema integration and database design tools for updating, reconceptualizing, and generating information schemas.

In addition to representing the semantics of the enterprise-wide information space, we incorporate reasoning agents in the environment. The agents are knowledgeable about information resources that are local to them, and cooperate to provide global access to, and better management of, the information. For the practical reason that the systems are too large and dynamic (i.e., open) for global solutions to be formulated and implemented, the agents need to execute autonomously and be developed independently. To cooperate effectively, the agents must have *models* of each other and of the available information resources.

For such an open information environment, the questions arise: what should the models be, what are their constituents, where do they come from, and how can agents obtain them? We discuss the types of models that might be available in an enterprise and how agents can acquire them. We use the ontology developed for the large knowledge-based system, Cyc, for semantic grounding of the models. We then describe the use of actors—one per agent—who manage communications among the agents. Each actor maintains the relationship between its agent and the common ontology (in the form of articulation axioms), and updates it as the ontology changes or the agent itself evolves. Integrity knowledge from a comparison of the integrated models.

2 Modeling

Enterprise information modeling is a corporate activity that produces the models needed for interoperability. The resultant models should describe all aspects of a business environment, including

- databases
- database applications
- software repositories
- part description repositories
- expert systems, knowledge bases, and computational agents
- business work flows, and the information they create, use, maintain, and own, and
- the business organization itself.

The models provide online documentation for the concepts they describe. They enable application code and data to be reused, data to be analyzed for consistency, databases to be constructed automatically, the impact of change on an enterprise to be assessed, and applications to be generated automatically.

An enterprise might have many models available, each describing a portion of the enterprise and each constructed independently. For example,

- the information present in a database is modeled by the schema for the database, which is produced through a process of logical data modeling
- the information present in an object-centered knowledge base is modeled by the ontology of the objects, which is produced through ontological engineering
- process models, possibly in the form of Petri nets, are produced through logical process modeling
- STEP (Standard for the Exchange of Product model data) schemas, written in Express, are produced from component and physical process modeling.

Although it might appear that a homogeneous global model is needed for interoperability, there are good reasons for retaining the many individual models: 1) they are easier to construct than a single large model; 2) enterprises may be formed dynamically through mergers, acquisitions, and strategic alliances, and the resultant enterprises might have inherited many existing models; 3) because enterprises are geographically dispersed, their resources are typically decentralized; and 4) as enterprises (and thus models) evolve, it is easier to maintain smaller models.

Unfortunately, the models are often mutually incompatible in syntax and semantics, not only due to the different things being modeled, but also due to mismatches in underlying hardware and operating systems, in data structures, and in corporate usage. In attempting to model some portion of the real world, information models necessarily introduce simplifications and inaccuracies that result in semantic incompatibilities. However, the individual models must be related to each other and their incompatibilities resolved [11], because

- A coherent picture of the enterprise is needed to enable decision makers to operate the business efficiently and designers to evaluate information flows to and from their particular application.
- Applications need to interoperate correctly across a global enterprise. This is especially important due to the increasing prevalence of strategic business applications that require *intercorporate linkage*, e.g., linking buyers with suppliers, or *intracorporate integration*, e.g., producing composite information from engineering and manufacturing views of a product.
- Developers require integrity validation of new and updated models, which must be done in a global context.
- Developers want to detect and remove inconsistencies, not only among models, but also among the underlying business operations that are modeled.

