A Flexible Meta-Wrapper Interface for Autonomous Distributed Information Sources *

Louiqa Raschid María Esther Vidal Jean-Robert Gruser

Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
{*louiqa,mvidal,gruser*}*@umiacs.umd.edu*

Abstract

We support flexible query processing with autonomous networked information sources. Flexibility allows a query to be accepted in a dynamic environment with unavailable sources. Flexibility provides the ability to identify equivalent sources, based on their contents; these equivalences are used to eliminate redundancy and provide alternate query plans, when some source is unavailable. We determine the best plan, i.e., the least-cost non-redundant plan, based on a cost-model for autonomous sources. These features are supported by a meta-wrapper component within the mediator. The meta-wrapper interface is defined by a structure and supported operations. WHOQL is a query language for queries and plans; it can represent sequential execution to obtain safe plans, and plans with redundancy (alternatives). A language WHODL defines the mapping from the meta-wrapper interface to each source. WHODL also describes the contents of a source. This content definition is used to determine equivalences of autonomous sources. We obtain a least-cost non-redundant plan in a dynamic environment. A meta-wrapper cost model uses three underlying sources of information: a selectivity model; a cost model for operators in the meta-wrapper; and a cost estimator for the query response time. The estimator uses a parameterized feedback technique to learn from query feedback, and to determine the relevance of various factors that affect response time. The cost model also provides feedback to the plan generator on low-cost plans.

1 Introduction

Architectures for access to data from heterogeneous databases or legacy servers, has been studied extensively and reported in [2, 3, 12, 18, 22, 23, 24]. More recent work, based on an architecture of mediators and wrappers [27], and a common object model, is reported in [1, 7, 10, 11, 12, 17, 19, 25]. The rapid growth of the Internet and Intranets, and vendor support of database interoperability protocols such as ODBC, OLE/DB [5], and ODMG/ODL, [8], etc., has increased the number of data sources accessible over networks. Scaling to a large number of data sources has also introduced several new problems. First, the Internet and Intranets are dynamic environments. One cannot guarantee that sources are always available. The first problem that must be solved is the ability to process a query in a dynamic environment of (un)available sources. These sources may support different interfaces and have different capabilities. The challenge is

*This research has been partially supported by the Advanced Research Project Agency under grant ARPA/ONR 92-J1929, the National Science Foundation under grant IRI9630102, by CONICIT, Venezuela; and by INRIA Rocquencourt, France.

to determine if a query can be accepted, without exact knowledge of the particular data sources that are available.

The second challenge that arises, with increased access to data sources, is replication of the contents of these sources. Although replication increases the probability that a query can be processed, the challenge is to eliminate redundancy of execution plans in sources and identify alternate plans, so that query processing is more efficient. Research in schema integration and semantic heterogeneity [4, 13, 14, 15] has clearly indicated that the task of information integration, when data is obtained from multiple sources, is extremely complex, and is difficult to automate. Information integration will continue to be an important task. However, a trade-off is needed between complete information integration, on the one hand, and the need to obtain answers from the available sources, with limited information on their contents. Many Web-based sources publish content descriptions that provide some semantic information on their contents. We provide several examples in this paper. Based on these content descriptions, it may be possible to identify when two or more sources can produce equivalent answers to a given query, subject to some set of constraints or restrictions. Our second challenge is to eliminate redundancy, and obtain non-redundant alternate execution plans, with (limited) equivalence information. We note that we do not claim to solve the broader problem of information integration, in general.

The third challenge is to obtain a least-cost plan. To develop a cost model for heterogeneous information servers, we must consider the following drawbacks: We do not have complete knowledge of selectivity, access paths, or the cost of physical algorithms exeuted on remote servers. The response time of these sources may be affected by dimensions such as the time and/or day that the query was submitted; the amount of data that is transferred; etc. The query processing time on the source is included in the total response time that can be measured, and it is not always possible to determine if the query processing time is significant, compared to the transfer time across the network.

We introduce a *meta-wrapper* component, into the wrapper mediator architecture, to solve these challenges. The meta-wrapper component resides within the mediator component. It provides specific functionality to the mediator, in a dynamic environment, when sources may be unavailable, and with possible redundancy of data in sources. A meta-wrapper assumes the existence of wrappers with their functionality to translate queries into a native format; and produce answers in the common model and format. A meta-wrapper also assumes that the following mediator functions are performed: decompose mediator queries into sub-queries for wrappers; handle wrappers of limited capability and wrappers whose data is a view over the mediator interface [10, 16, 20, 21]; handle semantic integration of data from multiple sources; obtain an optimal execution plan for the mediator sub-queries; etc. The meta-wrapper component will provide a special functionality to the mediator. For each mediator sub-query, it will determine a non-redundant, least-cost plan, for the available wrappers and data sources. This task cannot be accomplished by the wrappers and is indeed the responsibility of the mediator. We separate this responsibility from the other important mediator functions, and define a separate component to implement this functionality.

The meta-wrapper component has a source independent interface, SInMWrap, representing the capabilities of a number of autonomous sources that provide similar or related data. SInMWrap is loosely modeled on the ODMG standard [8]. In previous research, we solved a first problem of determining whether a mediator sub-query may be accepted in the meta-wrapper interface, i.e., is there a (combination of) source(s) that can evaluate the query, and for which there is a safe meta-wrapper plan ? The definition of the meta-wrapper interface, SInMWrap, and the algorithms to determine an accepted query, and to produce safe plans, was presented in [26]. A query language WHOQL is used to specify meta-wrapper queries and plans. WHOQL can specify that a sub-query is to be evaluated evaluated in a particular source; it can also specify the sequential ordering of subqueries in a safe plan.

A Wrapper High Order Definition Language (WHODL) is used to define the meta-wrapper interface, with respect to the sources, in Meta-Wrapper Structure Definition (MWSD). WHODL is also used to describe the contents of each source, and these content descriptions (CDs) information are used to identify "equivalent" sources, in Source Equivalence (SE). MWSD, SE, and the meta-wrapper operator implementation (MWOI), are used by an algorithm *SourcePlan*, to obtain a safe plan in some combination of sources. Algorithm *SourcePlan* perfoms two functions: (1) it will eliminate redundancy in plans; and (2) it will provide alternate evaluation plans. The language WHOQL provides a representation for the *alternate* evaluation of sub-queries in equivalent sources. When there is no source available, or the available sources do not define or support some attribute or operator used in the meta-wrapper plan, then no safe source plan will be obtained, and the meta-wrapper query is not executed. SInMWrap is designed for dynamic environments with autonomous sources; it cannot guarantee that there are always available source(s) for a safe plan.

In the event of multiple alternate plans, the meta-wrapper will identify a least-cost non-redundant plan, using a meta-wrapper cost model. There are two significant contributions of our research on a meta-wrapper cost model. First, we identify multiple sources of information (cost models) to describe this environment, and we propose a combined cost model for the meta-wrapper. The meta-wrapper cost model uses all of the following cost models: database selectivity estimated or measured from content descriptions of sources; costs for operations that are executed in the meta-wrappers; and a cost estimation of the unit response time for the sources, based on parameterized query feedback. We propose a query feedback process, similar to Hermes [1], to accurately model the total response time for such an environment. Since the response time on such networks may be sensitive to dimensions such as the time, day, the quantity of data transmitted, etc., the feedback process is parameterized to be sensitive to the measurement precision and relevance of each dimension. The second contribution of the meta-wrapper cost model is that we propose a technique, also based on parameterized query feedback, to provide the following novel feature – we can provide feedback to the plan generator, i.e., we can identify low cost plans, when the (low) cost of the plan is affected by query processing time on the source. Feedback is especially valuable with autonomous sources, where the meta-wrapper plan generator has no knowledge of the operations that are supported,or the access paths, on the source.

This paper is organized as follows: In section 2, we provide a motivating example of content descriptions of sources with replication. In section 3, we present a detailed architecture for SInMWrap, and highlight the query language WHOQL. In section 4, we introduce the language WHODL used to specify content descriptions of the sources. We demonstrate, using examples, how these content descriptions (CDs) are used to generate equivalences of sources. There are two kind of equivalences. One labelled with the keyword **equivalent** identifies alternate sources for information. These equivalences are used to generate alternate plans in any one of these sources. The second equivalence is labelled with the keyword **contains**. These equivalences are used to eliminate redundancy in plans. In section 5, we present the combined cost model, and show how a least-cost plan is obtained, and a how feedback on low cost plans is provided to the meta-wrapper plan generator. In Section 6, we describe how source equivalences (SE) is obtained from content descriptions (CDs), and we present the algorithm *SourcePlan*; it will produce non-redundant alternate source plans. Section 7 concludes.

