[21] C. K. Ogden and I. A. Richards. *The Meaning of Meaning*. Harcourt Brace Jovanovich, 1989 (First published 1923).

[22] S. Prabhakar, J. Richardson, J. Srivastava, and E. Lim. Instance-level integration in federated autonomous databases. In *Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences*, January 1993.

[23] X. Qian. Semantic interoperation via intelligent mediation. In *Proceedings of the Third International Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, pages 228–231, April 1993.

[24] L. Raschid, Y. Chang, and B.J. Dorr. Query mapping and transformation techniques for problem solving with multiple knowledge servers. In *Proceedings of the Second International Conference on Information and Knowledge Management*, 1993.

[25] F. Saltor, M. Castellanos, and G. Garcia-Solaco. Suitability of data models as canonical models for federated databases. *ACM SIGMOD Record*, 20(4), December 1991.

[26] E. Sciore, M. Siegel, and A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, June 1994.

[27] A. Sheth. Semantic issues in multidatabase systems. *ACM SIGMOD Record*, 20(4), December 1991.

[28] A. Sheth and S. Gala. Attribute relationships: An impediment to automating schema integration. In *Proceedings of the Workshop on Heterogeneous Database Systems*, 1989.

[29] A. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In *Proceedings of the IFIP TC2/WG2.6 Conference on Semantics of Interoperable Database Systems*. Elsevier Scientific Publishers, November 1992.

[30] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[31] J. D. Ullman. *Principles of Database and Knowledge Base Systems*, volume 2. Computer Science Press, 1989.

[32] G. Wiederhold. Views, objects, and databases. *IEEE Computer*, 19(12):37–44, December 1986.

[33] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

[6] S. Ceri and G. Pelagatti. *Distributed Databases — Principles and Systems*. McGraw-Hill, 1984.

[7] J. Chomicki and W. Litwin. Declarative definition of object-oriented multidatabase mappings. In M. Ozsu, U. Dayal, and P. Valduriez, editors, *Distributed Object Management*. Morgan Kaufmann, 1993.

[8] L. DeMichiel. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4):485–493, December 1989.

[9] W. Du, R. Krishnamurthy, and M. Shan. Query optimization in a heterogeneous DBMS. In *Proceedings of the Eighteenth International Conference on Very Large Data Bases*, pages 227–291, 1992.

[10] R. Fagin. Horn clauses and database dependencies. *Journal of the ACM*, 29(4):952–985, October 1982.

[11] H. Gallaire, J. Minker, and J.-M. Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):153–185, June 1984.

[12] D. Gangopadhyay and T. Barsalou. On the semantic equivalence of heterogeneous representations in multimodel multidatabase systems. *ACM SIGMOD Record*, 20(4), December 1991.

[13] J. Geller, Y. Perl, and E. Neuhold. Structure and semantics in OODB class specifications. *ACM SIGMOD Record*, 20(4), December 1991.

[14] M. L. Ginsberg. Knowledge interchange format: The KIF of death. *AI Magazine*, 12(3):57–63, 1991.

[15] W. Kent. Solving domain mismatch and schema mismatch problems with an object-oriented database programming language. In *Proceedings of the Seventeenth International Conference on Very Large Data Bases*, 1991.

[16] A. Klug. Calculating constraints on relational expressions. *ACM Transactions on Database Systems*, 5(3):260–290, September 1980.

[17] R. Krishnamurthy, W. Litwin, and W. Kent. Language features for interoperability of databases with schematic discrepancies. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 40–49, 1991.

[18] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, second edition, 1987.

[19] R. Neches, R. E. Fikes, T. Finin, T. Gruber, R. S. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.

[20] J.-M. Nicolas and H. Gallaire. Data base: Theory vs. interpretation. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 33–54. Plenum Press, 1978.

Much research remains to be done. First, we have focused on three components of the mediation architecture, namely the mediation language, the query mediator, and the conflict detectors. Research is needed in the other components. Although it is often straightforward to translate participating schemas into the mediation language, it is tricky to translate the queries of participating databases into the mediation language if they could involve higher-order constructs, such as in the case of object-oriented databases. It is even more challenging to bind queries in the mediation language to the representational constructs of participating databases such that they could be efficiently evaluated.

Second, we have assumed that the knowledge in the mediator's knowledge base is available. How to obtain such knowledge is certainly an important issue. Although the acquisition of such knowledge is likely to be a highly interactive process, automated acquisition tools would be valuable.

Third, we have restricted ourselves to constraints, relationships, and queries that do not involve negation. The semantics of query mediation could certainly be generalized to allow negation, as long as for example the result is stratified [31]. The approach could also be easily generalized to deductive databases containing rules in addition to constraints.

Finally, research is needed in the autonomous optimization of mediated query evaluation. Due to the autonomy of participating databases, the query mediator often does not have access to the performance information that is crucial in query optimization. The query mediator needs a cost model that is independent of the implementation structures of participating databases. Techniques are also needed for the mediator to obtain performance information by querying [9].

Acknowledgment

The first author would like to thank Louiqa Raschid, Alon Levy, Michael Siegel, Amit Sheth, James Richardson, and Umesh Dayal for helpful discussions.

References

[1] Y. Arens and C. Knoblock. Planning and reformulating queries for semantically-modeled multidatabase systems. In *Proceedings of the First International Conference on Information and Knowledge Management*, 1992.

[2] T. Barsalou and D. Gangopadhyay. M(DM): An open framework for interoperation of multimodel multidatabase systems. In *Proceedings of the Eighth International Conference on Data Engineering*, 1992.

[3] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, December 1986.

[4] B. W. Beach. Connecting software components with declarative glue. In *Proceedings of the Fourteenth International Conference on Software Engineering*, pages 120–137, 1992.

[5] R. J. Brachman. The future of knowledge representation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 1082–1092, 1990.

Employee2(e#, citizen?)
Project2(p#, manager)
PE(p#, e#)

$\text{Employee2}(x, y_1) \wedge \text{Employee2}(x, y_2) \rightarrow y_1 = y_2$
$\text{Project2}(x, y_1) \wedge \text{Project2}(x, y_2) \rightarrow y_1 = y_2$
$\text{Project2}(x, y) \rightarrow (\exists z)\text{Employee2}(y, z)$
$\text{PE}(x, y) \rightarrow (\exists z)\text{Employee2}(y, z)$
$\text{PE}(x, y) \rightarrow (\exists z)\text{Project2}(x, z)$

The relationships between these two relational schemas, as contained in the mediator's knowledge base, would be:

$$\text{Employee1}(x, y) \equiv \text{Employee2}(x, y) \quad (S_7)$$
$$\text{Project1}(x, y) \equiv \text{Project2}(x, y) \quad (S_8)$$
$$\text{EP}(x, y) \equiv \text{PE}(y, x) \quad (S_9)$$

The original query Q_7 would be translated into the following relational query Q_9:

```
SELECT p#
FROM   EP t, Employee1 s                                    (Q9)
WHERE  t.e# = s.e# AND s.citizen?  = false.
```

A mediated query of Q_9 to DB2 would be:

```
SELECT p#
FROM   PE t, Employee2 s                                    (Q10)
WHERE  t.e# = s.e# AND s.citizen?  = false.
```

It is straightforward for the mediator to derive Q_{10} from Q_9 using its knowledge S_7 and S_9, which involves simple substitutions of equivalent subformulas.