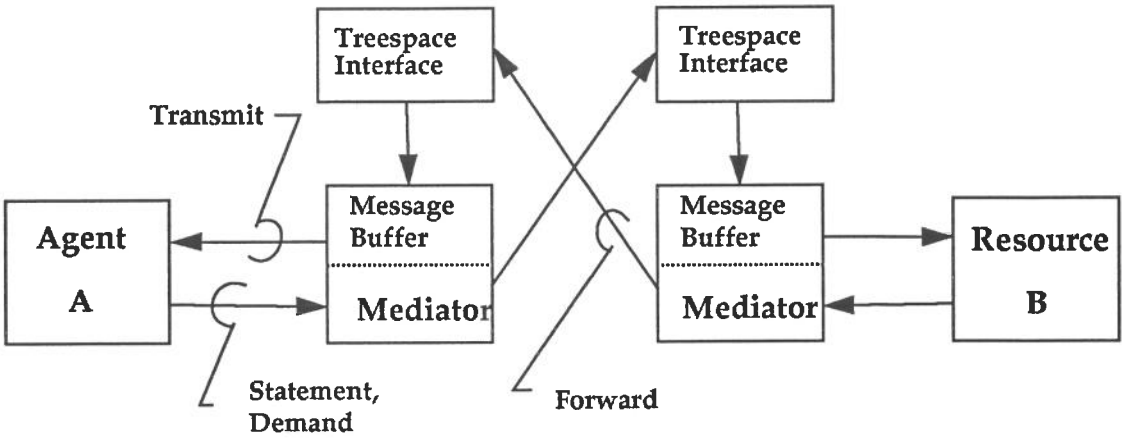

Figure 3: Each agent has an aide (Actor) that manages its communications through a tree space

We utilize a mediating mechanism based on an existing common ontology to yield the appearance and effect of semantic homogeneity among existing models. The mechanism provides logical connectivity among information resources via a semantic service layer that automates the maintenance of data integrity and provides an enterprise-wide view of all the information resources, thus enabling them to be used coherently. This logical layer is implemented as a network of interacting agents. Their interaction is via communication aides, and Figure 3 shows how the agents use Actors as the aides. The aides buffer messages, locate message recipients, and translate message semantics.

Significantly, the individual resources retain their autonomy. This is a fundamental tenet of the Carnot architecture [Woelk *et al.* 1992].

3 Semantic Integration via a Common Ontology

In order for agents to interact productively, they must have something in common, i.e., they must be either grounded in the same environment or able to relate their individual environments. We use an existing common context—the Cyc common-sense knowledge base [7]—to provide semantic grounding. The models of agents and resources are compared and mapped to Cyc but not to each other, making interoperation easier to attain. For n models, only n mappings are needed, instead of as many as $n(n-1)$ mappings when the models are related pairwise. Currently, Cyc is the best choice for a common context, because of 1) its rich set of abstractions, which ease the process of representing predefined groupings of concepts, 2) its knowledge representation mechanisms, which are needed to construct, represent, and maintain a common context, and 3) its size: it covers a large portion of the real world and the subject matter of most information resources.

The large size and broad coverage of Cyc's knowledge enables it to serve as a fixed-point for representing not only the semantics of various information modeling formalisms, but also the semantics of the domains being modeled. Currently, we can use models constructed using any of several popular formalisms, such as

- IRDS, IBM's AD/Cycle, or Bellcore's CLDM for entity-relationship models
- Ingres, Oracle, Objectivity, or Itasca for database schemas, and
- MCC's RAD or NASA's CLIPS for agent models.

Cyc's knowledge about metamodels for these formalisms and the relationships among them enables transactions to interoperate semantically between, for example, relational and object-oriented databases.

The relationship between a domain concept from a local model and one or more concepts in the common context is expressed as an articulation axiom. Each axiom has the form $ist(G\ \phi) \Leftrightarrow ist(C_i\ \psi)$, where ϕ and ψ are logical expressions and *ist* is a predicate that means "is true in the context." This axiom says that the meaning of ϕ in the common context G is the same as that of ψ in the local context C_i. Models are then related to each other—or translated between formalisms—via this common context by means of the articulation axioms, as shown in Figure 4. For example, an application's query about `Automobile` would result in subqueries to DB1 about `Car`, to DB2 about `Auto`, and to KB1 about `car`. Note that each model can be added independently, and the articulation axioms that result do not have to change when additional models are added. Also, applications need not be modified to access the extra information that becomes available. [Woelk *et al.* 1992] describes how transactions are processed semantically through the global and local views of several databases.

4 Semantic Transactions with Global and Local Views

As shown in Figure 5, a resource is integrated by specifying a syntax and a semantics translation between it and the global schema. The syntax translation provides a bidirectional translation between a local data manipulation language, DML_i, and the global context language, GCL, which is based on extended first-order logic. The semantics translation is a mapping between two expressions in GCL that have equivalent meanings. This is accomplished by a set of articulation axioms, as described above. At most n sets of axioms are needed for n resources.

