Towards Efficient and Scalable Mediation: The AURORA Approach

Ling Ling Yan
Laboratory for Database Systems Research
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
ling@cs.ualberta.ca

Abstract

We develop a 2-tier, plug-and-play mediation model for accessing a large number of heterogeneous data sources. This model defines a divide-and-conquer approach towards information integration. It is more suitable for applications such as electronic commerce than existing models. We also develop algebras that manipulate heterogeneous data, the *mediation enabling algebras*, that provide new techniques for efficient query processing in large-scale middleware. This paper presents the mediation model, architecture and techniques studied in the AURORA project.

1 Introduction

Today, a vast amount of digital information is provided by sources ranging from database systems, WWW pages, file systems and spreadsheets, to special-purpose data repositories. With the advent of the Internet, the way people use information is changing rapidly. In general, software that facilitates access to multiple heterogeneous information sources is needed. Such software is commonly known as a *middleware*.

Accessing heterogeneous and autonomous data sources is a complicated process due to the heterogeneities among the sources at levels ranging from platform to semantics [KS91]; this complexity grows rapidly as the number of sources increases. Middleware provides one-stop data access by managing the complexity on the application's behalf; it performs *mediation* between applications and diverse data sources.

The goal of the AURORA project is to develop a collection of mediators that can be used for constructing middleware which provides integrated (read-only) access to heterogeneous and autonomous data sources. This middleware must be:

1. *Scalable.* Adding or removing a data source from the access scope is easy.

2. *Efficient.* When processing a query, redundant data retrieval is minimized.

Research in AURORA has two main themes: a 2-tier mediation model and techniques for efficient query processing.

1.1 2-tier Mediation Model

AURORA models mediation as a 2-step process: homogenization followed by integration, each performed by respective mediators. Homogenization removes idiosyncrasies of a source in structure and semantics; it involves only

one data source; multiple sources can be homogenized independently and in parallel. After being homogenized, a data source can be "plugged" into an integration mediator and removed by "unplugging" it. Since all the sources are homogenized, the procedures of plugging and unplugging handle a small range of heterogeneities and require minimum expertise. Homogenization is a more difficult task and is assisted by AURORA tools.

AURORA's mediation model defines a divide-and-conquer approach towards information integration. It facilitates scalable data management in large scale applications such as electronic commerce. It also enables us to decompose the highly complex technical issues in large-scale middleware, such as query optimization, into more manageable, simpler problems. Although many previous mediation models exist, none provides the above facilities that are necessary in applications like electronic commerce. Many technical issues in large-scale middleware are known to be difficult (ref. Sections 4.1.3, 4.1.4).

1.2 Efficient Query Processing

Although much is known about identifying and resolving semantic heterogeneities [KS91, MIR93], the impact of this process on query processing efficiency is seldom discussed. We intend to address this missing link in AURORA. AURORA schema integration constructs are tightly tied with algebras suitable for query optimization. These algebras, called *Mediation Enabling Algebras* (MEAs), provide operations specifically designed for manipulating heterogeneous, distributed and autonomous data. With MEAs, we identify the impact of mediation and take it into account in query processing.

Mediator views as well as mediator queries are expressed in MEAs. The view expressions are used to modify a mediator view query into an expression that is then manipulated by an algebraic query optimizer in order to achieve optimal query performance. The following techniques are studied:

1. MEAs for AURORA mediators.
2. Algebraic query rewriting and optimization in each MEA.
3. Selective materialization of data to enable intelligent query decomposition.

1.3 Contributions

First, we develop a 2-tier, plug-and-play mediation model that defines a divide-and-conquer approach towards information integration; it is more suitable for applications such as electronic commerce than other models. Second, we develop architecture and enabling techniques for implementing this model. Especially, we define the concept of MEA that provides new techniques in query processing in large-scale middleware.

The rest of this paper is organized as follows. Section 2 describes an electronic commerce application. Section 3 reviews existing technologies. Section 4 gives an overview of the AURORA project. Section 5 presents enabling techniques in AURORA. Section 6 contains conclusions and future work.

2 Electronic Commerce

A virtual shopping mall is a typical electronic commerce (EC) application. A key component in this application is the catalog system. Companies organize their catalogs differently. This gives rise to a set of heterogeneous and autonomous catalogs. When the number of participating catalogs is large, it is difficult for a shopper to locate items of interest. One approach is to require all vendors to re-organize

their catalogs in a common format and merge all the catalogs into a central catalog database which allows customers to perform sophisticated searching without dealing with individual catalogs. This requires re-engineering of existing catalogs. In general, vendors want to participate in the central catalog without making changes to their existing catalogs. We study a *virtual catalog* that has the look and feel of a central catalog but holds no physical data. Upon a customer request, this catalog retrieves relevant information from (multiple) individual catalogs and assembles an answer. Such a virtual catalog should satisfy the following requirements: (1) it is up-to-date but does not violate the autonomy of the participating catalogs; (2) its search performance does not degrade as the number of participating catalogs increases; (3) it allows easy inclusion of new catalogs and integrates with other EC applications; and (4) it is easy to construct. Tools should be provided to assist in construction.

3 Existing Technologies

A few paradigms exist for facilitating integrated access to heterogeneous and autonomous data sources, notably the federated database systems (FDBs) [SL90], and the mediator systems [Wie92]. More recent approaches handle dynamic sources and sources with diverse query processing capabilities.