6 Conclusion

We have presented a query mediation approach to the interoperation of autonomous heterogeneous databases containing data with semantic and representational mismatches. We have developed an architecture of interoperation that facilitates query mediation, and have formalized the semantics of query mediation and conflict detection. Queries are mediated between multiple databases, and users of a local database access data in multiple databases using the local language and schema, making both the data and the applications accessing the data in legacy databases interoperable. Queries are automatically mediated, relieving users from the difficult task of resolving semantic and representational mismatches. Semantic heterogeneity is separated from representational heterogeneity by minimizing the representational bias in the mediation language, which reduces the space of potential heterogeneity, and improves the efficiency of automated query mediation.

$$(\forall x, y)(x \in \text{Employee1} \wedge y \in x.\text{assignment} \rightarrow$$
$$(\exists x', y')(y' \in \text{Project2} \wedge x' \in y'.\text{team}$$
$$\wedge x.\text{e\#} = x'.\text{e\#} \wedge x.\text{citizen?} = x'.\text{citizen?}$$
$$\wedge y.\text{p\#} = y'.\text{p\#} \wedge y.\text{manager} = y'.\text{manager})) \qquad (S_5)$$

$$(\forall x, y)(y \in \text{Project2} \wedge x \in y.\text{team} \rightarrow$$
$$(\exists x', y')(x' \in \text{Employee1} \wedge y' \in x'.\text{assignment}$$
$$\wedge x'.\text{e\#} = x.\text{e\#} \wedge x'.\text{citizen?} = x.\text{citizen?}$$
$$\wedge y'.\text{p\#} = y.\text{p\#} \wedge y'.\text{manager} = y.\text{manager})) \qquad (S_6)$$

Suppose that users of DB1 pose a query to retrieve all projects that involve employees who are not US citizens, which could be expressed in an object-oriented SQL-like query language as follows:

```
SELECT Y
FROM   Employee1 X, Project1 Y                          (Q7)
WHERE  Y in X.assignment AND X.citizen?  = false.
```

A mediated query of Q_7 to DB2 would be:

```
SELECT new-obj(Project1, X.p#, X.manager)
FROM   Project2 X, Employee2 Y                          (Q8)
WHERE  Y in X.team AND Y.citizen?  = false.
```

The mediator would have to derive Q_8 from Q_7 using S_5 and S_6, which could be expensive because it involves inference in a logic for object-oriented databases such F-logic. If instead our mediation language is first-order predicate calculus, then translation into and out of the mediation language acts like the flattening and nesting operators in the nested relational model. In particular, DB1 would be translated into the following relational schema:

$$\text{Employee1(e\#, citizen?)}$$
$$\text{Project1(p\#, manager)}$$
$$\text{EP(e\#, p\#)}$$

$$\text{Employee1}(x, y_1) \wedge \text{Employee1}(x, y_2) \rightarrow y_1 = y_2$$
$$\text{Project1}(x, y_1) \wedge \text{Project1}(x, y_2) \rightarrow y_1 = y_2$$
$$\text{Project1}(x, y) \rightarrow (\exists z)\text{Employee1}(y, z)$$
$$\text{EP}(x, y) \rightarrow (\exists z)\text{Employee1}(x, z)$$
$$\text{EP}(x, y) \rightarrow (\exists z)\text{Project1}(y, z)$$

and DB2 would be translated into the following relational schema:

$$\text{Staff-Salary}(x, r \times y) \rightarrow \text{Employee-Salary}(x, y) \quad (I_{11})$$
$$\text{Staff-Title}(x, y') \wedge R(y, y') \rightarrow \text{Employee-Title}(x, y) \quad (I_{12})$$
$$\text{Staff-Manager}(x, z) \rightarrow (\exists y)(\text{Regular}(x, y) \wedge \text{Department}(y, z)) \quad (I_{13})$$
$$\text{Employee-Salary}(x, y) \wedge \text{Regular}(x, z) \rightarrow \text{Staff-Salary}(x, r \times y) \quad (I_{21})$$
$$\text{Employee-Title}(x, y) \wedge \text{Regular}(x, z) \wedge R(y, y') \rightarrow \text{Staff-Title}(x, y') \quad (I_{22})$$
$$\text{Regular}(x, y) \wedge \text{Department}(y, z) \rightarrow \text{Staff-Manager}(x, z) \quad (I_{23})$$
$$\text{Consultant}(x, r \times y) \rightarrow \text{Employee-Salary}(x, y) \quad (I_{31})$$
$$\text{Consultant}(x, y) \rightarrow \text{Employee-Title}(x, \texttt{consultant}) \quad (I_{32})$$

Now it is possible to perform the query mediation of Section 3.4 in the Horn fragment of first-order logic, namely Prolog. Moreover, it eliminates the need to derive the trivial mediated query Q_2 in order to derive the nontrivial mediated query Q_3.

In summary, an appropriate mediation language should be based on first-order logic, and should be minimized in terms of *representational bias*. First-order logic without uninterpreted function symbols and with predicate symbols of minimal arities is such a language.[6] It is the language of relational and deductive databases. With such a mediation language, query mediation could often be performed by theorem provers for fragments of first-order logic, such as Prolog or even Datalog, which are well known to be much more efficient than query mediation in full first-order logic.

5.4 Example

Let us consider two object-oriented databases DB1 and DB2 on employees and projects, and a many-to-many relationship between them about which employees work on which projects. Suppose that DB1 chooses to represent the relationship as a set-valued attribute of employee objects, as follows:

```
class  Employee1                      class  Project1
       e#          : string,                 p#       : string,
       citizen?    : boolean,                manager  : Employee1.
       assignment  : set(Project1).
```

Suppose also that DB2 chooses to represent the relationship as a set-valued attribute of project objects, as follows:

```
class  Employee2                      class  Project2
       e#          : string,                 p#       : string,
       citizen?    : boolean.                manager  : Employee2,
                                             team     : set(Employee2).
```

If the mediation language is full first-order logic, then the relationships between DB1 and DB2, as contained in the mediator's knowledge base, could be specified as follows, which state that every employee-assignment pair in DB1 corresponds to a project-team pair in DB2, and vice versa:

[6] We are not claiming that representational bias is completely eliminated in this language. In fact, the arguments of nonunary predicate symbols could always be ordered in more than one way. The key point is that representational bias is reduced sufficiently to make query mediation reasonably efficient.

5.3 Minimal Mediation Language

Hence, instead of trying to entirely separate semantics and representations, we aim for a mediation language in which the amount of representational bias (i.e., the number of ways that the same semantics could be represented differently) is minimized. This effectively separates semantic heterogeneity from representational heterogeneity, and reduces the complexity of query mediation. Semantic heterogeneity is handled by the mediator, while representational heterogeneity is handled by the translators.

The number of representational constructs should be minimized in the mediation language [25]. Languages based on relational, functional, or object-oriented models all qualify, in the sense that each of them contains only one representational construct — relation, function, or object. On the other hand, languages based on the ER model do not qualify, since they contain two representational constructs: entity and relationship.

However, minimizing the number of representational constructs alone is not enough, because the same representational construct might be capable of encoding the same semantics in more than one way. For example, the Stock relationship in Section 5.1 could be represented in functional models as S_1, S_3, or even the partial functions:

$$\text{Date} \mapsto (\text{Company} \mapsto \text{Price}) \qquad (S_4)$$

In fact, a relationship could always be encoded in many biased ways in a language with uninterpreted function symbols, depending on which argument is chosen as the output of the function. Hence, uninterpreted function symbols should not be allowed. Notice that attributes in object-oriented models are essentially functions from objects to values. Thus languages based on functional or object-oriented models are not good candidates as mediation languages. For example, the binary marriage relationship could be represented as a Wife attribute of the Male object, or as a Husband attribute of the Female object, or as a Family object with Wife and Husband attributes, etc.

Even predicate symbols could be biased in representing relationships, since an n-ary relationship could always be represented either by an n-ary predicate or by a collection of binary predicates. Hence the arities of uninterpreted predicate symbols should be minimized to represent only the atomic information. For example, we could decompose relation schemes Employee of Figure 4 and Staff of Figure 6 into atomic ones, as follows:

Employee-Salary(Name1, Salary1)
Employee-Title(Name1, Title1)
Employee-Phone(Name1, Phone#)

Staff-Salary(Name2, Salary2)
Staff-Title(Name2, Title2)
Staff-Office(Name2, Office)
Staff-Manager(Name2, Manager2)

This decomposition allows us to express most knowledge in Figure 9 as Horn clauses, as follows:

$$\begin{aligned}
&\text{Stock}(\texttt{Company1}, x, y) \equiv \text{Company1}(x, y)\\
&\qquad\qquad\vdots\\
&\text{Stock}(\texttt{CompanyN}, x, y) \equiv \text{CompanyN}(x, y).
\end{aligned}$$

In comparison, the relationships in [7, 12, 15, 17] relate schema B_1 to an *infinite set* of schemas $\mathcal{B}$, each representing a relationship similar to S_2 for some finite subset of the infinite Company domain, leading to the need for higher-order languages. Since we consider the interoperation of a *finite* set of databases, the need for higher-order languages does not arise. In fact, the infinite set $\mathcal{B}$ of schemas is an approximation of the partial functions:

$$\text{Company} \mapsto (\text{Date} \mapsto \text{Price}) \qquad (S_3)$$

which are equivalent to S_1. Just because S_3 cannot be represented directly in the flat relational model,[5] it does *not* imply that S_3 cannot be represented in first-order logic. We could easily think of first-order representations of S_3 in nested relational models, functional models, or object-oriented models.