After integration, one can access the resources through the global view, which gives the illusion of a single information resource, but requires that GCL be used. Queries and updates can also be issued against a local view. In that case, they are first translated into GCL and then into different DML_i and distributed to appropriate information resources. Thus applications need not be modified to access the extra information that becomes available.

To illustrate the idea, we describe how transactions are processed semantically through the global and local views of two integrated databases. The two databases have the same domain (hotels), but use different data models. The Fodor database, shown in Figure 6, uses an object-oriented data model. A hotel is represented as a class called `FodorInfo`. The features of hotel are represented as fields of the class or as other object classes pointed at by `FodorInfo`. The AAA database, shown in Figure 7, uses the relational model. A hotel is called `AAAInfo` and represented as a relation. The features of hotel, such as name and address, are represented as columns. Note that these schemas represent different perspectives and different information about hotels.

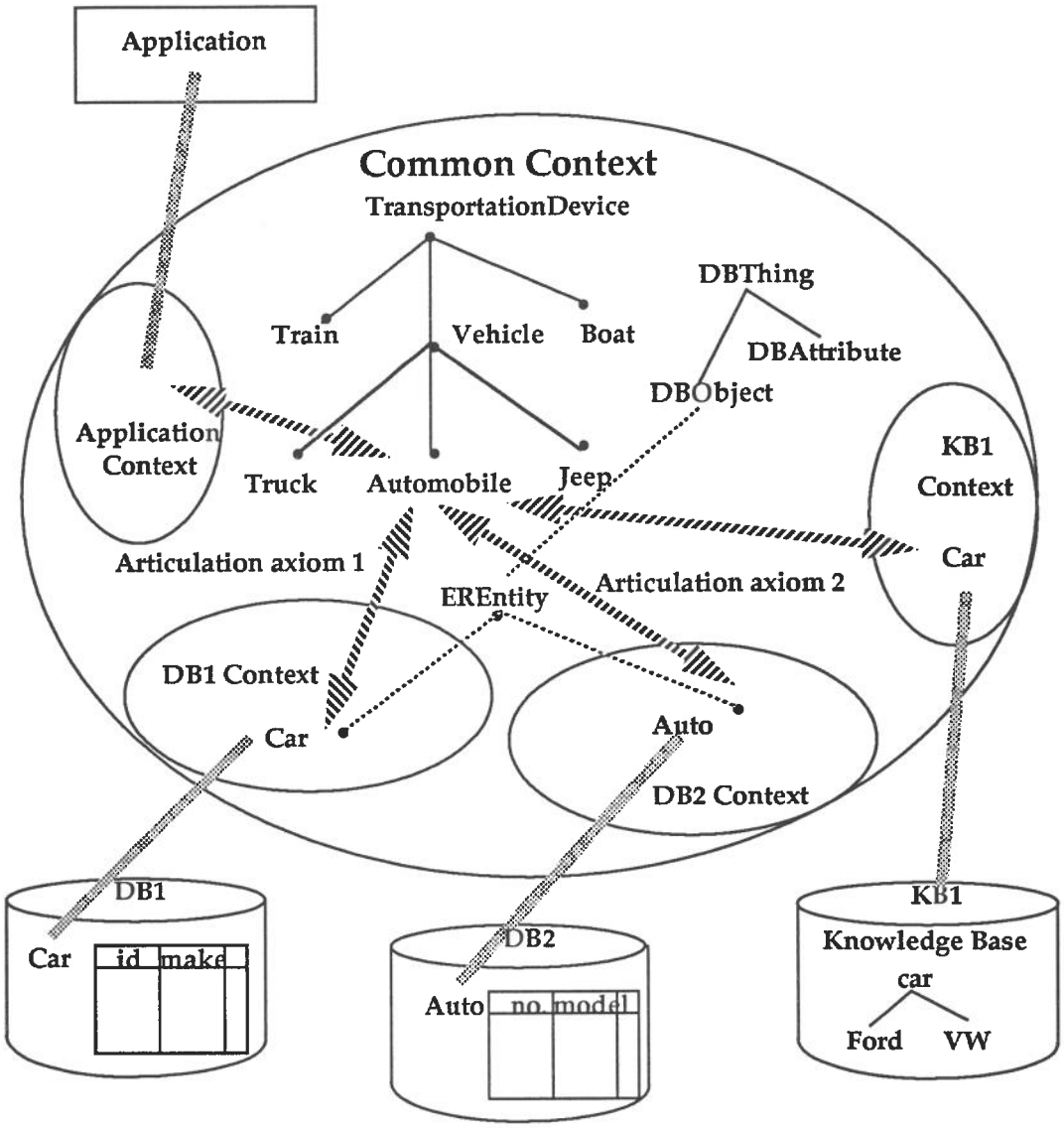

Figure 4: Concepts from different models are related via a common aggregate context by means of articulation axioms