2 Motivating Example

Consider the following autonomous sources with TV schedules; broadcast times are all in EST.

```
interface TvGuide { extent tvguide;
 attribute string SeriesTitle,EpisodeTitle,EpisodeDescription,Channel,CableTv;
 attribute enum ShowType {pay-per-view,public,cable};
 attribute date Date; attribute time Time;
 attribute enum Category {movie,sports,kids,news,comedy,educational}}
```

Figure 1: Meta-Wrapper Schema in ODL

Source S_1: A local site with a program guide for all cable channels nationwide. **Schedule(Time,Date,Channel,Program,Category)**
Source S_2: A site in California maintains program schedules only for the Discovery Channel. **Schedule(Date,Time,SeriesTitle,EpisodeTitle)**; **Episode(Title,Description)**
Source S_3: A site with a guide of movies and events on pay-per-view channels. **MovieEvent(Title,Description,Category)**; **Schedule(Channel,Title,Time,Date)**
Source S_4: A local site maintains information about a local cable TV company, Maryland Cable. Channels are *Basic, Premium* and *Pay-per-view.* **Channel(Channel,ChannelType)** **Schedule(Channel,Time,Date,SeriesTitle,EpisodeTitle,Description,Category)**

The source independent meta-wrapper interface SInMWrap for these sources is described by the meta-wrapper schema (MWS) of Figure 1.

Consider the following meta-wrapper query (in OQL syntax) posed against the SInMWrap interface:

```
select x1.EpisodeDescription
from x1 in tvguide
where x1.Channel="Discovery Channel" and x1.Date=Feb:04:97 and
      x1.EpisodeTitle="Next Step" and x1.Time=19:30.
```

The following subqueries against sources S_2 and S_4 (also in OQL syntax) will be generated:

Subquery in S_2	Subquery in S_4
select x2.Description from x1 in schedule,x2 in episode where x1.Time=19:30 and x1.Date=Feb:04:97 and x1.EpisodeTitle="NextStep" and x2.Title="NextStep"	select x1.EpisodeTitle,x1.Description from x1 in schedule where x1.Channel="Discovery Channel" and x1.Time=19:30 and x1.Date=Feb:04:97

Assume schedule and episode are the extents over the classes in the sources. Source S_2 completely evaluates the meta-wrapper query. Source S_4 will retrieve the title and description for whatever event is on the Discovery Channel. It cannot perform a selection on the title; the source operators SO, (to be discussed later), indicate that S_4 cannot execute this selection. The selection of *EpisodeTitle="Next Step"* will be performed in the meta-wrapper. Futher, using the content descriptions of the sources, SInMWrap will identify that these two subqueries actually retrieve equivalent data. The least-cost sub-query is selected for evaluation, based on the available sources. Mediator Systems such as RQDL [20] and IM [17] will not be able to determine that these queries are equivalent and both queries will be submitted.

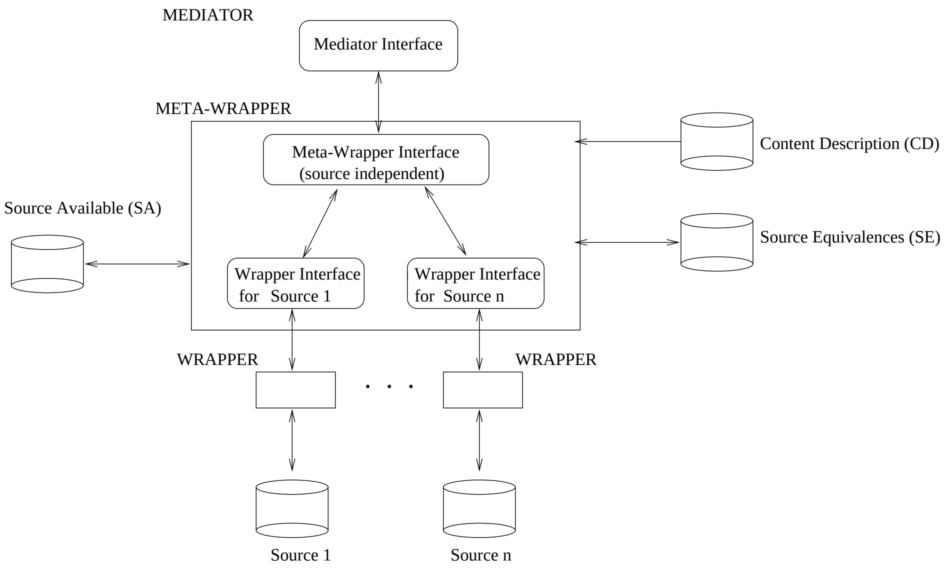

Figure 2: SInMWrap - A Source Independent Meta-Wrapper component within the Mediator component

3 Architecture for SInMWrap

The source independent meta-wrapper architecture for SInMWrap is shown in Figure 2. The SInMWrap interface is described by (1) a meta-wrapper schema description in ODMG-ODL; a set of meta-wrapper operators; the set of input restrictions or variables which must be bound in the input query; and a set of semantic restrictions or integrity constraints that describe the meta-wrapper. Each of the sources is in turn described by a set of source operators ; source input restrictions; and source semantic restrictions. In this paper, for simplicity, we do not consider either input restrictions or semantic restrictions. The meta-wrapper operators are the relational algebra operators, eg. select, project, union, etc., and the scan operator. The sources themselves may not be relational databases; for example, a source may be a URL of a CGI script, or an IR search engine. In this case, the wrapper for each source may implement some (subset) of the meta-wrapper operators. We assume that **all** wrappers support *at a minimum*, the scan operator on the source; this restriction has been discussed in [26].

We express the meta-wrapper queries in OQL syntax. The Level 1 Mediator model and language proposed in [6] follows the OQL syntax, and the Level 1 core language is equivalent to the subset of relational queries. We assume the same restriction for meta-wrapper queries. We note that OQL was chosen for the core because of its existing extensions to abstract types, complex types, etc.; (see [6] for details).

We refer to Figure 3 to illustrate query processing in SInMWrap. We use an algorithm *Meta-WrapperQuery*,

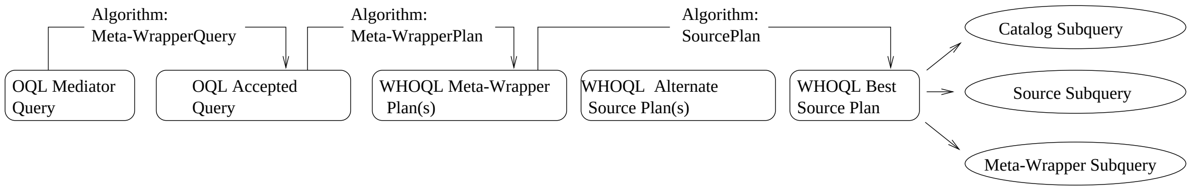

Figure 3: Dataflow in the SInMWrap architecture.

to determine (1) the *accepted* (OQL) meta-wrapper query, i.e., the set of operators that are supported in the meta-wrapper. Algorithm *Meta-WrapperQuery* produces an OQL query for evaluation in the meta-wrapper. Next, we translate the accepted OQL meta-wrapper query into a WHOQL meta-wrapper query. WHOQL is the language used to express queries and plans; its features are discussed later. Then, algorithm *Meta-WrapperPlan* makes sure there is a plan in the meta-wrapper for this query and produces a WHOQL meta-wrapper plan. All of this is processed in the meta-wrapper interface without identifying particular sources. It is also possible that a safe plan will not be found; (See [26] for details).

We use a definition language WHODL, for two purposes: (1) to provide the meta-wrapper structure definition MWSD, which defines the meta-wrapper structures in terms of the source structures; (2) to describe the contents of each of the sources or, the Content Definitions (CDs). CDs are used to generate "equivalence" and "containment" relationships between sources, and this is represented in Source Equivalence (SE). The implementation of meta-wrapper operators by source operators is specified in the meta-wrapper operator implementation (MWOI); MWOI is used to translate operators in a meta-wrapper plan into source operators in a source plan.