3.1 Federated Database Systems

Federated databases (FDBs) [Chu90, LR82, THMB95, Ahm91, ACHK93, CHS91, KS91, Car95] support variations of a five-level extended schema architecture as shown in Figure 1. A *local schema* is the conceptual schema of a component database; it is expressed in the data model of the component database. Different local schemas may be expressed in different data models. A *component schema* is derived by translating a local schema into a *canonical data model*. An *export schema* is a subset of a component schema that is made available to the federation. A *federated schema* is an integration of multiple export schemas. The *external schemas* are the views exposed to applications and end users.

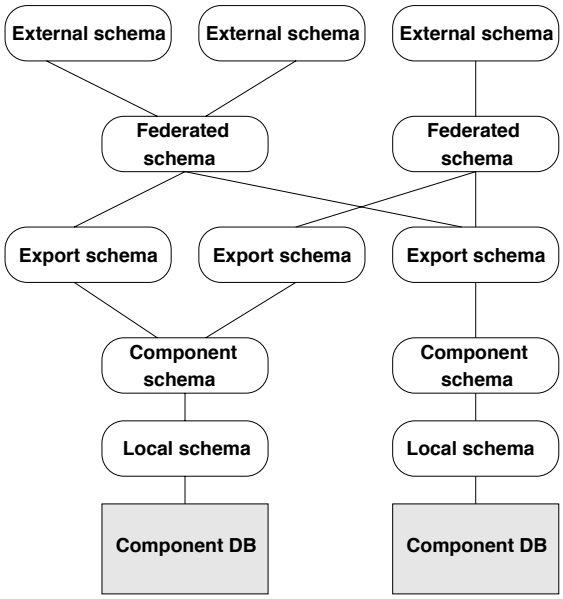


Figure 1: FDB Schema Architecture

There are two types of FDBs: the *tightly-coupled* and the *loosely-coupled*. Tightly-coupled systems require the construction of a *global schema* via which queries can be posed. Semantic heterogeneities among participating databases are resolved at the global scheme level and are hidden from users querying this schema. In loosely-coupled systems, there is no global schema. The end-user sees a loose collection of possibly inconsistent schemas that are represented uniformly in a *canonical data model*. The end-user queries these schemas using a *multidatabase query language*, such as MSQL [Lit90]. Since the schemas are not integrated, heterogeneities among them must be resolved by the end-user at query time. The dis-

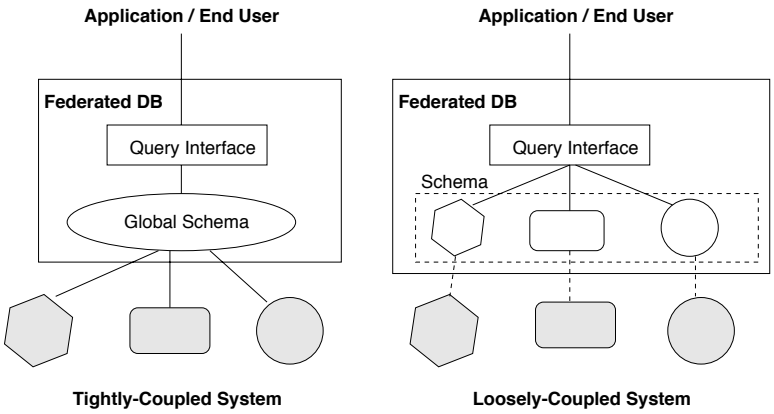


Figure 2: The Federated Database Systems

tinction between tightly-coupled and loosely-coupled systems is blurred in systems such as Pegasus [Ahm91] and UniSQL/M [KS91]. In these systems, users perform integration to various degrees and resolve other semantic discrepancies at query time.

3.2 Mediator Systems

A mediator is a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications [Wie92]. A mediator system has the form illustrated in Figure 3. A *wrapper* is a special mediator that handles idiosyncrasies of individual data sources.

Today, many mediator systems are being built [PGMW95, PGGMU95, PGMU96, PAGM96, Sub, QL94, RCD94, FRV95, FRV96, SSR94, GMS95]. Unlike federated databases that often aim at creating a single database illusion, mediator systems offer specific information services. Such services can be built based on existing services provided by other mediators. Thus, the mediator system is not monolithic but is a network of mediators, each accessing multiple heterogeneous data sources and/or other mediators. Compared with FDBs, the mediator systems put

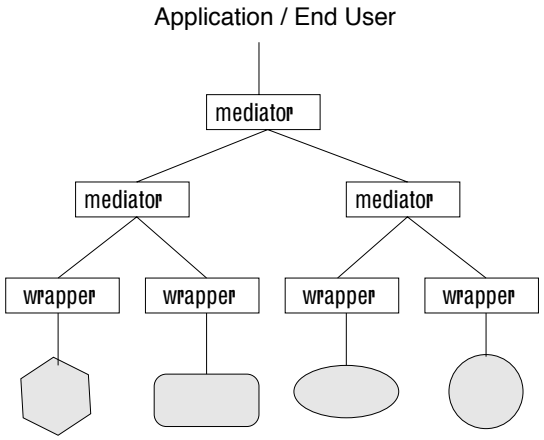


Figure 3: A Mediator System

more emphasis on flexibility, versatility, and knowledge-based capabilities. Domain specific mediators (the so-called *vertical mediators*) often exploit domain knowledge to enhance usability and performance.

3.3 Accessing Dynamic Sources

With the Internet and WWW, more data sources are open with diverse availability and query processing capabilities. A few approaches deal with this situation [LRO96b, TRV96, LPL96].