Some of the Cyc concepts used in integrating these databases are the collections `Lodging` and `Restaurant`, and the predicates `hasAmenities`, `phoneNumber`, and `instanceOf`. Articulation axioms that map between the two database schemas and the global schema include

ist(G instanceOf(?H Lodging))⇔ist(AAA instanceOf(?H AAAInfo))
ist(G phoneNumber(?H ?P))⇔ist(AAA phone(?H ?P))
ist(G hasAmenities(?H ?F))⇔ist(AAA facility(?H ?F))
ist(G instanceOf(?H Lodging))⇔ist(Fodor instanceOf(?H FodorInfo))
ist(G hasAmenities(?H ?F))⇔ist(Fodor facility(?H ?X)∧facilityCode(?X ?F))

Based on its local view of the AAA database, an application might issue the following query for the phone numbers of hotels that have a restaurant:

```
SELECT phone FROM AAAInfo WHERE facility = "Restaurant"
```

This local SQL query is first translated into GCL by the SQL-GCL syntax translator:

```
instanceOf(?L AAAInfo) ∧ instanceOf(?R Restaurant) ∧ facility(?L ?R) ∧ phone(?L ?P)
```

This expression is then mapped by articulation axioms into a new expression whose semantics is meaningful in the global schema:

```
instanceOf(?L Lodging) ∧ instanceOf(?R Restaurant) ∧ hasAmenities(?L ?R)
                       ∧ phoneNumber(?L ?P)
```

This is then translated into different local queries using the appropriate articulation axioms in reverse. The translation for the Fodor local schema is

```
instanceOf(?L FodorInfo) ∧ facilities(?L ?F) ∧ facilityCode(?F ?R)
                         ∧ instanceOf(?R Restaurant) ∧ phone(?L ?P) ∧ phoneNum(?P ?N)
```

These queries are then translated into appropriate DML_i before being sent to the databases. For example, the query sent to the Fodor database is the following object-oriented expression (using ITASCA^{TM} syntax):

```
(SELECT (FodorInfo phones phoneNum)
        (= (path (some facilities) (some facilityCode)) "Restaurant"))
```

After the transactions are executed, the distributor assembles the results in the local view. In this example, the result is a column of phone numbers, because the local view of the AAA database is in the relational model. Note that these phone numbers come from two databases and the list may be much longer than that from the AAA database alone. However, users and applications need not be aware of the extra source of information.

5 The Development of Articulation Axioms

The articulation axioms for an information resource are developed in three phases: 1) *schema representation*, in which a Cyc context containing a model for the resource is produced, 2) *concept matching*, in which concepts from the model are matched with concepts in Cyc's base context—the global schema, and 3) *axiom construction*, in which the matches are converted automatically into articulation axioms by instantiating templates for these axioms with terms from the matches. The matching phase requires human interaction: frames may have to be created in the global schema and the model of the local schema may have to be augmented

with additional properties (semantics) to ensure a successful match.

5.1 Schema Representation

In this phase, we represent the given schema as a set of frames and slots in a Cyc context created specially for it. These frames are instances of frames describing the data model of the schema, e.g., (for a relational schema) `Relation` and `DatabaseAttribute`.

We define three types of frames for representing schemas: 1) `DatabaseSchema` frames, describing the schemas for different data models, 2) `DatabaseComponent` frames, describing the major components of schemas, such as relations and entities, and 3) `DatabaseLink` frames, describing different kinds of links used to refine and relate the major components. Every schema and every one of its components (relation, attribute, etc.) is an `instanceOf` these types and belongs to a context characterizing that schema. The slot `dBSchemaMt`, defined for `DatabaseSchema`, is used to express the relationship between an instance of a schema and its context. Information about the usage of a resource and the functionalities it provides (DDL, DML, transactions) are represented similarly, i.e., using frames such as `RelationalService`, `ERService`, `RelationalDDLType`, and `ERTransactionType`.