Algorithm *SourcePlan* now obtains a safe source plan SP, for the accepted meta-wrapper plan. It uses the following: (1) the meta-wrapper structure definition (MWSD); (2)the meta-wrapper operator implementation (MWOI); (3) the set of available sources (SA); and (4) the source equivalences (SE) obtained from the CDs. Since *SourcePlan* uses SE, it is able to eliminate redundant plans and also generate alternate plans. Also a *best* safe source plan, SP, is chosen based on least cost. This safe plan SP is sent for evaluation to the respective wrapper(s) and if the wrapper(s) does not respond, an alternative safe plan may be evaluated. Finally, if some wrapper(s) respond and other do not respond, then some partial answers may be obtained.

We use the WHOQL query language to specify queries and plans. WHOQL uses different connectives to represent relationships between subqueries in a plan. WHOQL uses a parallel connective "∪" when subqueries are independent. The "∪" connective is also used to represent a *union* when a subquery can

OQL meta-wrapper query	WHOQL meta-wrapper query
select x1.EpisodeTitle from x1 in tvguide where x1.ShowType="pay-per-view" and x1.Category="movie" and x1.Date=Feb:10:97.	query(<episodetitle G1>):- tvguide(<episodetitle G1> 1 <showtype "pay-per-view"> <category "movie" ><date "Feb:10:97">). 2

Figure 4: An OQL meta-wrapper query and its WHOQL representation

be independently evaluated in multiple sources. WHOQL uses a sequential connective "," to represent an ordering between subqueries in a plan. This ordering ensures that the WHOQL plan is safe, i.e., all the input restrictions are satisfied, or the variables are bound. WHOQL uses an alternative connective"|" to represent alternatives in plan execution. A WHOQL source plan can also specify the evaluation of a subquery in a *particular* source. A WHOQL safe source plan is composed of different sub-queries: *catalog subqueries* retrieve schematic data from the catalogs; *source subqueries* are the queries evaluated against the sources; and *meta-wrapper subqueries* bind variables and combine answers from the sources. These subqueries are shown in Figure 3. WHOQL also has high order features to resolve schematic conflicts between the meta-wrapper structures and the source structures. A definition of WHOQL is in [26].

In Figure 4, an OQL meta-wrapper query and its corresponding WHOQL representation are shown. The WHOQL well formed query is constructed by WHOQL conjunctors and the connectives { |,",",∪ } described previously. A conjunctor is highlighted in a box labelled 2 and a term is highlighted in the box labelled 1, in Figure 4.

4 Equivalent Sources, Redundancy and Alternate Plans

A content description (CD) for each source, w.r.t the meta-wrapper interface schema is obtained, if possible. The language WHODL is used to specify CD. The definition of WHODL is in [26]. Here we use some examples to describe the CDs for the different sources. These CDs are then used to obtain source equivalence SE. The SE is used by the algorithm *SourcePlan* to eliminate redundancy in plans and produce alternate plans. There are two kind of equivalences. One labelled with the keyword "**equivalent**" identifies alternate sources for information. These equivalences are used to generate alternate plans in any one of these sources. The second equivalence is labelled with the keyword "**contains**". These equivalences are used to eliminate redundancy in plans. In this section we provide examples of CDs for different sources, and then examples of SE that are obtained from these CDs. We then show meta-wrapper plans where we eliminate redundancy

equivalent x in TvGuide 1 to **S**$_2$ 2
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,Category) 3
when (x.Channel="Discovery Channel" and
x.ShowType="cable" and x.Category="educational"). 4

equivalent x in **TvGuide** to **S**$_1$
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,Category).

equivalent x in **TvGuide** to **S**$_3$
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,EpisodeDescription,Category, ShowType)
when (x.ShowType="pay-per-view" and
(x.Category="movie" or x.Category="event")).

equivalent x in **TvGuide** to **S**$_4$
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,EpisodeDescription,Category, ShowType)
when (x.CableTv="Maryland" and
(x.ShowType="pay-per-view" or x.ShowType="cable")).

Figure 5: Source Content Definition (CD) for S_1, S_2 and S_3

and produce alternate plan.

Each CD comprises (1) the meta-wrapper class; (2) a source; (3) a list of meta-wrapper attributes of (1) defined on the source structures of (2); (4) a boolean expression. The boolean expression is a set of restrictions, in conjunctive normal form (CNF), that must be satisfied for this CD to be used. This boolean expression must be satisfied by the meta-wrapper query, if this CD is to be used to produce alternatives in the source plan. Each component is labelled 1,2,3 and 4 in Figure 5.

The CDs for all our sources are in Figure 5. In English, the CD for S_2 states that S_2 has data on attributes Time, Date, Channel, SeriesTitle, EpisodeTitle and Category and, the boolean expression futher specifies that S_2 only has data for the "Discovery Channel", where the ShowType is "cable" and where the Category is "educational".

The CDs are processed by algorithm *SourceEquivalence* to produce source equivalences SE. The actual structure of SE is in section 6, and the algorithm *SourceEquivalence* is in [26]. In Figure 6, we provide an informal specification of SE obtained from the CDs in Figure 5. Each equivalence in SE in Figure 6 comprises the following: (1) a label identifying the equivalence; (2) the keyword **equivalent** or **contains**; (3) the meta-wrapper class, e.g., (TvGuide) and a pair of sources participating in this equivalence; (4) a list of meta-wrapper attributes; (5) a boolean expression in conjunctive normal form (CNF) is a set of restrictions. The boolean expression must be satisfied in the query, if this source equivalence is to be used to eliminate redundancy or identify alternate plans. When the source equivalence is expressed with the keyword **equivalent**, it means that the data maintained in the 2 sources is equivalent, when the boolean expression holds. When the keyword is **contains**, then the data in the first source contains all the data in the second source, when the boolean expression holds.

Obtaining SEs from the CD that can be utilized is not straightforward. Consider the CDs of S_1 and S_3 and $\mathbf{SE}_3$ and $\mathbf{SE}_5$. The data in source S_3 is completely replicated in source S_1, i.e., source S_3 has data on pay-per-view movies and events and this information is also replicated in S_1. Thus, we **can** utilize the equivalence $\mathbf{SE}_3$ with the **contains** keyword. However, we **cannot** utilize the equivalence $\mathbf{SE}_5$ with the **equivalent** keyword, although the equivalence does hold. To explain, for SE_5 to be utilized, the two sources S_1 and S_3 must have the capacity to retrieve the subset of data that satisfies the equivalence. In source S_1, although we know that it contains data on pay-per-view channels, the source does not have the capability to identify or retrieve this subset of data. Thus, it cannot be considered an alternate source, when the query requests data on pay-per-view channels. Section 6 provides a definition for SE that can be utilized.

4.1 Obtaining Source Plans and Alternate Plans

For each meta-wrapper plan, algorithm *SourcePlan* will do the following: (1) it will identify those sources against which the plan can be evaluated; (2) It will identify equivalences among the sources, from SE, that are applicable, i.e., the boolean expression of the equivalence holds in the query; (3) it will determine availability of the sources in SA; (4) It will eliminate redundancy and identify non-redundant alternate source plans SP for the available sources. Algorithm *SourcePlan* is described in section 6. Here we use several examples to illustrate.

In Example 4.1, the sources against which the meta-wrapper plan can be evaluated are as follows: $(S_1 \cup S_2 \cup S_4)$, i.e., initially the algorithm determines that the query can be evaluated in all of these three sources. From SE, for the attributes and conditions of this plans we determine that the following sources are indeed equivalent, $\{S_1 \equiv S_2 \equiv S_4\}$, and SA indicates that source S_4 is unavailable. The alternate plan for the available sources is then evaluated against the following sources: $(S_1 \mid S_2)$, i.e., one of the sources can be chosen. The meta-wrapper cost model will be used to choose the least cost plan against one of the sources.

In Example 4.2, the meta-wrapper plan is initially found to be evaluable in three sources, as follows: $(S_1 \cup S_3 \cup S_4)$. From SE, for the attributes and conditions of this plan, we have the following equivalences that can be applied $\{S_3 \subset S_1\}$ and $\{S_4 \subset S_1\}$. In English, the answer to the query from source S_3, or from

SE$_1$: equivalent $\boxed{\textbf{TvGuide}, \textbf{S}_1, \textbf{S}_2}$ 3
for $\boxed{\text{(Time,Date,SeriesTitle,EpisodeTitle,EpisodeDescription)}}$ 4
when (x.Channel="Discovery Channel" and
x.ShowType="cable" and x.Category="educational") 5

SE$_2$: equivalent TvGuide, **S$_3$**, **S$_4$**
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,EpisodeDescription)
when (x.ShowType="pay-per-view" and
(x.Category="movie" or x.Category="event")).