DISCO [TRV96] extends the ODMG ODL to allow a bag of extents for a single inter-

face type. Adding a new source means adding an extent into this bag. DIOM [LPL96] intends to identify relevant data sources and bind to them at runtime. The Information Manifold project [LRO96b, Lev96, LRO96a] considers large number of data sources with varying query capabilities. It assumes the existence of a *worldview*, a pre-defined common application view. A source participating in the worldview is regarded as a *materialized view* of the worldview, with *capability records* attached describing what type of queries it can handle. Adding a new source only means adding a new materialized view. The problem of answering a query against the worldview is transformed to that of answering a query with existing materialized views, with additional constraints. This problem is solved in [LRO96b]. Handling sources with limited query capabilities is an especially useful technique for accessing sources such as those on the Web.

4 The AURORA Project

The AURORA project builds mediators that can be used for constructing middleware.

4.1 AURORA Mediation Model

A *mediation model* describes how heterogeneities among data sources are perceived and handled. The AURORA mediation model, shown in Figure 4, is a unique 2-tier model. It models mediation as a 2-step process: *homogenization* followed by *integration*, performed by respective, specialized mediators.

4.1.1 A 2-tier Mediation Model

We distinguish between two categories of heterogeneities among data sources: *schematic mismatches* that arise when the same application domain is modeled differently; and *instance level conflicts* that arise when inconsistent values on the same real world object are recorded. Schematic mismatches must be resolved before instance level conflicts are tackled. The process of resolving schematic mismatches is referred to as *homogenization*. In AURORA, specialized mediators, the *homogenization mediators*, support this process. The result of homogenizing a source is a *homogenizing view* that hides the deviations of this source from the target application view in both structure and semantics. The *integration mediator* "glues" a large number of data sources together by combining all the homogenizing views into an integrated view. Since all the sources are homogenized, the integration process is greatly simplified; it only considers instance level conflicts; schematic heterogeneities have been largely removed by homogenization.

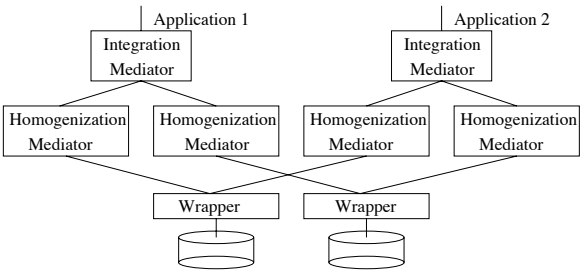


Figure 4: The AURORA Mediation Model

4.1.2 2-tier Model and Scalability

To include a new data source into the access scope, one must resolve two issues:

1. *Communication.* It must be possible to "talk" to the data source. This is achieved by a wrapper that removes idiosyncrasies of the data source in communication protocol, data model, and query language.

2. *Semantic integration.* It must be possible to construct an integrated view that protect the end-users from the scale and di-

versity of the access scope.

Scalability requires that both steps be performed rapidly. For 1, this requires rapid, if not automatic, wrapper generation. Various enabling techniques have already been developed [PGGMU95]. Much is known about *how* 2 can be done, the scalability aspect of it has not been well addressed (ref. Section 4.1.3).

The scalability issue in AURORA is reduced to the issues of rapid homogenization of individual sources and the rapid integration of multiple sources. Homogenization can be performed in parallel among sources. It concerns single data sources and has good potential to be fast. AURORA provides tools to assist in this process. Integration handles multiple but homogeneous (although still autonomous) sources. To achieve scalability of this step, AURORA integration mediator assumes that the integrated view is pre-defined by application requirements. Let this view be V_W. To integrate objects in each data source (it now appears as a homogenization mediator) into V_W, AURORA requires these objects to be *registered* as *fragments* of objects defined in V_W. Removing a source from the access scope only requires the relevant registrations to be cancelled. This way the integration is performed in a plug-and-play fashion.

With many mediation models and architectures already developed, one asks the question: *"does the world need another one?"*. In the next two sections, we answer this question.

4.1.3 Related Mediation Models

With respect to scalability, we distinguish between two types of mediation models: *derivation-based* and *plug-and-play*.

In derivation-based models, a mediator view is *derived* from a set of source descriptions, usually via a monolithic view specification. When sources participate or withdraw from the scope of the mediator, this specification must be modified. This modification may affect many parts of the specification, in the worst case, the specification has to be reconstructed. When the number of participating sources are large, this specification becomes large and complex, difficult to maintain. Most FDBs and mediator systems uses this model.

In plug-and-play models, a common application view is pre-defined. Data sources contribute to and withdraw from this view without affecting other participating sources or the applications that access this view. To our knowledge, two systems use this model: DISCO [TRV96] and Information Manifold (IM) [LRO96b].

Apparently, derivation-based models do not favor scalability since including or excluding a source is difficult when the number of sources involved is large. The plug-and-play model avoids this problem. AURORA's mediation model is indeed a plug-and-play model; like those used by DISCO and IM. However, the AURORA model is 2-tier, using two types of mediators, each handling a specific range of heterogeneities, while both DISCO and IM models are 1-tier, using a single mediator to handle the whole range of heterogeneities considered. IM assumes that wrappers handle some mismatches but it is not clear what these mismatches are; IM concentrates on query plan generation. Both DISCO and IM consider a limited range of mismatches; none of them considers instance level conflicts.

4.1.4 Why 2-tier?

A 2-tier model defines a divide-and-conquer approach to information integration. Such an approach facilitates applications such as electronic commerce that require access to large numbers of diverse data sources. It also allows us to better manage the technical complexity

in building large-scale middleware.