5.2 Matching

How articulation axioms are generated and how different schemas are integrated depends crucially on the process of matching them. For resource integration, the problem of matching is: given a (Cyc) representation for a concept, find its corresponding concept in the global schema. There are several factors that affect this phase: there may be a mismatch between the local and global schemas in the depth of knowledge representing a concept, and there may be mismatches between the structures used to encode the knowledge. For example, a concept in Cyc can be represented as either a collection or an attribute [7, pp. 339ff].

If the global schema's knowledge is more than or equivalent to that of the local schema's for some concept, then the interactive matching process described in this section will find the relevant portion of the global schema's knowledge. This knowledge will be in one of Cyc's two forms for concept representation. If the global schema has less knowledge than the local schema, then knowledge will be added to the global schema until its knowledge equals or exceeds that in the local schema; otherwise, the global schema would be unable to model the semantics of the resource. The added knowledge refines the global schema.

5.2.1 Lexical Heuristics

Finding correspondences between concepts in the local and global schemas is a subgraph-matching problem. We base subgraph matching on a simple string matching between the names or synonyms of frames representing the database schema and the names or synonyms of frames in the global schema. Matching begins by finding associations between attribute/link definitions and existing slots in the global schema. After a few matches have been identified, either by exact string matches or by a user indicating the correct match out of a set of candidate matches, possible matches for the remaining schema concepts are

greatly constrained. Conversely, after integrating an entity or object, possible matches for its attributes are greatly constrained.

5.2.2 Heuristics Based on Rules

While DBs are passive, KBSs are active. Intuitively, the rules of a KBS capture the operational semantics of the representations in the underlying KB; e.g., a `broker` is a broker because he (or it) participates in certain ways in certain financial transactions. Rules can thus be exploited in different ways in the matching phase.

Using heuristics based on rules for matching limits the set of possible matches significantly and improves the chances of obtaining a match that preserves the important semantic properties of the application. It also reduces the amount of input required from human designers, thereby making it simpler for them to guide the process of integration. Thus the added complexity of dealing with a KBS instead of a DB has immediate pay-backs in terms of the effort required for integration and the quality of the solution. In this abstract, we describe the simplest ways in which rules can be used.

Syntactic type checking: As an abstract example, consider a rule `If foo(?x ?y) bar(?y ?z) then baz(?x ?y ?z)`. By inspection, we can determine constraints such as the following on potential mappings of the predicates `foo`, `bar`, and `baz`. The type of the first argument of `foo` is compatible with the type of the first argument of `baz`: this is because if the rule ever fires, `?x` must be bound to something. Similarly, the types of the second argument of `foo` and of the first argument of `bar` must be compatible. This precludes matches in which `foo` and `bar` are matched with entities in Cyc that are incompatible in the appropriate sense.

Constraints on domains and ranges: While syntactic type checking is a useful heuristic, it is too cautious and can often be improved. The improvement is based on the intuition that rules are usually written for the entire concepts involved, not for any arbitrary subclasses. That is, the restrictions on the firing of a rule are explicit in its antecedent. This means that, in terms of the preceding example, the type of `foo` must not just be compatible with the type of the first argument of `baz`, it must be a subclass of the latter.

As a concrete example, consider a KBS containing a rule involving `broker` and `client`, where the rule refers to the relation of `isAdvisorOf` holding between an instance of `broker` and an instance of `client`. Thus in the matching process, we consider the binary relation `isAdvisorOf`, and the concepts, `broker` and `client`. The class `broker` must be matched to a subclass of the domain of `isAdvisorOf` and `client` to a subclass of its range. Similarly, if the rule makes a reference to the advice being given to `client` and involves a relation that restricts it to be financial in nature, the match of `isAdvisorOf` would be constrained to be with a subrelation of `isFinancialAdvisorOf`; i.e., the match would be improved. This would improve the other matches too, by constraining `broker` to have financial accreditation and the right to work in a particular market.