SE$_3$: contains TvGuide, **S$_1$**, **S$_3$**
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,EpisodeDescription,Category)
when (x.Category="movie" or x.Category="event").

SE$_4$: contains TvGuide, **S$_1$**, **S$_4$**
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,EpisodeDescription,Category)
when (x.ShowType="pay-per-view" or x.="cable").

SE$_5$: equivalent TvGuide, **S$_1$**, **S$_3$**
for (Time,Date,Channel,SeriesTitle,EpisodeTitle,EpisodeDescription)
when (x.ShowType="pay-per-view" and
(x.Category="movie" or x.Category="event")).

Note: **SE$_5$** cannot be utlized.

Figure 6: Source Equivalences

source S_4, is contained in the answer to the query from source S_1. Then, the final non-redundant plan will be on the following source: (S_1); all other plans produce partial answers and cannot be selected.

In Example 4.3, the meta-wrapper plan is initially found to be evaluable on two sources as follows: ($S_3 \cup S_4$), i.e., the query can be evaluated in both sources. From SE we have $\{S_3 \equiv S_4\}$. Consequently, the non-redundant source plan is on the following sources: ($S_3 \mid S_4$), i.e., we would choose only one source. We provide this example plans here, the plans in the two sources are completely different. We use this example to illustrate how the meta-wrapper cost combined model obtains the least-cost plan.

Finally, suppose we consider a plan which is initially determined to be evaluated against the following sources: ($S_1 \cup S_2 \cup S_3$). Also suppose that we have the following equivalences: $\{S_1 \equiv S_2\}$ and $\{S_1 \equiv S_3\}$. The conditions for each of these equivalences to hold are different, and hence it is **not** the case that $\{S_1 \equiv S_2 \equiv S_3\}$. From this, the alternate plan that is obtained is on the following sources: ($S_1 \mid S_2 \cup S_1 \mid S_3$). Now, the meta-wrapper cost model will select a least-cost plan. Suppose the least-cost plan is executed on the following sources: ($S_1 \cup S_3$). However, given that there is also a plan that can be evaluated only on one source: (S_1), it may seem that the latter is a better plan. When there is a significant overhead for accessing multiple sources, then the cost-model may choose the latter plan as the least-cost plan.

Example 4.1 Consider the following WHOQL meta-wrapper plan:

plan(<episodetitle G1>):-
    tvguide(<episodetitle G1><channel "Discovery Channel"><time "19:30" ><date "Feb:04:97">).

Algorithm *SourcePlan* initially produces a source plan against sources S_1, S_2 and S_4 as follows:

plan(<episodetitle G1>):-
    S$_1$**:**{schedule(<program G1><channel "Discovery Channel"><time "19:30" ><date "Feb:04:97">)}
    ∪
    S$_2$**:**{schedule(<episodetitle G1><time "19:30" ><date "Feb:04:97">)}
    ∪
    S$_4$**:**{schedule(<episodetitle G1><channel "Discovery Channel"><time "19:30" ><date "Feb:04:97">)}

Suppose source S_4 is unavailable. Futher $\{S_1 \equiv S_2\}$ for this query. The algorithm *SourcePlan* will produce the following alternate source plan in S_1 or S_2; the cost-model will be used to select the least-cost plan:

plan(<episodetitle G1>):-
    S$_1$**:**{schedule(<program G1><channel "Discovery Channel"><time "19:30" ><date "Feb:04:97">)}
    |
    S$_2$**:**{schedule(<episodetitle G1><time "19:30" ><date "Feb:04:97">)}□

Example 4.2 Consider the following WHOQL meta-wrapper plan:

plan(<episodetitle G1><channel G2 >):-
tvguide(<episodetitle G1><channel G2><category "movie"><date "Feb:04:97">).

SourcePlan will initially produce the following source plan, which has 3 parallel subqueries:

plan(<episodetitle G1><channel G2 >):-
S_1:{schedule(<program G1><channel G2><category "movie" ><date "Feb:04:97">)}
$\cup$
S_3:{schedule(<title G1><channel G2><date "Feb:04:97">),
movieevent(<title G1><category "movie">))}
$\cup$
S_4:{schedule(<episodetitle G1><channel G2><date "Feb:04:97"><category "movie">)}.

Using SE, *SourcePlan* can determine that $\{S_3 \subset S_4\}$ and $\{S_4 \subset S_1\}$, for the condition Category="movie".

Consequently, *SourcePlan* will produce the following non-redundant plan:

plan(<episodetitle G1><channel G2 >):-
S_1:{schedule(<program G1><channel G2><category "movie" ><date "Feb:04:97">)}.□

Example 4.3 Consider the following WHOQL meta-wrapper plan:

plan(<episodetitle G1><channel G2><time G3>):-
tvguide(<episodetitle G1><channel G2><time G3><date "Feb:04:97"><showtype "pay-per-view">
<category "movie">),("16:00" $\leq$ G3 $\leq$ "23:00").

This example illustrates many features of the Source Plan algorithm. Although source S_1 does contain the answer to this query, this source cannot provide information to satisfy the criterion ShowType="pay-per-view". Thus, *SourcePlan* will not consider source S_1. It will also not consider any equivalences involving S_1. Sources S_3 and S_4 do contain answers to this query and can satisfy the criterion ShowType="pay-per-view". In fact, source S_3 only contains information on pay-per-view channels. Then from SE we have $\{S_3 \equiv S_4\}$ for the restriction ShowType="pay-per-view". Consequently, alternate plans are generated as follows:

plan(<episodetitle G1><channel G2><time G3>):-
(**S_3:**{movieevent(<title G1><category "movie">),
schedule(<channel G2><time G3><date "Feb:04:97" ><title G1>)},
meta-wrapper:{"16:00" $\leq$ G3 $\leq$ "23:00"})
|
(**S_4:**{schedule(<channel G2><time G3><date "Feb:04:97"><episodetitle G1><category "movie">),
("16:00" $\leq$ G3 $\leq$ "23:00"),
channel(<channel G2><channeltype G4>)}),
meta-wrapper:{G4="pay-per-view"}).□

Source S_3 does not support a selection on attribute Time. Similarly, source S_4 does not support a selection on attribute ShowType (attribute ShowType in the meta-wrapper interface corresponds to attribute ChannelType the source S_4). *SourcePlan* will use MWSD and the meta-wrapper operator implementation MWOI to determine this. These two selections selections in the two plans, will actually be executed in the meta-wrapper. Details of all these features in [26].

5 Meta-wrapper combined cost-model

5.1 Motivation

Several important features differentiate a traditional database cost model from a cost-model for heterogeneous information servers. First, database cost-models have complete knowledge of database selectivity and the cost for physical operations; this is not the case with autonomous sources. The CDs of servers may provide information that can be used to obtain some selectivity information, by running some queries. Second, distributed database cost-models can accurately estimate the response times for queries. For servers accessed across the Internet and Intranets, such cost-models do not exist. Further, the transfer time for data may be more significant than the query processing time on the source. In addition, dimensions such as the day, the time, and the amount of data being transferred, affect the total response time. There has been no cost-model for this environment. There is also no research on determining when the total response time is affected/not affected by the processing time on the source.

To overcome these two limitations, we propose a feedback process, similar to HERMES [1] , that has two purposes : One is to accurately model the total response time, sensitive to the dimensions mentioned above. The feedback process is parameterized to be sensitive to the measurement precision and order of relevance of each dimension. The second purpose of the feedback process is to provide feedback to the plan generator of SInMWrapp.

Thus, there are two very significant contributions of our research. First, we identify multiple sources of information (cost-models) to describe this environment, and we propose a combined cost-model for the meta-wrapper. This meta-wrapper cost-model uses all of the following cost-models: database selectivity estimated or measured from CDs of sources; costs for operations that are executed in the meta-wrapper; and finally a cost estimation of the unit total response time based on parameterized feedback.

Second, we propose a technique, also based on the parameterized feedback technique, to provide the following novel feature – we can provide feedback to the SInMWrapp plan generator, i.e. we can identify low cost plans, when we believe the (low) cost of the plan is affected by the query processing time on the source. This feedback is especially valuable with autonomous sources, where SInMWrapp has no knowledge of access paths, physical algorithms, etc., on the source.

In this paper, we only describe a technique to provide feedback on (conjunctive) select-project queries. We believe that even a limited feedback is very useful in the context of heterogeneous servers, since many sources that we have accessed on the Internet are limited to accepting such select-project or select queries.