2-tier Mediation in EC

Typically, to include a supplier catalog into a virtual catalog, the supplier is first required to map his or her catalog into a format required by the virtual catalog. Essentially, the supplier catalog must be *homogenized* before participating in the virtual catalog. Homogenization is performed by suppliers independently referencing the common catalog format. Individual suppliers are not concerned with inter-catalog conflicts which are resolved at the central catalog level. Often, suppliers are provided with a *workbench* to perform homogenization. This workbench is indeed a homogenization mediator. The central catalog is an integration mediator. A supplier can participate in multiple virtual catalogs requiring varying catalog formats. In this case, the supplier must use multiple homogenization mediators.

Obviously, AURORA's 2-tier model models the activity of constructing virtual catalogs closely. For DISCO or IM to be used in the above scenario, some refinement to their mediation models must be done to clearly define which mismatches are to be resolved by the suppliers independently and which are to be handled at the central catalog level. In general, the 1-tier mediation models do not support application scenarios like the one described above.

Managing Complexities

Integrated access to a large number of highly heterogeneous data sources is complicated. There are two aspects to this complexity: integration and query processing.

Complexity in integration. When there are 100 sources involving many types of mismatches and conflicts, which one do we resolve first? Can multiple people work on a single integration task? None of the previous models address these issues.

Complexity in query processing. When a large number of highly heterogeneous sources are involved in a query, we face a complex optimization problem that is unknown to traditional data management systems. Query optimization in middleware systems is known as a difficult problem even without considering the scale of the system [LOG92]. In large scale middleware systems such as virtual catalogs in EC, this problem is even more difficult [Yan97].

AURORA's 2-tier model manages both complexities. To manage the complexity of integration, AURORA divides the integration task into two well-defined, smaller tasks, homogenization and integration, that are to be performed by multiple people assisted by specialized tools mandating specific mediation methodologies. In query processing, AURORA's 2-tier mediation model enables us to decompose the query processing issue into two smaller problems: query processing in homogenization mediators and that in integration mediators. It is our intension to take advantage of this simplification to develop new techniques in mediator query processing. As shown later, each type of mediator uses a specialized Mediation Enabling Algebra (MEA) to facilitate efficient query processing.

4.2 AURORA Mediators as Distributed Components

AURORA does not restrict the canonical data model to a specific one that is deemed to be most "suitable". Rather, AURORA provides mediators and wrappers in two popular data models, the relational data model and the ODMG object data model. Necessary facilities are provided to allow the two data models to coexist seamlessly. AURORA mediators are classified along two dimensions: the canonical data model and the mediator type, homoge-

nization or integration. Figure 5 shows this classification.

Mediator Type \ Canonical Data Model	Relational	Object-Oriented
Homogenization	AURORA-RH	AURORA-OH
Integration	AURORA-RI	AURORA-OI

Figure 5: AURORA Mediator Classification

We design and develop three types of software components: the wrappers, the homogenization mediators and the integration mediators. These are provided as distributed components that communicate and cooperate via an Object Request Broker (ORB), as shown in Figure 6. AURORA wrappers and mediators support pre-defined interfaces, as shown in Figure 7. These are the only interfaces via which a wrapper/mediator can be accessed by the application or by other AURORA mediators.

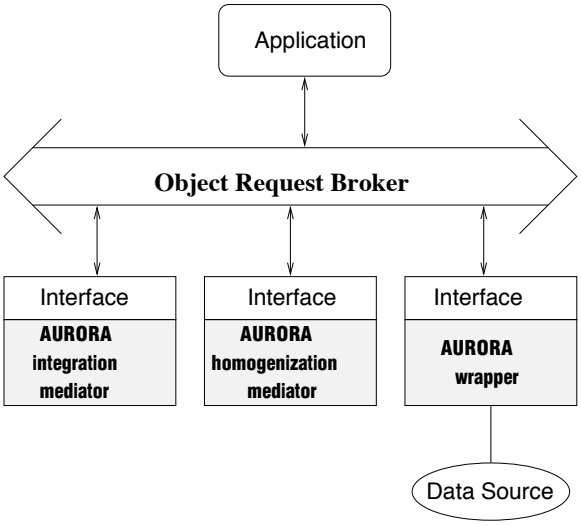
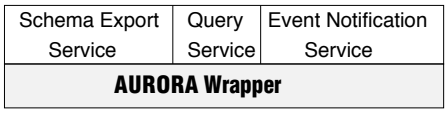


Figure 6: AURORA Components and ORB

A middleware that facilitates integrated access to multiple heterogeneous data sources can be constructed by using a network of mediators that cooperate with one another to provide



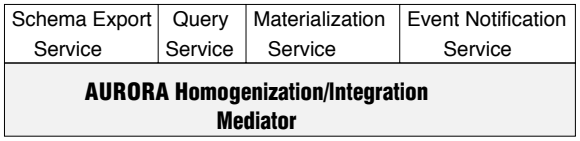


Figure 7: The AURORA Mediator Interfaces

an integrated data service. With AURORA, one can choose between relational and object-oriented components. The use of AURORA mediators in building middleware is best illustrated by Figures 8 and 9. AURORA-O mediators have the built-in capability of accessing AURORA-R components, but not vice versa.