Therefore, first-order logic should be sufficient as the semantic basis of mediation languages. Of course, this does not prevent us from having higher-order syntactic sugaring in mediation languages for the convenient specification of relationships between autonomous heterogeneous databases. The advantage of having first-order instead of higher-order logic as the semantic basis of mediation languages is obvious. Query mediation involves logical inference, and logical inference is more efficient in first-order than in higher-order logic.

5.2 Separate Semantics and Representation

A main source of complexity in the mediator's knowledge base is the potentially exponential combination of semantic and representational heterogeneity. Since representations are often application-specific, we would like to separate semantics from representations in the mediation language, thus making the semantics of data interoperable without requiring the representations of data to be interoperable. This minimizes the need for query mediation to deal with representational heterogeneity, and reduces the complexity of query mediation.

But what is semantics and what are representations? In the Dual Model [13], an attempt was made to distinguish the two in object-oriented databases, where the type system is considered to express representations and the class hierarchy is considered to express semantics. This distinction is inappropriate, because both type systems and class hierarchies capture representations, the former representing value semantics and the latter representing object semantics.

Semantics and representations correspond respectively to concepts and symbols in the meaning triangle [21]. For first-order logic, representations are captured by a theory, and semantics is captured by models of a theory. The relationship between semantics and representations is characterized by Gödel's completeness theorem. In essence, any language *encodes* some representation constructs, and any semantics has to be *expressed* in some language. There cannot be a language of semantics completely independent of representations, and hence semantics and representations cannot be completely separated in the mediation language.

[5] Notice that S_1 is an indirect representation of S_3 in the flat relational model.

Functional dependencies are a subclass of equality-generating dependencies, and techniques for deriving functional dependencies on queries are available [16].

For instance, the conflicts in the examples of Section 4.2 could be easily detected. There is a derived functional dependency on the answers of query Q_1 from Name1 to Title1, which is not satisfied in the answers of the mediated query Q_1 `union` Q_3 `union` Q_4. There is also a derived functional dependency on the answers of query Q_5 from Name1 to Manager1, which is not satisfied in the answers of the mediated query of Q_5.

5 The Mediation Language

As is obvious from Section 3, the efficiency of query mediation depends critically on the expressiveness of the mediation language — the more expressive the mediation language is, the less efficient the query mediation will be. In Section 3, we chose first-order predicate calculus as our mediation language. Here we discuss why this choice is justified, and what should be the appropriate features of a mediation language.

5.1 First-Order or Higher-Order Logic

Existing approaches [7, 12, 15, 17] argue that higher-order languages are necessary for reasoning with relationships between multiple databases, all of which demonstrate such need using the following stock example.

Suppose that schema B_1 contains the following relation scheme and constraint about companies, dates, and closing prices of stocks:

$$\begin{array}{l}\text{Stock(Company, Date1, Price1)}\\ \text{Stock}(x, y, z_1) \wedge \text{Stock}(x, y, z_2) \rightarrow z_1 = z_2.\end{array}$$

B_1 represents the partial functions:

$$\text{Company} \times \text{Date} \mapsto \text{Price} \qquad (S_1)$$

Now suppose that schema B_2 contains the following N relation schemes and constraints, for the dates and closing prices of the stocks of Company1, ..., CompanyN:

$$\begin{array}{l}\text{Company1(Date2, Price2)}\\ \text{Company1}(x, y_1) \wedge \text{Company1}(x, y_2) \rightarrow y_1 = y_2\\ \qquad\qquad \vdots\\ \text{CompanyN(Date2, Price2)}\\ \text{CompanyN}(x, y_1) \wedge \text{CompanyN}(x, y_2) \rightarrow y_1 = y_2.\end{array}$$

B_2 represents the partial functions:

$$\{\texttt{Company1}, \ldots, \texttt{CompanyN}\} \mapsto (\text{Date} \mapsto \text{Price}) \qquad (S_2)$$

Are the relationships between B_1 and B_2 representable in first-order logic? Yes, of course:

Name1	Manager1
Sam	Mark
Mark	Mark
Tom	Tom
Mary	John
Sam	John
⋮	⋮

Again, there cannot be a database that contains both DB1 and the fact that Sam's manager is John, since constraints C_1 and C_2 in Figure 5 state that employees have unique departments and departments have unique managers. The conflict is due to an inconsistency between the two databases. However, if we ask instead for the name and manager of regular employees who work in the database department:

```
SELECT Name1, Manager1
FROM   Regular, Department                                    (Q6)
WHERE  Dept = database AND Dname = database
```

then no data from DB2 would be accessed by query mediation, and no conflict would result, because no staff members are known in DB2 to work in the database department.

4.3 Constraint Derivation

Conflict detection is not computable using the definition of Section 4.1. Hence it is worthwhile to identify sufficient conditions under which conflicts could be detected efficiently.

Consider the interoperation of n databases b_i over schemas $B_i = (V_i, A_i)$ for $1 \leq i \leq n$. Also suppose that the mediator's knowledge base consists of theory $B = (\bigcup_{i=1}^{n} V_i \cup V, A)$ and structure b. Given a query q on B_1 with free variables $x_1, \ldots, x_m$ and a mediated query p of q on the combined schema $\bigcup_{i=1}^{n} B_i \cup B$, let C be an equality-generating dependency over vocabulary $V_1' = V_1 \cup \{P\}$ such that

$$A_1, (\forall x_1, \ldots, x_m)(P(x_1, \ldots, x_m) \equiv q) \vdash C$$

where P is a new predicate symbol not in V_1. Also let d be the structure over V_1' which is b_1 plus the answers of p in the combined database $\bigcup_{i=1}^{n} b_i \cup b$ assigned to P. Suppose that C is not satisfied in d. Notice that C must involve P. If there is a database b_1' over B_1 where $b_1 \subseteq b_1'$ such that the answers of q in b_1' are identical to the answers of p in $\bigcup_{i=1}^{n} b_i \cup b$, then there is a structure d' over V_1' which is b_1' plus the answers of p in $\bigcup_{i=1}^{n} b_i \cup b$ assigned to P. Since $d \subseteq d'$ and C is satisfied in d', we have that C is satisfied in d, a contradiction. Hence b_1' does not exist, and there is a conflict in the answers of p in $\bigcup_{i=1}^{n} b_i \cup b$.

The above analysis suggests that the equality-generating dependencies on query q could serve as sufficient conditions for detecting conflicts in the answers of the mediated queries of q.

Name2	Pay
Peter	80,000
⋮	⋮

Assuming that one US dollar is worth two German marks, the answers of the original query Q_1 of Section 3.4 are the following:

Name1	Title1
Sam	software engineer
⋮	⋮

and the answers of the mediated query Q_1 `union` Q_3 `union` Q_4 of Section 3.4 are the following:

Name1	Title1
Sam	software engineer
Sam	computer scientist
Peter	consultant
⋮	⋮

Obviously, there cannot be a database that contains both DB1 and the fact that Sam's job title is computer scientist, since constraint C_1 in Figure 5 states that employees have unique job titles. The conflict is due to an uncertainty in the mediator's knowledge of Figure 8, which states that an MTS could correspond to either a software engineer or a computer scientist. If we ask for the name and manager of regular employees in DB1:

```
SELECT Name1, Manager1
FROM   Regular, Department                                    (Q5)
WHERE  Dept = Dname
```

query mediation would return the following answers, even though DB2 does not know which department Mary is in:

be imported from other participating databases as well as the mediator's knowledge base. The additional data in the answers of p but not in the answers of q, combined with data in b_1, do not necessarily form a valid database over B_1, in which case a *conflict* has occurred.[4]

More formally, the answers of the mediated query p in the combined database $\bigcup_{i=1}^{n} b_i \cup b$ are *conflict-free* if there is a database b_1' over B_1, where $b_1 \subseteq b_1'$, such that the answers of q in b_1' are identical to the answers of p in $\bigcup_{i=1}^{n} b_i \cup b$:

$$(\forall x_1, \ldots, x_m)(b_1' \models q \iff \bigcup_{i=1}^{n} b_i \cup b \models p).$$

4.2 Example (Continued)

We continue with the example of Section 3.4. Suppose that the contents of DB1 are the following:

Name1	Salary1	Title1	Phone#
Sam	35,000	software engineer	856-2232
Mark	60,000	program director	856-1596
Tom	58,000	principal scientist	856-6015
⋮	⋮	⋮	⋮