5.2 Related work

In [9, 12] a mediator cost-model is developed for data sources. They assume that calibration databases can be built on remote sources, i.e. they accept updates. The generic logical cost-model is calibrated by experiments on a calibrating database created in each data source. The mediator system automatically populates and queries each remote calibrating database to instantiate the parameters of the cost-model. Unfortunately, most data sources we accessed do not accept updates. The DISCO project [25] contacts wrappers to get the cost of each plan. The wrapper stores, for each data source, the available physical operators and their corresponding costs. However, wrappers for web data sources do not store or communicate such information.

The HERMES system is able to model web data sources since it uses only query feedback. Each query is stored in tables, with information on the domain of attributes and query bindings, i.e. if the attributes are bound or free. A Domain Cost and Statistics Module (DCSM) uses the cardinality, the time for a first answer, and the total response time for each query. Using off-line loss-less summarization of these tables, HERMES gives an approximation of the selectivity and the response time for each input query. Our approach is different. Instead of storing information for each query, we directly store some global meta-data characterizing the response time. The meta-wrapper refines this global meta-data using query feedback. Also, the meta-wrapper combined cost-model uses a database selectivity model based on simple statistics. This avoids the need to store information for each query as in HERMES, but still provides selectivity estimation. We also learn from information on additional dimensions such as the day, the time, and the quantity of transmitted data. Finally, we provide feedback to the plan generator.

5.3 Examples of finding a least-cost plan

WHOQL logical plans are first transformed into physical plans. The meta-wrapper logical operators are transformed into physical algorithms ($operator \rightarrow PHY\_Algorithm$) to be executed in the meta-wrapper. Logical operators on remote data source are transformed into remote physical operators ($operator \rightarrow Rem\_PHY\_Operator$); this is passed to the wrapper for that data source. When data is transferred from a remote data source to the meta-wrapper, a *Transmit* operator is inserted into the plan. The structure of the data to be transmitted is determined by the projection in the remote data source physical operator.

In order to calculate the cost of each query plan, the combined cost model estimates the cost of each operator in a bottom-up fashion. The cost of the *Transmit* operator is modeled by the total response time, which is estimated by the parameterized feedback process. *This response time includes the time for remote query processing on the source.* Our procedure for estimating this time is to first determine the quantity of

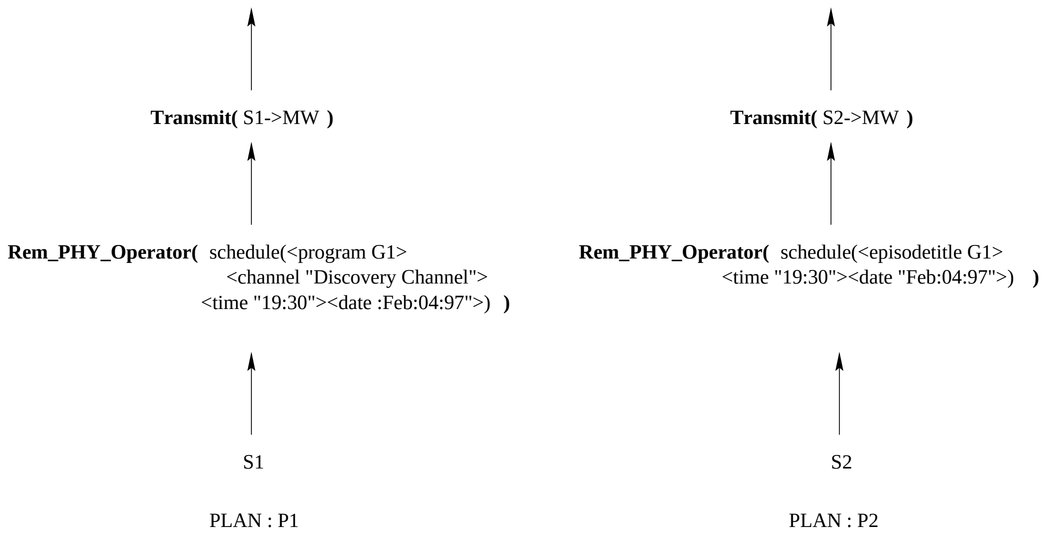

Figure 7: Similar meta-wrapper plans evaluated in alternate sources

the transmitted data using the selectivity model. We then calculate the cost of the *Transmit* operator using the cost estimator. The meta-wrapper cost model is used to calculate the cost of operations executed in the meta-wrapper.

Our first example illustrates the effect of the transmit operator on the cost of the plan. The two alternate plans of Figure 7 are the equivalent physical plans of the source plan of example 4.1. Each physical plan accesses one data source, S_1 and S_2 respectively. The two sources support the same selection operation and the two plans have very similar remote select queries. The amount of data transfered is also the same. However, suppose the two data sources are located in different time zones, and the parameterized cost estimator has differentiated between the two sources, based on the best and worst access windows. In this example, the time the query is submitted is important. The combined cost-model will choose a source with the best access window. This will be done by the cost-model that estimate the unit response time for the sources.

The second example chooses between two different meta-wrapper plans, i.e the two plans execute different operations on the source and meta-wrapper. The plans of Figure 8 are the equivalent physical plans of the source plan of example 4.3. The source S_3 of the first plan only contains data on pay-per-view programs; however, this source can only select the programs for a given date. For this plan, the selection of programs for a given time is executed in the meta-wrapper. The source S_4 of the second plan contains data on all channels including pay-per-view. This source can select programs for a given date and time. The selection on pay-per-view programs is not obvious; S_3 provides pay-per-view data, but S_4 can perform a selection on

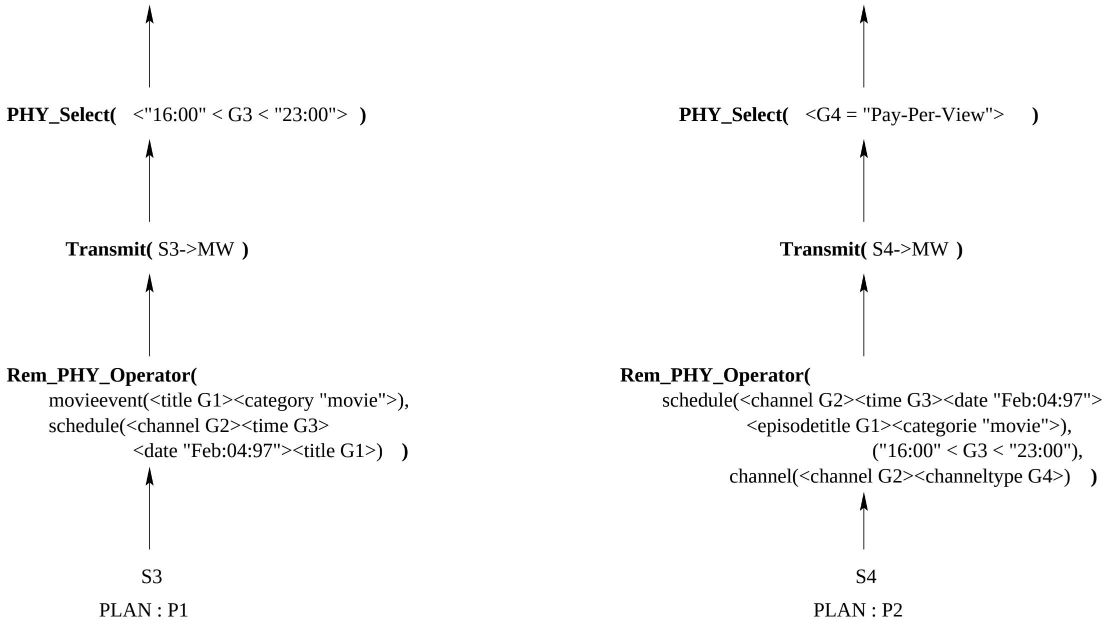

Figure 8: Alternate meta-wrapper plans with different sources

the time and date. The combined cost-model will use all three cost-models in this case to determine the least cost plan, since the quantity of data transferred and the operations in the plan are different. In addition, the sources will have different unit response time.

Our third example considers the first plan P1 in Figure 7 and describes the feedback given to the plan generator. The cost estimator may determine that the unit response time is sensitive to the query bindings in the selection operation on the source. The explanation may be that the query processing of the operators on this source is efficient. For example, when there is a selection on the *date* field in the query, there may be a significant deviation in the response time, independent of other dimensions. The cost estimator may conclude that these bindings are significant and may return information on this least-cost plan to the query generator. The query generator can exploit such information for a source that only allows selections on either the *date* field or the *time* field, but not on both. Many sources accessible over the Internet have similar restrictions, when neither the wrapper nor the source can evaluate a boolean expression. Processing of this particular query (and bindings) on this source may be affected by the organization (indexing or clustering) of data in this source.