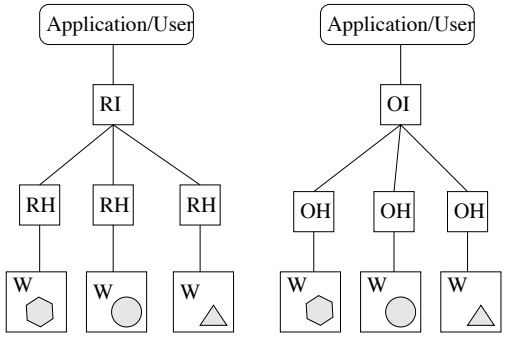


Figure 8: AURORA Application: uniform

4.3 AURORA Mediator Development Workbench

Each AURORA mediator is a *mediator development workbench* consisting of a mediator skeleton and a toolkit named MAT.

Mediator Skeletons. The most important components in a mediator are an integrated view over (multiple) data sources and a query processor that answers queries posed against this view. Building a mediator means building the view, the query processor, and software

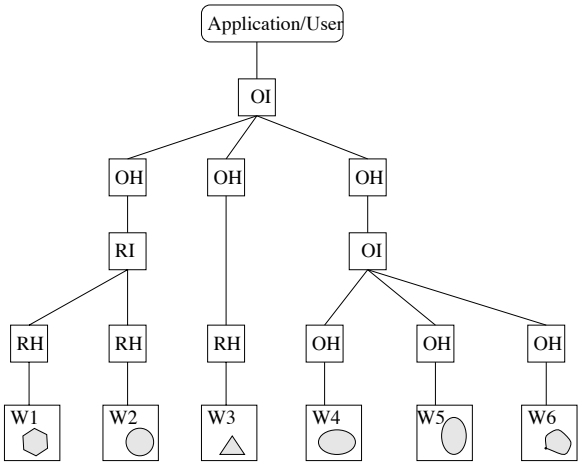


Figure 9: AURORA Application: mixed

modules that support other standard services. In AURORA, mediators are constructed from *mediator skeletons* that have all the necessary components of a mediator except for the view.

Mediator Author's Toolkits (MATs). In AURORA, a *mediator author* chooses a mediator skeleton, identifies heterogeneities among the sources, and defines views into the mediator skeleton to resolve the heterogeneities. AURORA MATs assist the mediator authors in performing such tasks. This scenario is shown in Figure 10. A MAT has two main functionalities: it mandates a *mediation methodology* and it provides *Mediation Enabling Operator*s (MEOs).

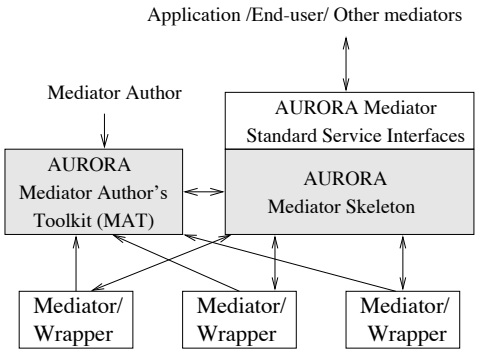


Figure 10: An AURORA Workbench

4.4 Constructing Virtual Catalogs with AURORA

Now we go back to our example of virtual catalogs described in Section 2 and see how the AURORA approach could be used. Consider a number of vendors each having an autonomous catalog database. A virtual catalog effort can be initiated by a third-party broker who seeks to offer value-added services. The broker first designs a common catalog structure, its data model and query language. To include a vendor into the virtual catalog, the broker first homogenizes the vendor's catalog using an AURORA homogenization mediator. This process maps the vendor catalog structure and semantics into those in the common catalog. After homogenization, it should be straightforward to "plug" a catalog into an AURORA integration mediator that supports the common catalog. While homogenization is a more complex process, the broker can hire a few people to homogenize individual vendor catalogs in parallel. An integration mediator is where large number of virtual catalogs merge but the integration is a simple mechanism. Overall, construction of the virtual catalog is scalable.

5 Enabling Techniques

Each AURORA component requires a suite of enabling techniques. The basis of such a suite is a *Mediation Enabling Algebra*, a MEA, that provides *Mediation Enabling Operators*, MEOs, that are specially designed for manipulation of heterogeneous and autonomous data. Different mediators require different MEAs. Once a MEA is designed, development of the rest of the suite involves the following:

- Design of a MAT. This involves the design of a mediation methodology and the supporting tools. Each tool offers a subset of

the MEOs in the MEA.

- Design of a skeleton. This involves (1) developing a query rewriting algorithm in the MEA; (2) developing query transformation rules in the MEA that potentially allow query optimization; (3) design of a query optimization strategy; and (4) developing techniques for evaluating expensive MEOs efficiently.

A specific suite for AURORA-RH mediator has already developed. It has been described in detail in [YOL97a] and is reviewed briefly in Section 5.1. This suite indeed sets the paradigm of all suites. Other suites can be developed in a similar fashion. Work in integration mediator suites are in early stage. We give an overview of these suites in Section 5.2.

5.1 Homogenization Mediators

The architecture of AURORA-RH is shown in Figure 11. **MAT-RH** is a toolkit that assists a mediator author in constructing a homogenizing view. It provides a set of MEOs and mandates a homogenization methodology. Each tool in the toolkit allows specification of *transformations* and *domain mappings*. Transformations are expressions consisting of the AURORA-RH MEOs and the usual relational operators. Domain mappings are arbitrary mappings. This information is captured in the **View Definition Repository** and are used for query processing. **AURORA-RH Primitives** are MEOs; they extend the relational algebra to form a *Mediation Enabling Algebra* (MEA), MEA-RH. **AURORA-RH Query Processor (AQP)** processes mediator queries posed against the target database. It rewrites such a query into a (optimal) set of queries over the source database, sends these queries for execution and assembles the answer to the mediator query from the returned data.