Capturing implicit restrictions: Reasoning about the rules in a KBS can also yield constraints on potential matches that are not obvious from the (local) schemas to be integrated, but that are nevertheless important in the global schema. These concern features that are implicit in the original local schemas, but need to be considered explicitly in the global schema, because it might contain entities that do not have them.

Consider a KBS with a rule that selects hotels for handicapped persons by looking at the preferences of a given person and the features available in a given hotel. Assume that the KBS implicitly considers only hotels that are wheelchair accessible, because those are the only ones listed in its local KB. Suppose this KBS is to be integrated with a general DB containing entries for hotels in general. For this integration to be correct, the generated articulation axioms must related `hotel` in the local schema to `hotel` $\cap$ `accessible` in the global schema. When `persons` in the original KBS is matched with `handicappedPerson` in the global schema, the relation `acceptableHotel` is constrained to match with the subrelation of `acceptableHotel` in the global schema that applies to the given subclass of `person`. This forces `hotel` to be matched with `accessibleHotel`, as desired. As a result, the correct articulation axiom would be generated such that where the KBS queried its local KB for hotels, it would now query the integrated DB for hotels that are wheelchair accessible. The rule can then fire as before.

5.2.3 Concept Matching

Let a_{ij}, $j = 1, 2, \ldots, n$ denote the attributes of concept E_i in a local schema. E_i is the domain of the attributes, i.e., the entity, relationship, relation, class, or object for which the a_{ij} are defined. Let s_j be the global schema slot that corresponds to, or matches, a_{ij}.

Observation 1 *The domain C_j of slot s_j is a generalization of the concept in the global schema that matches E_i.*

For example, the domain of the attributes `facility` and `phone` is the entity `AAAInfo`, whereas the domains of the corresponding Cyc slots `hasAmenities` and `phoneNumber` are the frames `HumanOccupiedStructure` and `Agent`, respectively. These are generalizations of Cyc's `Lodging`, which is the frame whose semantics most closely corresponds to `AAAInfo`.

As we match each of the attributes of E_i, we compute the common subdomain of the domains of their corresponding slots. The resulting common subdomains, although still generalizations of E_i, approximate it more and more closely.

Observation 2 *The "best" match for E_i is $\bigcap_{j=1}^{n} C_j$, the most general common subdomain (greatest lower bound in the generalization hierarchy) of the slot domains.*

In the above example, the most general common subdomain of `HumanOccupiedStructure` and `Agent` is `ServiceOrganization`, a generalization of `Lodging`. This would be suggested as the approximate match for `AAAInfo`. If no other attributes are matched, this would also be the *best* match that could be determined automatically for `AAAInfo`.

The greatest lower bound might not exist as a single frame in the global schema, however; it might be a set of frames. For example, the greatest lower bound would be the set {`HumanOccupiedStructure Agent`} if the frame `ServiceOrganization` did not exist. In such a case, a frame would be created in the Cyc knowledge base with the frames in the set listed as its generalizations.

5.3 Constructing Articulation Axioms

An articulation axiom is constructed for each match found. For example, the match between the relational attribute `phone` and the Cyc slot `phoneNumber` yields the axiom

ist(Cyc phoneNumber(?L ?N))⟺ist(AAA phone(?L ?N))

which means that the `phone` attribute definition determines the `phoneNumber` slot in the global schema, and vice versa. Articulation axioms are generated automatically by instantiating stored templates with the matches found.

6 Discussion

We described an experiment in integrating information models. In our approach, the integration of resource schemas is based on articulation axioms defined between two contexts: the context of a resource schema and a global schema context provided by the Cyc knowledge base. Our methodology is based on the following principles:

- Existing data should not have to migrate or be modified to achieve integration.
- Existing applications should not have to be modified due to integration.
- Users should not have to adopt a new language for communicating with the resultant integrated system, unless they are accessing new types of information.
- Resources should be able to be integrated independently, and the mappings that result should not have to change when additional resources are integrated.

The above principles are incorporated in an integration tool for assisting an administrator in integrating a resource and a transaction tool for providing users and applications with access to the integrated resources. The integration tool uses an extensive set of semantic properties to represent an information resource declaratively within the global schema and to construct bidirectional mappings between the resource and the global schema. The mappings are used by the transaction tool to translate queries and updates written against any local schema into the appropriate form for each information resource. These tools constitute part of the semantic services of Carnot [3], under development at MCC. Carnot will enable development of open applications that can be tightly integrated with information stored on existing, closed systems. The semantic service layer of Carnot provides facilities to specify and maintain the semantics of an organization's integrated information resources.