5.4 The cost-model

The meta-wrapper combined cost-model combines three different cost-models :

- A database selectivity model : For uniform distribution of data we store the usual system-R [29] meta-data such as the minimum (*min*), the maximum (*max*) and the number of distinct values (*NDist*) for each attribute. The system stores the cardinality (*Card*) of collections, and the size (*Size*) of objects (in the wrapper interface), for each data source. For non-uniform distributions, we use histograms. These histograms are built by processing the frequencies (*fi*) of each value i of the attribute domain. The results are integrated into buckets. The number of buckets (*Nbucket*) determines the size and the precision of the given histogram [28]. For some data sources, the attribute domains might not be enumerated. In that case, uniform approximation will be used. As in database systems, if the selectivity model performance decreases, the system may decide to refine the model by re-processing some selectivity meta-data. A non-uniform distribution is easily detected, when the cardinality of the result differs significantly from the uniform selectivity estimation.

- The meta-wrapper cost-model. This model reflects main memory physical implementation for the usual operators : union, join, selection, etc.

- A parameterized cost estimator : It estimates the total response time from remote data sources. The cost estimation is based on query feedback. If two plans have similar response times for two remote sources, then the meta-wrapper cost-model is used to select between the two plans.

The first two cost-models are similar to those reported in the literature. We now discuss the parameterized cost estimator for autonomous data sources.

5.4.1 The MDT to store meta-data and the cost estimation for plans

The cost estimator uses a parameterized **Multi Dimensions Table** (MDT); this stores the unit response time of past query plans. For each response time, the MDT also stores a confidence coefficient, based on the number of queries used to approximate these values. Table 1 shows the MDT for source S_1. The MDT is a hierarchical structure. The actual structure is determined by the order of relevance of each dimension. For the MDT of source S_1, the ordering of the dimensions is day, time, and quantity of data transferred, in decreasing order of relevance.

The MDT structure has been designed to quickly match each incoming query plan, based on the ordering of the dimensions, starting from the most relevant. For example based on the MDT of Table 1, a query to be processed on Tuesday at 7pm, estimated to transmit 80K-Bytes of data, matches the MDT unit transfer cost of 45.63 milliseconds per K-Bytes. This is used to estimate the response time of the input query plan, which will be calculated to be 3.65 seconds.

Monday-Friday(25.67ms)								Saturday-Sunday(12.7ms)
8am-6pm(27.02ms)				6pm-8pm(36.32ms)			8pm-8am(11.57ms)	1am-12pm(12.7ms)
<5K	5K-50K	50K-60K	>60K	<10K	10K-50K	>50K	0K-100K	0K-100K
25.02ms	25.63ms	25.80ms	35.80ms	25.84ms	36.72ms	45.63ms	11.57ms	12.7ms

Table 1: MDT for source S_1 after some learning based on query feedback

In the MDT, for each intermediate dimension, we maintain an approximation of the response time. This is calculated to reflect the confidence of each of the costs (of the next most deeply nested dimension) over which this approximation is made. For example, the entry for Monday-Friday, 8am-6pm, has a value 27.02ms. This is (weighted) average of $((25.02 + (4 * 25.63) + 25.80 + 35.80)/7)$ where the 4 values of the next most inner dimension had a confidence of 1, 4, 1 and 1 respectively.

Each dimension of the MDT has a sliding scale, and this determines the precision of the cost estimation. If an input query plan has an exact match in the MDT for a given dimension then the estimation will be more accurate. The more precise the scale on the domain of a dimension, then the more accurate this dimension is to the cost estimation of this data source. For example, on Saturday-Sunday, the time and the quantity of data transfered dimensions are at the maximum of their scale, the estimation will be less accurate. Given the particular scale of the dimension domains of the MDT of Table 1, the maximum number of entries stored in a MDT is : $7 + 7 * 24 + 7 * 24 * 10 = 1855$ pairs of (dimension, response time).

5.4.2 Learning in the parameterized cost estimator

Let us first assume for simplicity that the order of relevance of the dimensions has been determined. Later we explain how this order of relevance is determined. The initial MDT stores the maximum scale of the domain of each dimension, and a unique initial value of response time. For example, the initial value for MDT of Table 1 was 12.7ms per K-Bytes for Monday to Sunday, 1am to 12pm, 0 K-Bytes to 100 K-Bytes. The goal of the parameterized cost estimator learning algorithm is to either create new entries in the initial MDT, or to correct values in the existing structure, to improve the global behavior of the cost estimator.

When a new response time for an executed plan is obtained, the learning algorithm parses the MDT entries to determine :

- (i) if the new response time is significantly different, i.e, outside the deviation range, from the "matching" entry in the MDT. If so, the algorithm will split the domain of the associated dimension of the MDT. A new entry (Thursday, 1:30am, 50k, 11.57ms) will split an initial value Monday-Friday, 1am-

```
Algorithm Learning Algorithm(query plan feedback qpf:(day,time,quantity,response_time))
For each value d:(value,response_time,confidence,deviation) of each dimension of the MDT
    If match(qpf,d)
        If Within_Deviation(d.deviation,qpf,d) correct(d,qpf,d.confidence)
        Else split_MDT(qpf,d)
```

Figure 9: The Learning Algorithm

12pm, 0K-100k (25.67ms) into two Monday-Friday, 8am-8pm, 0K-100k (25.67ms) and Monday-Friday, 8pm-8am, 0K-100k (11.57ms). Note that the range[1] 8am-8pm is not affected by this response time remains at 25.67ms.

- (ii) if the approximated response time for the "matching" entry in the MDT has to be corrected. For example the entry (Tuesday, 7pm, 40k, 37.12ms) will correct the initial value Monday-Friday, 6pm-8pm, 10K-50k (36.32ms) to Monday-Friday, 6pm-8pm, 10K-50k (36.72ms).

The learning algorithm is described in Figure 9. Learning is controlled by the following factors :

- The ordering of the dimension determines the structure of the MDT. The ordering corresponds to the relevance of the dimension in estimating the cost of the *Transmit* operator.
- The minimum scale of each dimension, e.g. 1 hour for time, affects the accuracy of the cost estimation and determines the size of the MDT.
- The allowed deviation of the response time of each dimension: For each dimension, we consider the approximation value stored in the MDT, and then determine the deviation of the new response time compared to the stored value. If the deviation is greater than the pre-determined allowed deviation for that dimension, the learning algorithm decides that a new range has to be created for this dimension in the MDT. If not, the current value for this dimension is corrected according to its confidence coefficient.

The initial order of relevance for the MDT is chosen as day, time, quantity of data transmitted as in Table 1. As part of the learning algorithm, we run off-line statistical algorithms, and determine, for each dimension, its variance, normalized with respect to the scale of the domain. This can be used to determine a new order of relevance and we note that this will restructure the MDT entirely and all average values for intermediate dimensions have to be approximated again [30].

[1] Later, this range has been split in Table 1 into (8am-6pm) and (6pm-8pm).

5.5 Feedback to the plan generator

Feedback to the plan generator is in the form of a low-cost plan. we provide this feedback when the algorithm can determine that the processing time on the source is significant to the total response time and that this is independent of all dimensions. Then the query pattern (bindings) for such low-cost plans is returned to the plan generator.

For this learn to occur, a feedback algorithm maintains, for each deepest dimension entry of a MDT, the query patterns that were used to obtain the average response time for that entry. Then, for a given dimension value, this algorithm is able to determine the best bindings corresponding to low-cost plans. If the same best pattern is found to be independent for all other dimensions, then the feedback algorithm deduces that this is a best pattern for the plan generator.

This information can be sent to the query plan generator. The query plan generator can use heuristics to always use this best binding. This drastically improves the performance of the optimization algorithm since we are greatly limiting the search space.

6 WHOQL Source Plan Generation

A meta-wrapper plan is transformed into one or more source plans SP. An algorithm *SourcePlan* performs this transformation. This algorithm uses the meta-wrapper structure definition (MWSD), the meta-wrapper operator implementation (MWOI), the set of available sources (SA), and the source equivalences (SE). Since *SourcePlan* uses SE, it is able to eliminate redundant plans and also generate alternate plans. Finally, using a cost model, a least-cost non-redundant plan is chosen. An example WHODL definition MWSD, for the meta-wrapper interface TvGuide, in source S_2 is shown in Figure 10. Each definition of MWSD comprises (1) the meta-wrapper class; (2) a list of sources; (3) the meta-wrapper attributes of (1) defined on the source structures of (2); and (4) a boolean expression. The boolean expression must be satisfied by the meta-wrapper query, if this definition is to be used to obtain a source plan. [26] has details of the MWOI.