5.1.1 MEA-RH

The MEA based on which AURORA-RH is built is MEA-RH. MEA-RH extends the usual relational algebra with the following MEOs:

$R' = retrieve(Q)$. Submits query Q for evaluation by the source and return the result.

$R' = pad(R, A, c)$. "Pad" each tuple in relation R with a new attribute A valued c.

$R' = rename(R, A, n)$. Rename attribute A in R to a new name n.

$R' = deriveAttr(R, L_1, N_1, f_1, ..., L_k, N_k, f_k)$. From each tuple in $t \in R$, *deriveAttr* derives a new tuple t' such that $t'[N_i] = f_i(t[L_i])$.

5.1.2 MAT-RH

MAT-RH identifies a wide range of domain and schema mismatches and mandates a methodology to resolve them systematically. It also provides constructs for expressing such resolutions. Let B be a source and M be the target view. MAT-RH identifies the following types of mismatches:

Cross-over schema mismatches. A **type 1** cross-over mismatch happens when a concept is represented as data in M but as relations in B. A **type 2** cross-over mismatch happens when a concept is represented as data in M but as attributes in B.

Domain structural mismatches. This mismatch happens when a domain in M corresponds to a domain with a different data type or several data domains in B.

Domain unit mismatches. This mismatch happens when a domain in M assumes different units of measurement from the corresponding domain(s) in B.

Domain population mismatches. This mismatch happens when a domain in M assumes different population from the corresponding domain(s) in B.

MAT-RH mandates a 6-step methodology for homogenization. It supports each step by a

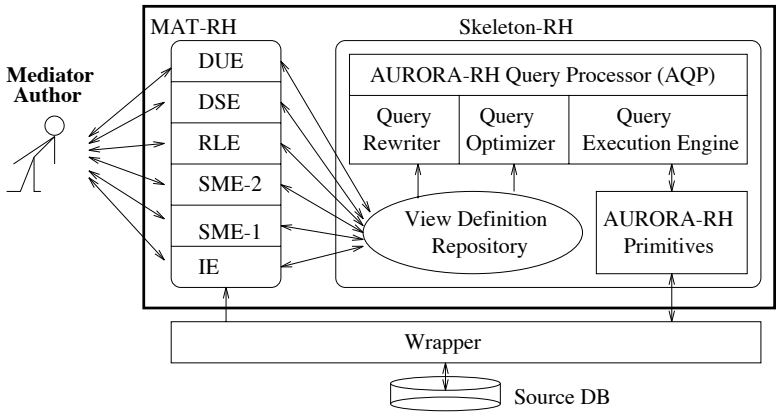


Figure 11: AURORA-RH Workbench

specialized tool (environment) that provide a set of MEOs and allows certain types of transformations and domain mappings to be specified. This is shown in Figure 11 and is further described below:

step 1. Construct an import schema. The supporting environment is the Import Environment (IE). MEOs provided by IE are the usual relational operators.

step 2. Resolve type 1 schema mismatches. The supporting environment is the Schema Mismatch Environment 1 (SME-1). SME-1 provides a special transformation, the *RELmat*, that is defined as following:

> Given $D^R = \{R_1, ..., R_n\}$, a group of relations with identical schemes, let A be an attribute, $A \notin ATTR(R_1)$:
> $RELmat(D^R, A) =$
> $\bigcup_{i=1}^{n} pad(R_i, A, RELname(R_i))$

Its meaning is illustrated in Figure 12.

step 3. Resolve type 2 schema mismatches. The supporting environment is the Schema Mismatch Environment 2 (SME-2). SME-2 provides a special transformation, the *ATTRmat*, that is defined as following:

> Given $D^A = \{A_1, ..., A_n\}$, a group of attributes in a relation S that have identical data types, let N_A and N_V be attributes, $N_A, N_V \notin ATTR(S)$, then:
> $ATTRmat(S, D^A, N_A, N_V) =$
> $\bigcup_{i=1}^{n} pad(rename(\pi_{ATTR(S)-D^A \cup \{A_i\}}(S),$
> $A_i, ATTRname(N_V)),$
> $N_A, ATTRname(A_i))$

Its meaning is illustrated in Figure 13.

Sales

month	*ibm_pc*	*mac*	*laptop*
Feb/96	*6700*	*6900*	*8000*
Mar/96	*7600*	*8400*	*7800*

D^A = {ibm_pc, mac, laptop}

ATTRmat(Sales, D^A, product_type, salesAmt)

month	*salesAmt*	*product_type*
Feb/96	*6700*	*ibm_pc*
Mar/96	*7600*	*ibm_pc*
Feb/96	*6900*	*mac*
Mar/96	*8400*	*mac*
Feb/96	*8000*	*laptop*
Mar/96	*7800*	*laptop*

Figure 13: The *ATTRmat* operator

step 4. Link relations. The supporting environment is the Relation Linking Environment (RLE). RLE mandates that for each relation in the homogenizing view, a prototype relation must be specified. This relation is close to the target relation modulo domain mismatches. The MEOs provided by RLE include *rename* and the usual relational operators.

step 5. Resolve domain structural mismatches. The supporting environment is the Domain Structural Environment (DSE). DSE allows specification of *domain structural functions*; these can be user-defined functions or stored mapping tables.