Dname	Manager1
database	Mark
AI	Tom
⋮	⋮

Name1	Dept
Sam	database
Mark	database
Tom	AI
⋮	⋮

and the contents of DB2 are the following:

Name2	Salary2	Title2	Office	Manager2
Mary	100,000	distinguished MTS	2D-232	John
Sam	70,000	MTS	2C-301	John
⋮	⋮	⋮	⋮	⋮

[4]Notice the difference between this and the intuition of the federated database approach. For example, employees and staff members are similar concepts. Query mediation tries to access both concepts by importing staff members into the employee context to which name uniqueness should apply. In contrast, a federated database tries to access both concepts by creating another concept, say workers, which is the union of employees and staff members. Name uniqueness does not have to apply to the context of the new concept.

$$Q_1[f_i(x_1,\ldots,x_m)/y_i]_{1\le i\le n} \leftarrow P_1,\ldots,P_k$$
$$\vdots$$
$$Q_l[f_i(x_1,\ldots,x_m)/y_i]_{1\le i\le n} \leftarrow P_1,\ldots,P_k$$

where $f_i(x_1,\ldots,x_m)$ is a skolem function. For every m-ary predicate symbol P in $V_1,\ldots,V_n$, or V, we add a new m-ary predicate symbol P_0 and the following definite Horn clause:

$$P(x_1,\ldots,x_m) \leftarrow P_0(x_1,\ldots,x_m).$$

A deductive database (with equality) [11] could be constructed by taking these Horn clauses as the IDB. The EDB consists of, for every predicate P in $V_1,\ldots,V_n$, or V, the new predicate P_0 whose extent is the relation assigned to P by $b_1,\ldots,b_n$, or b.

Let M be the initial model of this deductive database [18]. Also let U be the universe of M which is the set of equivalence classes of ground terms over $\bigcup_{i=1}^n V_i \cup V \cup \{f_1,\ldots,f_n\}$, and $G \subseteq U$ be the set of equivalence classes containing ground terms over $\bigcup_{i=1}^n V_i \cup V$. Given a query q on B_1 with free variables $x_1,\ldots,x_m$, the *definite answers* of q in M are the answers of q in M that are in G^m.

Given a mediated query p of q on the combined schema $\bigcup_{i=1}^n B_i \cup B$. If p is sound, then every answer of p in the combined database $\bigcup_{i=1}^n b_i \cup b$ is a definite answer of q in M. If p is trivial, then every answer of p in $\bigcup_{i=1}^n b_i \cup b$ is an answer of q in b_1 and hence a definite answer of q in M. If p is complete, then every definite answer of q in M is an answer of p in $\bigcup_{i=1}^n b_i \cup b$.

When the IDB of this deductive database is bounded [31], there is a query p not involving IDB predicates, such that every answer of q in M is an answer of p in M and vice versa. Hence there is a query p' on the combined schema $\bigcup_{i=1}^n B_i \cup B$, such that every definite answer of q in M is an answer of p' in the combined database $\bigcup_{i=1}^n b_i \cup b$ and vice versa. In other words, p' is the sound and complete mediated query of q.

In general, we could view query mediation as the first-order approximation of definite answers in the initial model of a deductive database, which is formed by taking participating databases as the EDB, and by taking (the skolemization of) the mediator's knowledge and the constraints in participating schemas as the IDB. The more complete a mediated query is, the closer its answers are to the definite answers of the original query in the initial model. The boundedness of the IDB serves as a sufficient condition for the existence of sound and complete mediated queries.

4 Conflict Detection

4.1 The Semantics of Conflicts

Consider the interoperation of n databases b_i over schemas $B_i = (V_i, A_i)$ for $1 \le i \le n$. Also suppose that the mediator's knowledge base consists of theory $B = (\bigcup_{i=1}^n V_i \cup V, A)$ and structure b. Given a query q on B_1 with free variables $x_1,\ldots,x_m$, suppose that the evaluation of q in b_1 is replaced by the evaluation of the mediated query p of q in the combined database $\bigcup_{i=1}^n b_i \cup b$. Intuitively, the need for query mediation arises because b_1 does not capture a complete picture of the real world as far as q is concerned, thus additional data (in the answers of p) need to

```
SELECT Name2, consultant
FROM   Consultant t                                          (Q4)
WHERE  t.Pay/r < 50,000.
```

where `consultant` in the select clause is a constant. It could be verified that the combined query Q_1 `union` Q_3 `union` Q_4 is the sound, nontrivial, and complete mediated query of Q_1.

3.5 Meaning of Query Mediation

Consider the interoperation of n databases b_i over schemas $B_i = (V_i, A_i)$ for $1 \leq i \leq n$. Also suppose that the mediator's knowledge base consists of theory $B = (\bigcup_{i=1}^{n} V_i \cup V, A)$ and structure b. Given a query q on B_1, the objective of query mediation is to replace the evaluation of q in b_1 by the evaluation of the sound, nontrivial, and complete mediated query p of q in the combined database $\bigcup_{i=1}^{n} b_i \cup b$. The soundness of p ensures that such replacement is meaningful with respect to the constraints in $B_1, \ldots, B_n$ and the relationships in B. In the example of Section 3.4, the mediated queries ensure that salary values from DB2 are properly converted to US dollars before they are compared to the constant $50,000$, and that job titles from DB2 are properly converted according to the correspondence relation R in Figure 8 before they are returned to the user.

If the mediated query p is trivial, then the answers of p are contained in the answers of the original query q, since databases $b_1, \ldots, b_n$ are valid and $\bigcup_{i=1}^{n} b_i \models (\forall x_1, \ldots, x_m)(p \rightarrow q)$. Hence query mediation does not yield additional data. In the example of Section 3.4, the answers of the mediated query Q_2 are contained in the answers of the original query Q_1. Hence replacing Q_1 by Q_2 does not yield additional data. In comparison, the answers of mediated queries Q_3 and Q_4 could yield additional data, because DB2 might contain staff members or consultants who are not recorded as employees in DB1.

The completeness of the mediated query p ensures that all the data that satisfy the original query q, whether they are in databases $b_1, \ldots, b_n$, or the mediator's knowledge base b, will be accessed by evaluating p. In the example of Section 3.4, the mediated query Q_1 `union` Q_3 `union` Q_4 ensures that the name and title of all the employees who earn less than $50,000 will be accessed, whether they are recorded as employees in DB1, or as staff members or consultants in DB2.

Query mediation could be easily automated with the help of a first-order theorem prover that is tuned to goal-directed reasoning, such as a theorem prover based on algebraic rewriting.

3.6 Semantics of Query Mediation

Consider the interoperation of n databases b_i over schemas $B_i = (V_i, A_i)$ for $1 \leq i \leq n$. Suppose that the mediator's knowledge base consists of theory $B = (\bigcup_{i=1}^{n} V_i \cup V, A)$ and structure b. Every equality-generating dependency in $A_1, \ldots, A_n$ is a definite Horn clause. Through skolemization, every tuple-generating dependency in $A_1, \ldots, A_n$, or A of the form

$$(\forall x_1, \ldots, x_m)(P_1 \wedge \cdots \wedge P_k \rightarrow (\exists y_1, \ldots, y_n)(Q_1 \wedge \cdots \wedge Q_l))$$

could also be transformed into l definite Horn clauses:

A mediated query p is *complete* if it is logically implied by all possible mediated queries p' of q. Intuitively completeness means that every valid answer of the original query is an answer of the mediated query. This is expressed as follows:

$$\bigcup_{i=1}^{n} A_i \vdash (\forall x_1, \ldots, x_m)(p' \rightarrow p)$$

for every mediated query p' of q.

When the mediator's knowledge base is empty: $A = \emptyset$, any query q is the sound, trivial, and complete mediated query of itself. However, the sound, nontrivial, and complete mediated query of q does not exist. In general, if a sound, nontrivial, and complete mediated query exists, then it is always unique up to equivalence, because if p and p' are two sound, nontrivial, and complete mediated queries of q, then they are equivalent: $\bigcup_{i=1}^{n} A_i \vdash (\forall x_1, \ldots, x_m)(p \equiv p')$.