6.1 Equivalent Sources Definition

For each meta-wrapper class C_i, SE has an element SE_i. Each SE_i contains a set of elements $\{SE_{i,j}\}$. Each $SE_{i,j}$ has a keyword **equivalent** or **contains** and identifies either a single source, or a combination of sources, $SS_{i,j}$, with redundant data for some attributes $Att_{i,j}$ of C_i. Each $SE_{i,j}$ specifies a set of conditions, $Cond_{i,j}$. Equivalences labelled with the keyword **equivalent** are used to generate alternate plans in SP;

define **x** *in* **TvGuide** 1 *as*

y1 *in* $\mathbf{S}_2$:Schedule,**y2** *in* $\mathbf{S}_2$:Episode 2

x.SeriesTitle:=**y1**.SeriesTitle; x.EpisodeTitle:=**y1**.EpisodeTitle ;
x.EpisodeDescription:=**y2**.Description; (if y1.EpisodeTitle=y2.EpisodeTitle)
x.Date:=**y1**.Date ; x.Time:=**y1**.Time 3

when (x.Channel="Discovery Channel" and
x.ShowType="cable" and x.Category="educational"). 4

Figure 10: Meta-Wrapper Structure Definition

those with the keyword **contains** are used to eliminate redundancy in SP.

Definition 6.1 (Source Equivalence) $SE = \{(C_1, SE_1), \cdots, (C_n, SE_n)\}$,

- C_i *is a meta-wrapper structure.*
- $SE_i = \{SE_{i,1}, \cdots, SE_{i,p}\}$ *where* $SE_{i,j} = (SS_{i,j}, Att_{i,j}, Cond_{i,j},$ {**equivalent** | **contains**})
  - $SS_{i,j}$: $\{SS_{i,j,1}, \cdots, SS_{i,j,m}\}$.

    $SS_{i,j,k}$ *is a single source* $S_{i,j,k,1}$ *or a source sequence* $S_{i,j,k,1} \cdots S_{i,j,k,t}$.
  - $Att_{i,j}$: *attributes of* C_i *with equivalent values in all the* $SS_{i,j,k}$.
  - $Cond_{i,j}$: *conditions that must hold in all the* $SS_{i,j,k}$ *for the equivalence of sources.*
  - **equivalent** *indicates that the data in sources* $\{SS_{i,j,1}, \cdots, SS_{i,j,m}\}$ *are equivalent;*

    contains *indicates that the data in each source* $SS_{i,j,l} \subset SS_{i,j,1}$, $2 \leq l \leq m$.

Definition 6.2 (Equivalent Sources) *Let* C *be a meta-wrapper class. Let* S_i, S_j, *be sources. Let* cd_i, cd_j, *be the following CDs in WHODL:*

(cd_i*: equivalent* C *to* S_i *for* Att_i *when* $Cond_i$*), and (*cd_j*: equivalent* C *to* S_j *for* Att_j *when* $Cond_j$*).*

Then ($SE_{i,j}$*: equivalent* C*,* S_i *to* S_j *for* $Att_{i,j}$ *when* $Cond_{i,j}$*), is an element of SE if and only if*

Compatible*(*$Cond_i$*,*$Cond_j$*).*

Definition 6.3 (Contained Sources) *Let* C *be a meta-wrapper class. Let* S_i, S_j, *be sources. Let* cd_i, cd_j, *be the following CDs in WHODL:*

(cd_i*: equivalent* C *to* S_i *for* Att_i *when* $Cond_i$*), and (*cd_j*: equivalent* C *to* S_j *for* Att_j *when* $Cond_j$*).*

Then ($SE_{i,j}$*: contains* C*,* S_i*,* S_j *for* $Att_{i,j}$ *when* $Cond_i$*), is an element of SE if and only if*

Implies*(*$Cond_i$*,*$Cond_j$*).*

Definition 6.4 (Compatible) *Compatible($Cond_i$,$Cond_j$) if and only if:*

- $Cond_i = Cond_j$ *or*
- **Implies**($Cond_i$,$Cond_j$) *and* **Evaluable**($Cond_j$, S_i). *or*
- $COND$=**Intersect**($Cond_i$,$Cond_j$) *and* **Evaluable**($COND$,S_i) *and* **Evaluable**($COND$,S_j) *or*
- $Cond_{i,k} \subseteq Cond_i$ *and* **Compatible**($Cond_{i,k}$,$Cond_j$) *and* **Compatible**(($Cond_i$ - $Cond_{i,k}$),$Cond_j$).

Definition 6.5 (Evaluable) **Evaluable**($Cond_j$,S_i) *if and only if:*

- *there is a definition in MWSD for each* Att_k *in source* S_i *and the meta-wrapper supports the select operator in* Att_i.
- *the value for* Att_i *can be obtained from MWSD (defined to be a constant).*

Implies and **Intersect** are defined in the normal manner over a lattice. See [26] for details.

6.2 The Algorithm to Obtain a Source Plan

Definition 6.6 (Source Plan) *A WHOQL source plan SP is defined as follows:*

- *SP is a sequence of subqueries* (Ssq_1, Ssq_2, $\cdots$, Ssq_n), *separated by the connective* ,
- Ssq_i *is a set of parallel/union subqueries* ($Ssq_{i,1} \cup Ssq_{i,2} \cup \cdots \cup Ssq_{i,m}$), *separated by* $\cup$
- $Ssq_{i,j}$ *is a singular subquery* ($Ssq_{i,j,1}$) *or a set of alternative singular subqueries* ($Ssq_{i,j,1}$ $|Ssq_{i,j,2}$ $| \cdots |$ $Ssq_{i,j,t}$), *separated by* $|$
- $Ssq_{i,j,k}$ *is a source singular subquery* ($Ssq_{i,j,k,1}$) *or a sequence of source singular subqueries* ($Ssq_{i,j,k,1}$, $Ssq_{i,j,k,2}$ $\cdots$, $Ssq_{i,j,k,q}$), *separated by* ,
- $Ssq_{i,j,k,l}$ *is the WHOQL conjunctor,* **X**:$\{L_{i,j,k,l}\}$. **X** *identifies a particular source (source subquery), or the meta-wrapper (meta-wrapper subquery), or a catalog (catalog subquery) where* $Ssq_{i,j,k,l}$ *is evaluated.* $L_{i,j,k,l}$ *is a sequence of WHOQL literals.*

The algorithm *SourcePlan* (Figure 11), rewrites the WHOQL meta-wrapper plan as a WHOQL source plan SP. The body of the meta-wrapper plan, is a sequence of WHOQL conjunctors, C_i where each C_i is $(L_{i,1}, L_{i,2})$, $L_{i,1}$ is a singular literal $Class_p(T_1, \cdots, T_v)$ and $L_{i,2}$ is a set of literals $L_{i,2,t}$ of the form $X\theta Y$, where one variable is bound to an attribute of $Class_p$. Part of $L_{i,2,t}$ may correspond to *meta-wrapper subqueries* in the source plan.

For each C_i, from the corresponding entry in SE for $Class_p$, there may be several elements $SE_{p,j}$ of the form ($SS_{p,j}$, $Att_{p,j}$, $Cond_{p,j}$,**equivalent**). For each of these elements, we check that the meta-wrapper plan satisfies $Cond_{p,j}$, i.e., this element of SE can be used to identify alternate plans. The algorithm also ensures that all attributes of $Class_p$ occurring in the conjunctor C_i of the meta-wrapper plan are included in this element $SE_{p,j}$. If these two conditions are satisfied, this element $SE_{p,j}$ can be used to generate a safe plan.