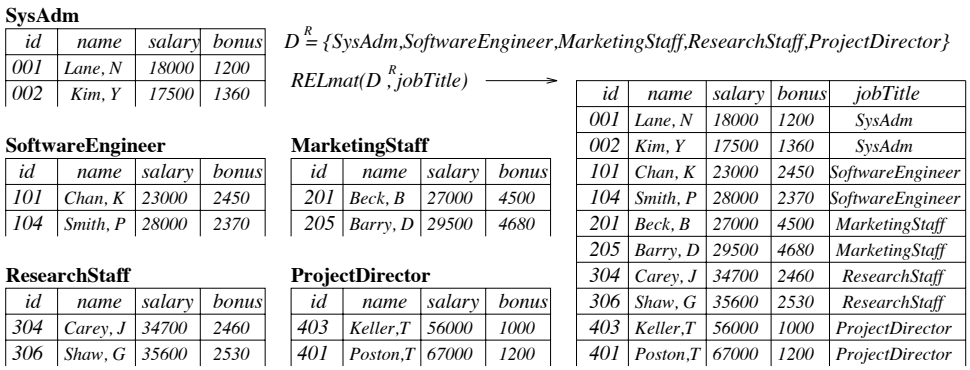


Figure 12: The *RELmat* operator

step 6. Resolve domain unit/population mismatches. The supporting environment is the Domain Unit Environment (DUE). DUE allows specification of *domain value functions*; these can be user-defined functions or stored mapping tables.

Detailed description of these environments and examples of their application are given in [YOL97a].

5.1.3 AURORA-RH Query Processor

Query processing in AURORA-RH is based on MEA-RH. Homogenizing views are defined using these operators via MAT-RH. A mediator query is processed in three steps. First, we use the view definitions to rewrite the query into an algebraic formula in MEA-RH. A query rewriting algorithm has been developed and is given in [YOL97a]. Second, we transform this formula into an "optimal" form using transformation rules. A complete set of transformation rules are given in [YOL97a]. These rules allow exchanging the usual relational operators with MEA-RH operators. Query optimization in AURORA aims at *maximizing* queries sent to the source so as to leverage the query optimization capabilities of the source. Third, we *assemble* query results returned from the data source to produce an answer to the mediator query. This assembly uses MEA-RH operators. A complete walk-through of query processing is given in [YOL97a] by an example.

5.2 Integration Mediators

AURORA integration mediators, AURORA-RI and AURORA-OI (Figure 5), are responsible for integrating a large number of *homogenized* sources. Since the sources are homogenized, the types of heterogeneities that the integration mediator must handle is limited. First, lets give a closer look at the 2-tier mediation model and the meaning of homogenization.

5.2.1 Plug-and-Play Integration

Lets assume that the application view consists of a single relation R_g. A data source is said to be *homogenized in regard to* R_g if:

1. It is structurally homogenized. It contains a single relation R_s that is a *fragment* of R_g, that is, $ATTR(R_s) \subseteq ATTR(R_g)$.

2. It is semantically homogenized. Each attribute in R_s is the *same* as that in R_g with the same attribute name.

Now the following plug-and-play integration mechanism can be imagined: To "plug" a data source into the the integration mediator, we

first homogenize this data source in regard to the application view, that is, construct a *homogenizing view* on top of the data source. We then *register* each relation in this view with the integration mediator as a fragment of a global relation. To "unplug" a data source from the integration mediator, we remove all fragments from this source. When a particular source is down, all fragments from this source are considered to be empty.

At query time, the integration mediators derive the population of global relations and resolve instance level conflicts. We study the following techniques:

1. MEAs for integration. This involves designing new MEOs specially for integration. It also involves development of query modification algorithms and query transformation rules.

2. Efficient evaluation of expensive MEOs in such MEAs. In particular, we explore selectively materializing data to facilitate intelligent query decomposition and join order selection. An initial study along this direction has been done under the name of *Mediator Join Index (MJI)* [YOL97b].

6 Conclusions and Future Work

We have described AURORA, a project that develops techniques for building efficient and scalable mediation. Our contributions are the following. First, we have defined a 2-tier, plug-and-play mediation model (Figure 4). Technically, this model enables us to take a divide-and-conquer approach towards building integrated access to heterogeneous sources. Mediation is divided into 2 steps, homogenization followed by integration, and respective mediation methodologies are provided for each step. The general mediator query processing techniques are also divided into two categories: those in homogenization mediators and those in integration mediators. AURORA develops specialized mediation enabling algebras (MEAs) for each category. Second, we have described a complete suite of techniques used by a specific AURORA mediator, AURORA-RH. With this, we have explored the feasibility of our paradigm and are ready to develop technique suites for other AURORA mediators.

Research wise, our goal is to design a collection of mediators, AURORA-RI, AURORA-OH, and AURORA-OI (Figure 5). These mediators will be of similar forms as AURORA-RH but require different mediation methodologies and MEAs. Different query rewriting algorithm and transformation rules must also be developed. We plan to implement proof-of-concept wrappers and mediators as distributed components communicating and cooperating via an ORB. Our target application is electronic commerce.

Acknowledgment. I thank my advisors, Professor M. Tamer Özsu, for the support, direction, and opportunities, and Dr.Ling Liu, for her kind help and encouragement. I thank Dr.Kelly Lyons from CAS for working with me and for her encouragement. I thank Dr.WeiDong Kou from CAS for his valuable comments.

Specially, I would like to thank Professor Alberto Mendelzon for the discussions and comments that have lead to significant improvement of this paper.