3.4 Example (Continued)

In the example of Section 3.2, suppose that we pose the following query to DB1,[3] which asks for the name and title of all the employees who earn less than $50,000:

```
SELECT Name1, Title1
FROM   Employee                                                    (Q1)
WHERE  Salary1 < 50,000.
```

Using constraints C_4 and C_5 in Figure 5, which state that every regular employee is an employee and Dept is a foreign key to Department, we could derive from Q_1 the following sound but trivial mediated query, which asks for the name and title of all regular employees who earn less than $50,000:

```
SELECT Name1, Title1
FROM   Employee t, Regular s, Department u                         (Q2)
WHERE  t.Name1 = s.Name1 AND s.Dept = u.Dname AND Salary1 < 50,000.
```

This mediated query is sound because every name-title pair in the answers of Q_2 is also in the answers of Q_1. It is trivial however since the mediator's knowledge is not needed to derive it. Using relationship I_1 in Figure 9, which says that every staff member is a regular employee, we could derive from Q_2 the following sound and nontrivial mediated query, which asks for the name and (converted) title of staff members who earn less than (converted) $50,000:

```
SELECT Name2, Title1
FROM   Staff t, R s                                                (Q3)
WHERE  t.Salary2/r < 50,000 AND t.Title2 = s.Title2
```

where R is the correspondence relation between job titles in Figure 8, and r is the exchange rate between US dollars and German marks. Similarly, using relationship I_3 in Figure 9, which says that every consultant is an employee with job title `consultant`, we could derive another sound and nontrivial mediated query of Q_1, which asks for the name of consultants who earn less than (converted) $50,000:

[3] For ease of reading, all the queries in this section are expressed in SQL. It is straightforward to translate them into first-order predicate calculus.

$$\text{Staff}(x, r \times y, z', u', w) \wedge R(z, z')$$
$$\rightarrow (\exists u, v)(\text{Employee}(x, y, z, u) \wedge \text{Regular}(x, v) \wedge \text{Department}(v, w)) \quad (I_1)$$
$$\text{Employee}(x, y, z, u) \wedge R(z, z') \wedge$$
$$\text{Regular}(x, v) \wedge \text{Department}(v, w) \rightarrow (\exists u')\text{Staff}(x, r \times y, z', u', w) \quad (I_2)$$
$$\text{Consultant}(x, r \times y) \rightarrow (\exists z)\text{Employee}(x, y, \texttt{consultant}, z) \quad (I_3)$$

Figure 9: Relationships in the Mediator's Knowledge Base

3.2.3 Semantic and Representational Mismatches

There are both semantic and representational mismatches between the schemas of DB1 and DB2. The domain mismatches in salaries and job titles are examples of semantic mismatches. In addition, DB1 cares about phone numbers, while DB2 cares about offices.

There are also representational mismatches. The relationship between regular employees and managers is represented indirectly in DB1 through departments, but the same relationship between staff members and managers is represented directly in DB2. In addition, consultants are represented by a value in the domain of attribute Title1 in DB1, but by a relation scheme in DB2.

Moreover, the mediator's knowledge about these mismatches is uncertain and sometimes incomplete. For example, a software engineer in DB1 could be either a programmer or an MTS in DB2, and there could be a job title `contract administrator` in DB1 whose correspondence in DB2 is not known to the mediator.

3.3 Properties of Query Mediation

Consider the interoperation of n databases b_i over schemas $B_i = (V_i, A_i)$ for $1 \leq i \leq n$. Also suppose that the mediator's knowledge base consists of theory $B = (\bigcup_{i=1}^{n} V_i \cup V, A)$ and structure b. Given a query q on B_1 with free variables $x_1, \ldots, x_m$, a *mediated query* p of q is a query on the combined schema $\bigcup_{i=1}^{n} B_i \cup B = (\bigcup_{i=1}^{n} V_i \cup V, \bigcup_{i=1}^{n} A_i \cup A)$ with the same list of free variables. Notice that, although q is expressed on one schema B_1, p could encompass multiple schemas from $B_1, \ldots, B_n$ and the mediator's knowledge base B.

A mediated query p is *sound* if it logically implies the original query using the mediator's knowledge. Intuitively soundness means that every answer of the mediated query should be a valid answer of the original query. This is expressed as follows:

$$\bigcup_{i=1}^{n} A_i \cup A \vdash (\forall x_1, \ldots, x_m)(p \rightarrow q).$$

A mediated query p is *trivial* if it is sound even when the mediator's knowledge base is empty: $A = \emptyset$. Intuitively trivialness means that every answer of the mediated query is obtainable by asking the original query. This is expressed as follows:

$$\bigcup_{i=1}^{n} A_i \vdash (\forall x_1, \ldots, x_m)(p \rightarrow q).$$

Staff(Name2, Salary2, Title2, Office, Manager2) (R_1')
Consultant(Name2, Pay) (R_2')

Figure 6: Schema of DB2

$$\text{Staff}(x, y_1, z_1, w_1, u_1) \wedge \text{Staff}(x, y_2, z_2, w_2, u_2) \rightarrow y_1 = y_2 \wedge z_1 = z_2 \wedge w_1 = w_2 \wedge u_1 = u_2 \quad (C_1')$$

$$\text{Consultant}(x, y_1) \wedge \text{Consultant}(x, y_2) \rightarrow y_1 = y_2 \quad (C_2')$$

$$\text{Staff}(x, y_1, z_1, w_1, u_1) \rightarrow (\exists y_2, z_2, w_2, u_2)\text{Staff}(u_1, y_2, z_2, w_2, u_2) \quad (C_3')$$

Figure 7: Constraints of DB2

mediator's knowledge base. In addition, suppose that there is not an exact one-to-one correspondence between job titles in DB1 and job titles in DB2. The mediator keeps track of the correspondence between job titles in the two databases as relation R in Figure 8.

Title1	Title2
`software engineer`	`programmer`
`program director`	`department head`
`principal scientist`	`distinguished MTS`
`computer scientist`	`MTS`
`software engineer`	`MTS`
⋮	⋮

Figure 8: Relation R in the Mediator's Knowledge Base

The mediator's knowledge also states that staff members are regular employees and vice versa, and consultants are employees with the job title `consultant`. These are expressed by the dependencies in Figure 9.

3.2 An Example of Query Mediation

Suppose that a US-based company is merged with a Europe-based company, which brings the need to interoperate the two relational databases DB1 and DB2 previously developed and maintained independently by the two companies respectively.

3.2.1 Schemas

The schema of DB1 in Figure 4 consists of three relation schemes. Figure 5 shows five constraints in DB1. The first three constraints state that employees, departments, and regular employees have unique names. The last two constraints state that every regular employee is an employee, and attribute Dept in relation scheme Regular is a foreign key to relation scheme Department.

Employee(Name1, Salary1, Title1, Phone#)	(R_1)
Department(Dname, Manager1)	(R_2)
Regular(Name1, Dept)	(R_3)

Figure 4: Schema of DB1

$\text{Employee}(x, y_1, z_1, w_1) \wedge \text{Employee}(x, y_2, z_2, w_2)$	
$\quad \rightarrow y_1 = y_2 \wedge z_1 = z_2 \wedge w_1 = w_2$	(C_1)
$\text{Department}(x, y_1) \wedge \text{Department}(x, y_2) \rightarrow y_1 = y_2$	(C_2)
$\text{Regular}(x, y_1) \wedge \text{Regular}(x, y_2) \rightarrow y_1 = y_2$	(C_3)
$\text{Regular}(x, y) \rightarrow (\exists z, w, u)\text{Employee}(x, z, w, u)$	(C_4)
$\text{Regular}(x, y) \rightarrow (\exists z)\text{Department}(y, z)$	(C_5)

Figure 5: Constraints of DB1

The schema of DB2 in Figure 6 consists of two relation schemes. Figure 7 shows three constraints in DB2. The first two constraints state that staffs and consultants have unique names. The last constraint states that attribute Manager2 in relation scheme Staff is a foreign key to the same relation scheme.

3.2.2 Mediator's Knowledge

The mediator's knowledge about the relationships between DB1 and DB2 consists of the following. Suppose that salary in DB1 is represented in US dollars, and salary and pay in DB2 are represented in German marks. The exchange rate is represented by a constant r in the

of autonomous heterogeneous databases. As a result, translation becomes straightforward. We will illustrate translation in general by an example in Section 5, after we present the justification for choosing first-order predicate calculus as the mediation language. Formal treatment of the translation is out of the scope of this paper.

3.1 Schemas and Databases

Intuitively a database represents a *perception* (called the *perceived world* [20] or the *model world* [27]) of the real world. Data in a database represents the knowledge of truth values of statements about the real world. A schema specifies the *vocabulary* in which data is expressed, and the *invariant* properties of data. It also supplies a context within which queries could be expressed meaningfully.