Thus our approach to resource integration allows a user or application to use a familiar local schema, while still benefiting from newly added resources.

References

[1] M. Ahlsen and P. Johannesson, "Contracts in Database Federations," in S. M. Deen, ed., *Cooperating Knowledge Based Systems 1990*, Springer-Verlag, London, 1991, pp. 293–310.

[2] O. P. Buneman, S. B. Davidson, and A. Watters, "Querying Independent Databases," *Information Sciences*, Vol. 52, Dec. 1990, pp. 1–34.

[3] P. E. Cannata, "The Irresistible Move towards Interoperable Database Systems," *First International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, April 7-9, 1991.

[4] C. Collet, M. N. Huhns, and W.-M. Shen, "Resource integration using a large knowledge base in Carnot," *IEEE Computer*, Vol. 24, No. 12, Dec. 1991, pp. 55–62.

[5] R. V. Guha, "Micro-theories and Contexts in Cyc Part I: Basic Issues," MCC Technical Report Number ACT-CYC-129-90, Microelectronics and Computer Technology Corporation, Austin, TX, June 1990.

[6] D. Heimbigner and D. McLeod, "A Federated Architecture for Information Management," *ACM Transactions on Office Information Systems*, Vol. 3, No. 3, July 1985, pp. 253–278.

[7] D. Lenat and R. V. Guha *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1990.

[8] W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, Sept. 1990, pp. 267–296.

[9] S. B. Navathe, S. K. Gala, S. Geum, A. K. Kamath, A. Krishnaswamy, A. Savasere, and W. K. Whang, "Federated Architecture for Heterogeneous Information Systems," *NSF Workshop on Heterogeneous Databases*, Dec. 1989.

[10] A. P. Sheth and S. K. Gala, "Attribute Relationships: An Impediment in Automating Schema Integration," *NSF Workshop on Heterogeneous Databases*, Dec. 1989.

[11] A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, Sept. 1990, pp. 183–236.

[12] J. de Souza, "SIS—A Schema Integration System," *Proc. Fifth British National Conference on Databases (BNCOD5)*, Cambridge University Press, 1986, pp. 167–185.

[13] R. Wang and S. E. Madnick, "Facilitating Connectivity in Composite Information Systems," *Data Base*, Fall 1989, pp. 38–46.

[Woelk *et al.* 1992] Darrell Woelk, Wei-Min Shen, Michael N. Huhns, and Philip E. Cannata, "Model-Driven Enterprise Information Management in Carnot," in Charles J. Petrie Jr., ed., *Enterprise Integration Modeling: Proceedings of the First International Conference*, MIT Press, Cambridge, MA, 1992.