References

[ACHK93] Y. Arens, C.Y. Chee, C-N. Hsu, and C.A. Knoblock. Retrieving and Integrating data from multiple information sources. *International Journal of*

Intelligent and Cooperative Informations Systems, pages 127–158, June 1993.

[Ahm91] R. Ahmed et al. The Pegasus Heterogeneous Multidatabase System. *IEEE Computer*, 24(12):19–27, December 1991.

[Car95] M J. Carey et al. Towards Heterogeneous Multimedia Information Systems: the Garlic Approach. In *Fifth Int. Workshop on Research Issues in Data Engineering – Distributed Object Management (RIDE-DOM'95)*, pages 124–131, Tai Pei, TaiWan, March 1995.

[CHS91] C. Collet, M. Huhns, and W. Shen. Resource Integration Using a Large Knowledge Base in Carnot. *IEEE Computer*, 24(12):55–62, December 1991.

[Chu90] C. Chung. Dataplex: An access to Heterogenous Distributed Databases. *CACM*, 33(1):70–80, January 1990.

[FRV95] D. Florescu, L. Raschid, and P. Valduriez. Using Heterogeneous Equivalences for Query Rewriting in Multidatabase Systems. In *CoopIS 95*, 1995.

[FRV96] D. Florescu, L. Raschid, and P. Valduriez. Defining the search space for query optimization in a heterogeneous database management system. In *Under Review*, 1996.

[GMS95] C H. Goh, M E. Madnick, and M D. Siegel. Ontologies, Context, and Mediation: Representing and Reasoning about Semantic Conflicts in Heterogeneous and Autonomous Systems. Working Paper 3848, MIT Sloan School of Management, 1995.

[KS91] W. Kim and J. Seo. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer*, 24(12):12–18, December 1991.

[Lev96] A. Levy. Obtaining Complete Answers from Incomplete Databases. In *VLDB 96*, Bombay, India, September 1996.

[Lit90] W. Litwin et al. MSQL: A multidatabase Language. *Information Sciences*, 49(1-3):59–101, October 1990.

[LOG92] H J. Lu, B C. Ooi, and C H. Goh. On Global Multidatabase Query Optimization. *ACM SIGMOD Record*, 21(4):6–11, December 1992.

[LPL96] L. Liu, C. Pu, and Y. Lee. An Adaptive Approach to Query Mediation Across Heterogeneous Information Sources. In *Int. Conf. on Cooperative Information Systems (CoopIS)*, pages 144–156, June 1996.

[LR82] T. Landers and R. Rosenberg. An overview of Multibase. In H J Schneider, editor, *Distributed Databases*, pages 153–184. North-Holland, Netherland, 1982.

[LRO96a] A. Levy, A. Rajaraman, and J. Ordille. Query Answering Algorithms for Information Agents. In *Proceedings of the 13th National Conference on Artificial Intelligence, AAAI-96,*, Portland, Oregon, August 1996.

[LRO96b] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *VLDB 96*, Bombay, India, September 1996.

[MIR93] R. Miller, Y.E. Ioannidis, and R. Ramakrishnan. The Use of information capacity in schema integration and translation. In *VLDB 93*, pages 120–133, 1993.

[PAGM96] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object Fusion in Mediator Systems. In *VLDB 96*, Bombay, India, September 1996.

[PGGMU95] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. Ullman. A

Query Translation Scheme for Rapid Implementation of Wrappers. In *International Conference on Deductive and Object-Oriented Databases*, 1995.

[PGMU96] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A Mediation System Based on Declarative Specifications. In *ICDE 96*, pages 132–141, New Orleans, February 1996.

[PGMW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. In *ICDE 95*, pages 251–260, Taipei, Taiwan, March 1995.

[QL94] X. Qian and T F. Lunt. Semantic Interoperation: A Query Mediation Approach. Technical Report SRI-CSL-94-02, Computer Science Laboratory, SRI International, April 1994.

[RCD94] L. Raschid, Y. Chang, and B. Dorr. Query transformation techniques for interoperable query processing in cooperative information systems. In *CoopIS 94*, 1994.

[SL90] A. Sheth and J. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3), September 1990.

[SSR94] E. Sciore, M. Siegel, and A. Rosenthal. Using Semantic values to facilitate interoperability among heterogeneous information systems. *ACM TODS*, 19(2):254–290, June 1994.

[Sub] V S. Subrahmanian et al. HERMES: Heterogeneous Reasoning and Mediator System. Unpublished document, University of Maryland.

[THMB95] M. Templeton, H. Henley, E. Maros, and D.J. Van Buer. InterViso: Dealing With the Complexity of Federated Database Access. *VLDB Journal*, 4(2):287–317, 1995.

[TRV96] A. Tomasic, L. Raschid, and P. Valduriez. Scaling Heterogeneous Databases and the Design of Disco. In *Proceedings of the International Conference on Distributed Computer Systems*, 1996.

[Wie92] Gio Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, pages 38–49, March 1992.

[Yan97] L L. Yan. Building Scalable and Efficient Mediation: the AURORA Approach. *PhD thesis in preparation*, 1997.

[YOL97a] L L. Yan, T. Ozsu, and L. Liu. Accessing Heterogeneous Data Through Homogenization and Integration Mediators. In *Second IFCIS Conference on Cooperative Information Systems (CoopIS-97)*, Charleston, South Carolina, USA, June 1997.

[YOL97b] L L. Yan, T. Ozsu, and L. Liu. Mediator Join Indices. In *Seventh International Workshop on Research Issues in Data Engineering: High-Performance Database Management for Large Scale Applications (RIDE'97)*, Birmingham, England, April 1997.