Formally, a *dependency* is a sentence in first-order predicate calculus of the form

$$(\forall x_1, \ldots, x_m)(P_1 \wedge \cdots \wedge P_k \rightarrow (\exists y_1, \ldots, y_n)(Q_1 \wedge \cdots \wedge Q_l))$$

where $m, n \geq 0$, P_i is an atomic formula for $1 \leq i \leq k$, and Q_i is either an atomic formula or an equality (when $n = 0$ and $l = 1$) for $1 \leq i \leq l$. A dependency is *equality generating* if $n = 0, l = 1$, and Q_1 is an equality. A dependency is *tuple generating* if Q_i is an atomic formula for $1 \leq i \leq l$ [10].

A *schema* B is a theory (V, A) in first-order predicate calculus, where V is a vocabulary of predicate symbols called *relation schemes*, arguments of relation schemes are called *attributes*, and A is a set of equality-generating or tuple-generating dependencies expressed in V called *integrity constraints*.

A *database* b over B is a structure over V, consisting of a nonempty domain D and an n-ary relation over D assigned to every n-ary predicate symbol in V. It is *valid* if b is a model of B. Given two databases b_1 and b_2 over schema B, $b_1 \subseteq b_2$ is true if, for every predicate $P \in V$, the relation assigned to P by b_1 is contained in the relation assigned to P by b_2. Given two databases b_1 and b_2 over schemas $B_1 = (V_1, A_1)$ and $B_2 = (V_2, A_2)$ respectively where $V_1 \cap V_2 = \emptyset$, $b_1 \cup b_2$ is the database over $(V_1 \cup V_2, A_1 \cup A_2)$ such that $b_1 \cup b_2$ assigns the same value to P as b_i does for every predicate $P \in V_i$, for $i = \{1, 2\}$.

A *conjunctive query* q on B is a conjunction of atomic formulas over V with a (possibly empty) list of free variables. A *query* q on B is a disjunction of conjunctive queries on B. Given a database b over B with domain D and a query q with free variables $x_1, \ldots, x_m$, the answers of q in b are the m-tuples $(v_1, \ldots, v_m)$ in D^m such that q instantiated by $v_1, \ldots, v_m$ is satisfied in b: $b \models q[v_1/x_1, \ldots, v_m/x_m]$.

We consider the *interoperation* of n valid, autonomous, and heterogeneous databases b_i with domains D_i and over schemas $B_i = (V_i, A_i)$ respectively for $1 \leq i \leq n$, where $V_i \cap V_j = \emptyset$ for $1 \leq i \neq j \leq n$. We assume that b_i is empty if the i-th database is virtual. The mediator's knowledge base consists of a theory $B = (\bigcup_{i=1}^{n} V_i \cup V, A)$ in first-order predicate calculus and a structure b over V with domain D, where $V \cap V_i = \emptyset$ for $1 \leq i \leq n$, and A is a set of tuple-generating dependencies. The mediator's knowledge captures the relationships among schemas $B_1, \ldots, B_n$, which specify how data in databases $b_1, \ldots, b_n$ should be related semantically.

3. The mediator's knowledge base is not the schema with which users interact.

4. The knowledge in the mediator's knowledge base is not enforced as constraints cross database boundaries; thus the participating databases do not form a model of it.

Point (2) above shows a big advantage of our architecture over the federation architecture in terms of automation: users or database designers only need to identify, but do not have to resolve, the semantic and representational mismatches in order to access data in multiple databases, thus removing a big hurdle to automation.

Point (3) above shows another important advantage of our architecture over the federation architecture in terms of autonomy: users of a local database access data in multiple databases through the local language and schema instead of a federated schema or a multidatabase language. This is especially appealing for legacy databases: both the data and the applications accessing the data are interoperable.

Point (4) above shows a third advantage of our architecture over the federation architecture: the following difficult issues associated with the federated database approach become nonissues with the query mediation approach.

- *Global consistency*. Consistency is always relative to a view of the world. Since participating databases are not required to form a single logical view, there is no need to enforce global consistency at update time. In other words, updates are not interoperatable.[2]

- *View update*. Since query mediation does not require the establishment of a single logical view over participating databases, updates are not performed through views. Instead, updates are carried out directly in participating databases.

- *Object identity*. Object identifiers are used by an object-oriented database to identify objects in a view of the world. They are essentially LISP Gensyms. Their correspondence to real-world objects is not capturable anywhere within the database, and hence they do not carry any semantics. Since the participating object-oriented databases do not form a single logical view, sharing object identifiers between them is meaningless. In other words, object identifiers are not interoperable.

In general, mediators are knowledge base systems. Since it is unrealistic to expect a single, general-purpose mediator with optimal power [5], multiple mediators should coexist (just like the coexistence of multiple federated schemas in the federated database approach), offering *information communication services* at various levels [19, 33]. These mediators could differ in their tradeoffs between communication cost and capability (bandwidth), and users would *subscribe* to the services that are optimal for their applications.

3 Query Mediation

For easy presentation of query mediation and conflict detection, we choose first-order predicate calculus both as the mediation language of the mediator, and as the representation language

[2]Of course, one database could always monitor changes in other databases, or notify them of its own changes. But participating databases are not required to synchronize with each other.

has, the more data it could help communicate. In other words, a participating database does not have to be completely definable as a view on other participating databases. If a certain part of the database is (directly or indirectly) related to other databases, then accessing that part would lead to accessing data in multiple databases. Otherwise, accessing that part would only result in accessing data in the local database.

We emphasize that our architecture accommodates the federation architecture [30] as a special case. For example, the virtual database on the right in Figure 1 could be considered as a federated schema. If a schema mapping is constructed from the mediator's knowledge base by removing semantic and representational discrepancies, and queries are mediated only in the direction from the federated schema to databases A and B, then we get a federated database in which all queries go through the federated schema in order to access data in both databases A and B, as shown in Figure 3 (arrows represent data flow).

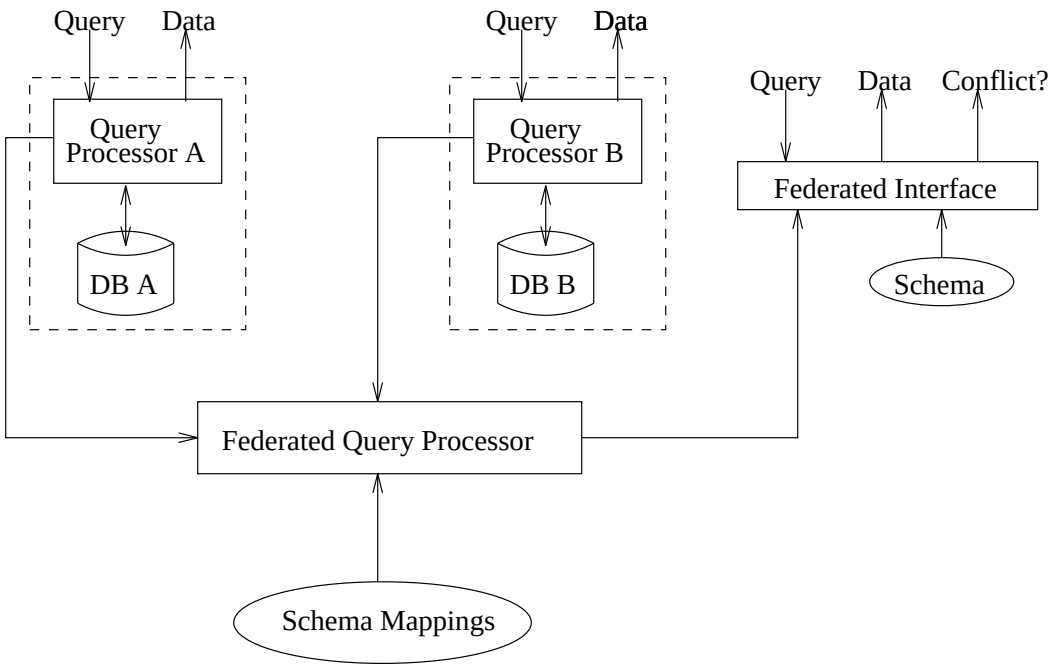

Figure 3: Federation Architecture

We also emphasize that the mediator's knowledge base, together with all the participating schemas, should not be equated to a global or federated schema, for the following reasons.

1. The mediator's knowledge base is at most a very poor schema, because it contains semantic and representational discrepancies and redundancies.

2. The semantic and representational discrepancies and redundancies in the mediator's knowledge base are not removed, since the removal would violate autonomy and would have high complexity.