Algorithm **SourcePlan**($MWPB, MWSD, MWOI, SA, SE$)
For each conjunctor C_i : $(L_{i,1}, L_{i,2})$ in $MWPB$, $L_{i,1}$=$Class_p(T_1, \cdots, T_v)$ and $(Class_p, SE_p)$ in SE
For each $SE_{p,j} = (SS_{p,j}, Att_{p,j}, Cond_{p,j}$,**equivalent**) in SE_p such that
($Satisfiable(MWPB, Cond_{p,j})$ and $(Att_{p,j} \supseteq attribute(Class_p)))$
For each element $SS_{p,j,k}$ in $SS_{p,j}$
If $Available(S_{p,j,k}, SA)$
$Ssq_{i,j,k} \leftarrow Transform(C_i, MWSD, SS_{p,j,k}, Cond_{p,j}, MWOI)$
$Ssq_i \leftarrow AlternatePlan(Ssq_i, SE, MWPB)$

Figure 11: Algorithm SourcePlan

The selected element $SE_{p,j}$ is a set of elements $SS_{p,j,k}$, where $SS_{p,j,k}$ is a single source or a combination of sources. The algorithm checks that the sources in $SS_{p,j,k}$ are available in SA. At this step, the algorithm can use $SS_{p,j,k}$ to produce a sequence of subqueries $Ssq_{i,j,k,l}$ of the source plan.

For each $SS_{p,j,k}$, *SourcePlan* uses an algorithm *Transform*, [26], to produce each $Ssq_{i,j,k,l}$. Each $Ssq_{i,j,k,l}$ may be a source subquery, a catalog subquery, or a meta-wrapper subquery. *Transform* must check the following for each element of $SS_{p,j,k}$:

- All attributes of $Class_p$ that occur in C_i are defined in $SS_{p,j,k}$ in the corresponding definition in MWSD.
- All the meta-wrapper operators in C_i are implemented, as indicated in MWOI.

Finally, *SourcePlan* uses an algorithm *AlternatePlan* **for each conjunctor** C_i**, and** Ssq_i. Ssq_i is the parallel source subplan for this conjunctor.

- *AlternatePlan* uses equivalences in SE labelled with the keyword **contains**, to eliminate redundancy in each Ssq_i. It removes $Ssq_{i,j,l,t}$ from Ssq_i if the following holds:

  $Ssq_{i,m,k,p}$=$Source_a$:$\{subquery_a\}$ and
  $Ssq_{i,j,l,t}$=$Source_b$:$\{subquery_b\}$ and
  $Source_b \subset Source_a$.

- *AlternatePlan* uses the combined cost-model to find the least-cost source subplan for Ssq_i.

The algorithm *SourcePlan* will fail to produce a safe source plan SP *if it cannot find a safe plan for each conjunctor* C_i *of MWP*, i.e., it will fail if it cannot find a plan for each subquery $Ssq_{i,j,k}$. If algorithm *SourcePlan* fails, the meta-wrapper query is aborted.

7 Conclusions

We present solutions to the challenges introduced by scaling mediator/wrapper architectures to hundreds of sources in a dynamic environment. We propose a source independent meta-wrapper component, SInMWrap, within a mediator component. It will accept queries in a dynamic environment without complete knowledge of the available sources. Using information on available sources, and equivalences among the content descriptions of sources, SInMWrap will eliminate redundancy and produce alternate source plans. Using a meta-wrapper combined cost-model for networked heterogeneous sources, SInMWrap will choose a least-cost plan for execution.

In our current prototype, the SInMWrap algorithms are being implemented in Java. We have built several Java and JavaScript wrappers for a variety of Internet sources; the sources return HTML documents as answers. We expect to incorporate the meta-wrapper component with the DISCO mediator [25]; in the DISCO protocol, the wrappers return Java objects as answers. The meta-wrapper combined cost model, and the query feedback algorithms for modifying the MDT structure, has been implemented in Java. We are currently conducting experimental studies of the combined cost-model.

References

[1] Adali, S. *et al* . "Query Caching and Optimization in Distributed Mediator Systems." *Proceedings of the ACM Sigmod Conference*, 1996.

[2] Ahmed, R. *et al* . "The Pegasus Heterogeneous Multidatabase System." *IEEE Comp.,24(12)*, 1991.

[3] Arens, Y. *et al* . "Retrieving and Integrating Data From Multiple Information Sources." *Proc. of the Intl. Journal of Intelligent and Cooperative Information Systems, 2(2)*, 1993.

[4] Batini, C. *et al* . "A Comparative Analysis of Methodologies for Database Schema Integration." *ACM Computing Surveys*, Volume 18, Number 4, pages 323-364, December 1986.

[5] Blakeley, J. "Data Access for the Masses through OLE DB", *Proceedings of the ACM Sigmod International Conference*, 1996.

[6] P. Buneman and L. Raschid and J. Ullman. "Mediator Languages – a Proposal for a Standard." To appear in the *ACM SIGMOD Record, March 1997.*

[7] Carey, M. *et al* . "Towards Heterogeneous Multimedia Information Systems: the Garlic Approach." *Technical Report, IBM Almaden Research*, 1995.

[8] Cattell, R.G.G. *et al* . *The Object Database Standard - ODMG 93, 1.2.* Morgan Kaufmann, 1996.

[9] Du W. *et al* . "Query Optimization in a Heterogeneous DBMS." *Proceedings of the Very Large Data Bases Conference* 1992.

[10] Florescu, D. *et al* . "Answering Queries Using OQL View Expressions." Presented at the *Workshop on Materialized Views: Techniques and Applications,* in conjunction with ACM SIGMOD, 1996.

[11] Florescu, D., *et al* . "A Methodology for Query Reformulation in CIS using Semantic Knowledge." *Intl. Journal of Intelligent and Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems,* 1996.

[12] Gardarin, G. *et al* . "IRO-DB: A Distributed System Federating Object and Relational Databases." In *Object-Oriented Multidatabase Systems : A solution for Advanced Applications,* Bukhres, O. and Elmagarmid,A, editors, Prentice Hall, 1996.

[13] Kent, W. "Solving Domain Mismatch and Schema Mismatch Problems with an Object-Oriented Database Programming Language." *Proceedings of the VLDB Conference*, 1991.

[14] Kim, W. *et al* . "On Resolving Schematic Heterogeneity in Multidatabase Systems." *Distributed and Parallel Databases, 1(3)*, 1993.

[15] Krishnamurthy, R. *et al* . "Language Features for Interoperability of Databases with Schematic Discrepancies." *Proceedings of the ACM SIGMOD Conference*, 1991.

[16] Levy, A.Y. *et al* . "Answering queries using views." Proceedings of the *ACM Symp. on Principles of Database Systems,* pages 95-104, 1995.

[17] Levy A.Y. *et al* ., "Querying Heterogeneous Infomation Sources Using Source Descriptions." *Proceedings of the VLDB Conference*, 1996.

[18] Mullen, J. *et al* . "InterBase*: A Multidatabase System." In *Object-Oriented Multidatabase Systems : A solution for Advanced Applications,* Bukhres, O. and Elmagarmid,A, editors, Prentice Hall, 1996.

[19] Papakonstantinou, Y. *et al* . "A Query Translation Scheme for Rapid Implementation of Wrappers." Proceedings of the *Intl. Conference on DOOD*, 1996.

[20] Papakonstantinou, Y. *et al* . "Capabilities-Based Query Rewriting in Mediator Systems." Proceedings of the *Intl. Conference on Parallel and Distributed Information Systems,* 1996.

[21] Qian, X., "Query Folding." Proc. of the *Intl. Conference on Extended Database Technology,* 1996.

[22] Raschid, L. and Chang, Y., "Interoperable Query Processing from Object to Relational Schemas Based on a Parameterized Canonical Representation." *Intl. Journal of Intelligent and Cooperative Information Systems,* 1995.

[23] M. Rusinkiewicz, et al, "Query processing in a heterogeneous multidatabase environment." *Proc. of the IEEE Symposium on Parallel and Distributed Processing,* 1989.

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

[25] Tomasic, A. *et al* . "Scaling Heterogeneous Databases and the Design of Disco." Proc. of the *Intl. Conf. on Distributed Computing Systems*, 1996.

[26] Vidal, M., "SInWrap: An Architecture for a Source Independent Meta-Wrapper." *Dissertation proposal, Universidad Simón Bolívar, Department of Computer Science,* 1997.

[27] Wiederhold, G., "Mediators in the Architecture of Future Information Systems." *IEEE Computer*, pages 38-49, March, 1992.

[28] Ioannidis Y. and Poosala V., "Histogram-Based Solutions to Diverse Database Estimation Problems." *Data Engineering Bulletin*, volume 18, Sep, 1995.

[29] Selinger P. G. *et al* . "Access Path Selection in a Relational Database Management System." *Proceedings of the ACM SIGMOD Conference*, 1979.

[30] "Cost Estimator based on Parameterized Feedback." *In preparation*, 1997.