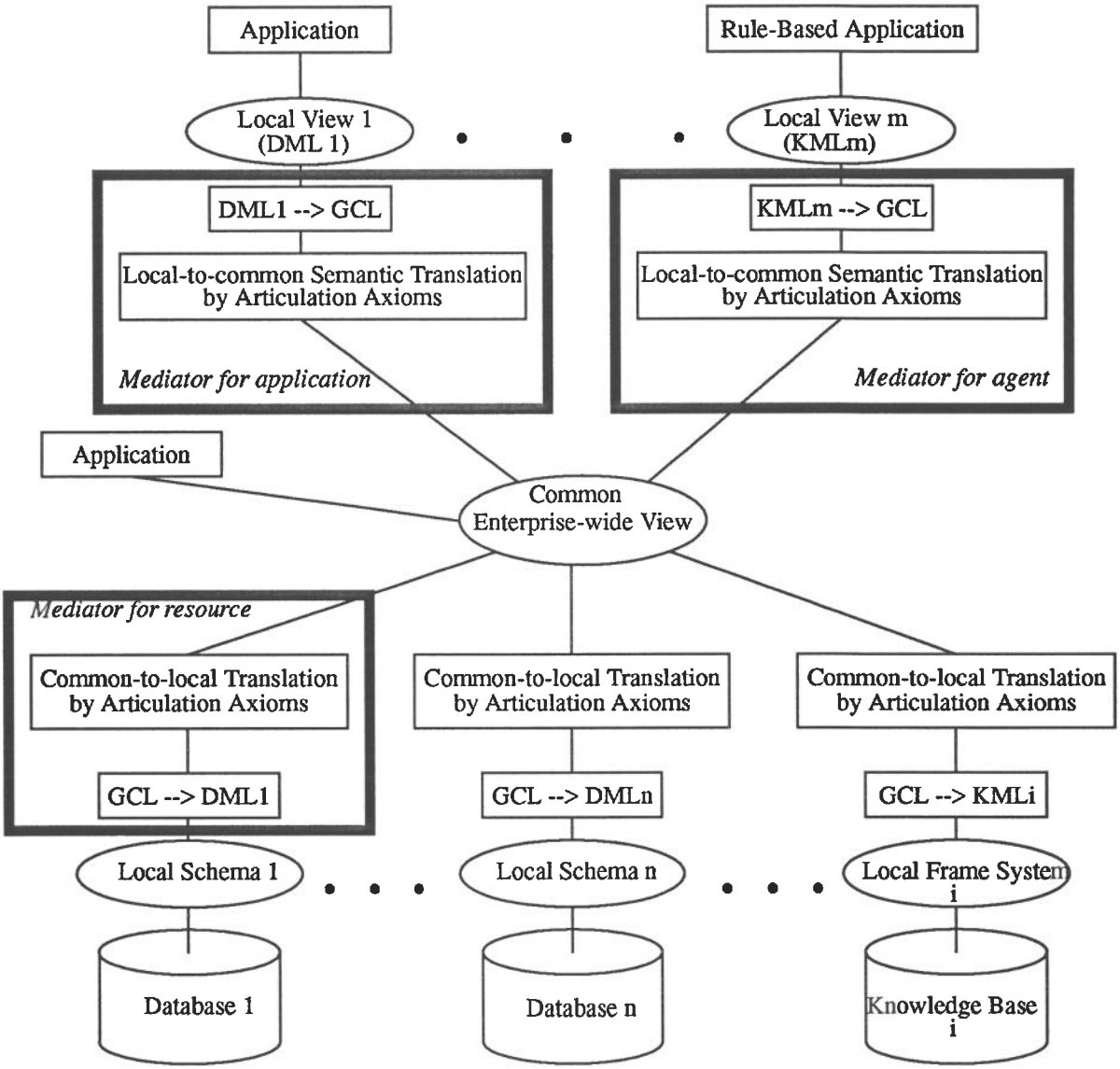

Figure 5: Global and local views in semantic transaction processing

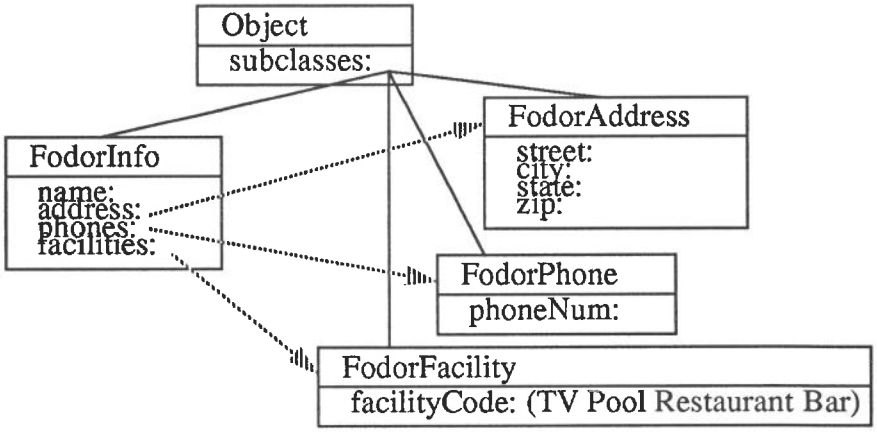

Figure 6: The object-oriented schema for the Fodor database

RELATIONS	COLUMNS
AAAInfo	name* address rateCode lodgingType phone facility
AAADirection	address* direction
AAACredit	name* creditCard*
AAARate	name* season* 1P 2P1B 2P2B XP fCode

Figure 7: A relational database schema for the AAA Tour Book database