7. The Mediator derives answer D'_A expressed in the mediation language from answer D'_B based on its knowledge about the relationships between databases A and B, and sends it to Translator A.

8. Translator A translates answer D'_A to answer D_A expressed in the language/schema of database A, and sends it to Wrapper A.

9. Wrapper A merges answers L_A and D_A, detects conflicts in the merged answer, and presents it to users as the answer to query Q_A.

This mediation process is shown in Figure 2.

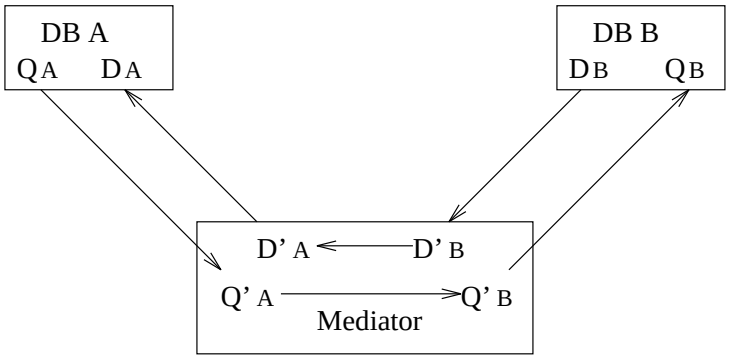

Figure 2: Query Mediation

2.3 Discussion

We could make several important observations of the mediation architecture in Figure 1 and the mediation process of Figure 2.

- **Legacy Databases**. A legacy database (e.g., the dotted boxes in Figure 1) could be made interoperable by wrapping it up with a translator, which makes the database talk in the mediation language, and a conflict detector, which gives users the option of being notified of potential problems in query mediation. The applications that access data in the legacy database become capable of accessing data in multiple databases without having to switch first to a new language or new schema.

- **Virtual Databases**. A participating database could be a virtual one containing only a schema but no data, serving purely as an interface to autonomous heterogeneous databases (e.g., the one on the right in Figure 1). For example, an application designer could define his favorite schema, and specify some relationships of his schema with other participating databases. From then on, users of the application could formulate queries in this schema, and get meaningful access to related data in other databases through query mediation.

- **Incomplete Knowledge Base**. Although we assume that the mediator's knowledge is given, this knowledge does not have to be complete. The more knowledge the mediator

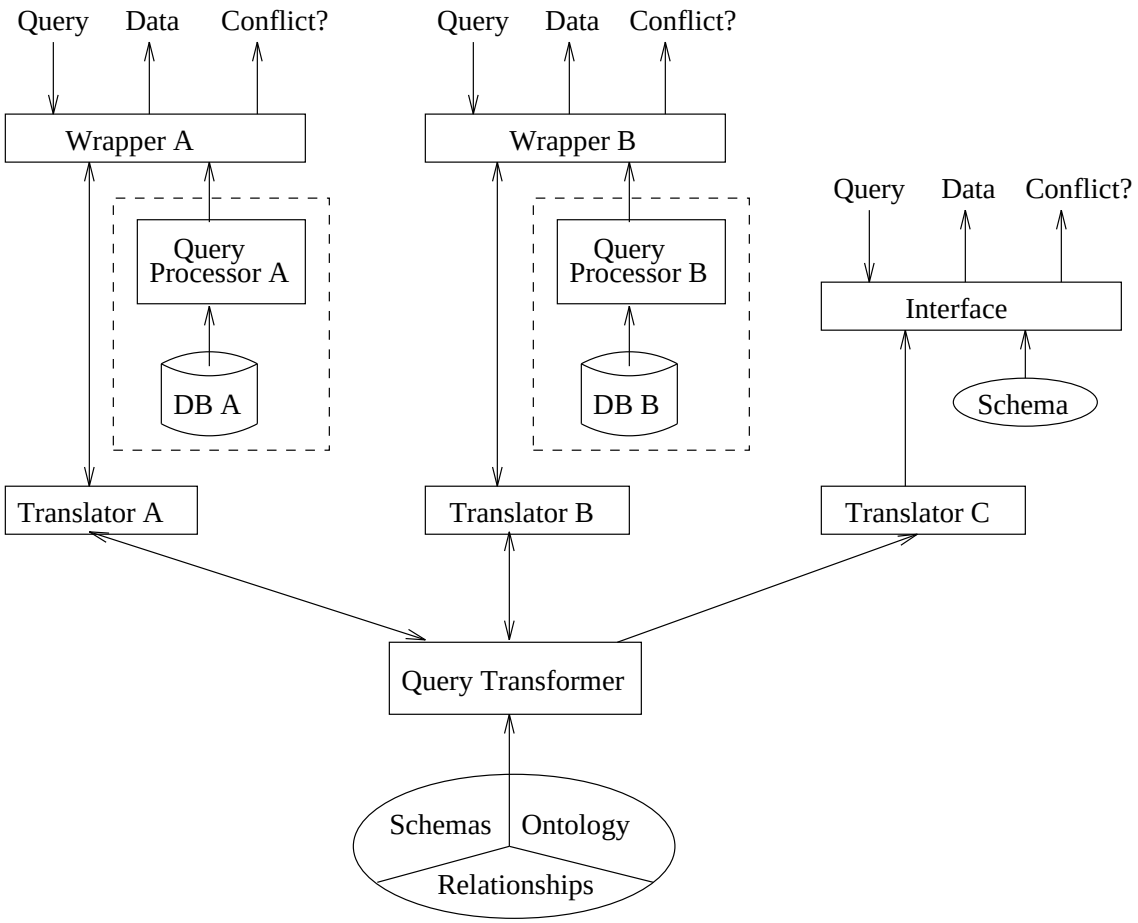

Figure 1: Mediation Architecture

2. Translator A translates query Q_A to query Q'_A expressed in the mediation language, and sends it to the Mediator.

3. From query Q'_A, the Mediator computes a mediated query Q'_B expressed in the mediation language based on its knowledge about the relationships between databases A and B, and sends it to Translator B.

4. Translator B translates query Q'_B to query Q_B expressed in the language/schema of database B, and sends it to Wrapper B.

5. Wrapper B sends query Q_B to Query Processor B to get answer D_B expressed in the language/schema of database B, and sends it to Translator B.

6. Translator B translates answer D_B to answer D'_B expressed in the mediation language, and sends it to the Mediator.

this type of languages differs from the representation languages (i.e., data models) of participating databases. A representation language captures the knowledge about data for the appropriate abstraction and efficient representation of one class of applications, while a mediation language captures the knowledge about data for the meaningful and efficient communication between many classes of applications.

- **Knowledge Base**. Meaningful communication between autonomous heterogeneous databases is based on the relationships between participating databases. These relationships capture the commonalities and mismatches in semantics or representations between these databases. They are expressed in the mediation language and form a *knowledge base*.

- **Mediator**. A mediation language alone is not sufficient to ensure meaningful communication, because autonomous heterogeneous databases might contain data that mismatch in semantics or representations. We need a *mediator* [33] to mediate the communication by resolving potential mismatches. Equipped with the knowledge base of relationships between participating databases, the mediator accepts queries from one database, determines which other databases contain relevant data, generates queries to these databases, and mediates resulting data back to the original database. The mediation is carried out in the mediation language.

- **Translators**. Since the representation languages of participating databases very likely differ from the mediation language of the mediator, we need *translators* to translate queries and data between these representation languages and the mediation language, in order for queries and data to be communicable by the mediator.

- **Conflict Detectors**. When related data from multiple participating databases are merged to give answers to a query, conflicts are always possible because the merged data might be inconsistent with respect to the constraints of the original database in which the query is specified. We need *conflict detectors* to detect such potential problems. The conflict detectors support the communication of globally inconsistent but locally reasonable data [19].

- **Wrappers**. Participating databases are wrapped up by *interface modules* to redirect incoming queries to the mediator, to respond to queries from the mediator, and to merge answers from the mediator.

Figure 1 shows this mediation architecture of interoperation (arrows represent data flow), with three autonomous heterogeneous databases interoperating through a mediator — a *data bus*.

2.2 How Queries are Mediated

Suppose that users issue query Q_A in database A in Figure 1, expressed in the language/schema of database A. The mediation of query Q_A proceeds as follows.

1. Wrapper A intercepts query Q_A, and sends it both to Query Processor A to get answer L_A and to Translator A for mediation.

minimal representational bias. There, the relational model is proposed as such a language, from which object-oriented views are compiled by binding relational data to object templates. The relational model has been used as the mediation language for resolving domain mismatches [8] and as the glue language for interconnecting software components [4]. In [14], first-order logic is recommended as the language for knowledge sharing.

We present a query mediation approach to the interoperation of autonomous heterogeneous databases containing data with semantic and representational mismatches [23]. We develop an architecture of interoperation that facilitates query mediation, and formalize the semantics of query mediation and conflict detection. Queries are mediated between multiple databases, and users of a local database access data in multiple databases using the local language and schema, making both the data and the applications accessing the data in legacy databases interoperable. Queries are automatically mediated, relieving users from the difficult task of resolving semantic and representational mismatches. Semantic heterogeneity is separated from representational heterogeneity by minimizing the representational bias in the mediation language, reducing the space of potential heterogeneity, and improving the efficiency of automated query mediation.

2 Mediation Architecture

The ultimate goal of the interoperation of autonomous heterogeneous databases is to *share* the data stored in these databases. As observed in [19], data sharing does not necessarily mandate the sharing of representations. In fact, since many databases are legacy databases, and today's modern databases will be tomorrow's legacy databases, it is not practical to expect representation sharing. As long as autonomous heterogeneous databases could *communicate* with one another, they could benefit from each other's data without having to be bound to a common representation. In fact, the history of database systems has demonstrated precisely this trend of data sharing with less and less representation sharing: centralized databases mandate physical database sharing, distributed databases mandate logical but not physical database sharing, and federated databases mandate schema but not database sharing.

With the advance in semantic data models and knowledge base systems, data processing has evolved into intelligent information processing, where the availability of semantics and knowledge about data greatly enhances the capability of information abstraction. A similar evolution from data communication into intelligent information communication is essential for the interoperation of autonomous heterogeneous databases. Data should not be communicated as raw bits (i.e., syntactic communication). Instead, they should be *mediated* (i.e., semantic communication) to ensure that data from the sender will be correctly understood for processing by the receiver.

2.1 Components

Our architecture consists of the following components that together support the interoperation of autonomous heterogeneous databases.

- **Mediation Language**. Communication between autonomous heterogeneous databases must be carried out in a *mediation* language or *interlingua*. As pointed out in [19],

1.2 Existing Approaches

The distributed database approach ensures interoperation by forcing users to share a single logical database [6]. Although data might be physically distributed in many databases, conceptually there is only one database, one schema, and one query language — there are no mismatches between the many physical databases either in semantics or in representations. Database autonomy is completely sacrificed with this approach.

The dominating approach to the interoperation of heterogeneous databases has been the federated database approach [1, 30]. With this approach, users of a local database have to switch to a federated schema or a multidatabase language in order to access data in multiple databases, which almost always involve different data models and query languages.[1] In other words, the data in a local database are made interoperable, but the applications in the local database that access the data remain not interoperable, since these applications are coded in the data model and query language of the local database. This is especially impractical for legacy databases because the bulk of the significant investment made by organizations in such databases is in the applications that access the data.

In the federated database approach, either a federated database administrator or a user has to first *identify* the semantic and representational mismatches, and then construct a federated schema to *resolve* these mismatches, before data in multiple databases could be accessed. The construction of the federated schema is essentially a schema integration process [3], which offers little hope for automation [28].

In addition, most researchers advocate the use of a powerful interoperation language in federated databases that could directly express all the representational constructs of heterogeneous databases [7, 2, 15, 17]. Although mapping heterogeneous databases into constructs of the language becomes straightforward, all the semantic and representational mismatches still have to be resolved in the language, which offers little hope for efficiency because of the rich semantics and representations of the language. For example, higher-order logic has to be used in [12] to reason about the equivalence of heterogeneous representations.

In [22], the existence of conflicts in merging data from multiple databases is recognized, and a method is proposed to resolve conflicts when they are detected. However, the semantics of conflicts is not formally defined, and conflict detection is not handled.

The idea of information processing and communication via intelligent mediation is introduced in [33] as a framework of future information systems. Meta-attributes have been used in [26] to specify the contexts associated with attribute values. Relationships between contexts are encoded as conversion rules, and queries are mediated through these relationships to ensure that they are meaningful with respect to their contexts. This is a special case of query mediation, where the mediation is restricted to context matching and value conversion. An example of query mediation from object-oriented databases to relational databases is given in [24], where the schemas and the relationships between schemas are encoded as rules in F-logic.

It is first observed in [32] that data should be shared in some mediation language with

[1] Theoretically it is possible, in the reference architecture of [30], to have external schemas whose data models and query languages are different from a federated schema. However, it remains open whether and how this could be done with the federated database approach. Moreover, having an external schema identical to a local schema would introduce architectural redundancy.

- **Scope**. The set of employees includes retirees as defined by the benefits department, whereas it includes consultants as defined by the payroll department.
- **Granularity**. Gross personal salary is used for job survey, whereas adjusted family income is used for taxation.
- **Temporal Basis**. Branch offices are concerned with weekly sales, whereas the central office is more interested in monthly revenue.

Direct comparison and combination of data with such semantic mismatches would be meaningless. In addition to semantic mismatches, the same data could be represented in various incompatible structures, and the same structure could be used to represent data with incompatible semantics [29]. The representational differences are caused by the need to bind data to representations that are most natural and efficient with respect to specific applications. In general, there simply does not exist a universal representation that is perfect for every application [5, 14, 19]. Examples of *representational mismatches* are:

- **Identification**. Employees could be identified by employee ID numbers in the personnel department, but by social security numbers in the payroll department. The nature of operations in these two departments demands that different identifiers be used, since employee data is most likely accessed by social security numbers, not employee ID numbers, in the payroll department.
- **Type Conflict**. Marriage is considered by the Internal Revenue Service as a one-to-one relationship between men and women for current marriages, but by the Census Bureau as a many-to-many relationship between men and women for marriage history. It would complicate the operation of the IRS if current marriage had to be represented as a many-to-many relationship.
- **Biased View**. The one-to-one marriage relationship between men and women could be represented as a binary predicate, a binary boolean function, a wife attribute attached to man objects, or a husband attribute attached to woman objects. It is impractical to represent the relationship in all possible structures.

Because of the diverse needs of autonomous organizations, heterogeneity will persist rather than disappear. To support the interoperation of autonomous heterogeneous databases containing data with semantic and representational mismatches, three critical issues have to be addressed:

- **Autonomy**. Database autonomy should be respected and preserved. Users should not be required to switch to new query languages or new schemas in order to access data in multiple databases.
- **Automation**. Interoperation should be automated. Users should not be required to manually resolve all the semantic and representational mismatches in order to access data in multiple databases.
- **Efficiency**. Automated interoperation should be computationally efficient. In particular, it should not require expensive mechanisms such as theorem-proving in higher-order logics.

Semantic Interoperation: A Query Mediation Approach*

Xiaolei Qian and Teresa F. Lunt
Computer Science Laboratory, SRI International
333 Ravenswood Avenue, Menlo Park, CA 94025

Abstract

We present a query mediation approach to the interoperation of autonomous heterogeneous databases containing data with semantic and representational mismatches. We develop an architecture of interoperation that facilitates query mediation, and formalize the semantics of query mediation. The main contributions are the automated mediation of queries between databases, and the separation of semantic heterogeneity from representational heterogeneity. Query mediation in heterogeneous legacy databases makes both the data and the applications accessing the data interoperable. Automated query mediation relieves users from the difficult task of resolving semantic and representational mismatches. Decoupling semantic and representational heterogeneity improves the efficiency of automated query mediation.

1 Introduction

The interoperation of heterogeneous databases is a pressing need today as organizations attempt to share data stored in legacy databases. These databases are independently developed and maintained to each serve the needs of a single organization. The exchange of data between such databases could be problematic not only because of differences in the representation (syntax) of data but also due to often subtle differences in the intended interpretation (semantics) of data. Thus, although translators could be constructed to reformat data from one representation to another, such a translation does not guarantee that the combined, translated data are meaningful — we could be attempting to compare apples with oranges.

1.1 Problems and Issues

Heterogeneity in the semantics of data arises naturally. The semantic differences are caused by the diverse needs of applications. Moreover, the relationships between heterogeneous data could be incomplete or uncertain. Examples of *semantic mismatches* are:

*This work was supported in part by U.S. Department of Defense Advanced Research Projects Agency and U.S. Air Force Rome Laboratory under contract F30602-92-C-0140.