Information Exchange Among Collaborating Enterprises

David Levermore[1], Gilbert Babin[2], and Cheng Hsu[1]

[1] Rensselaer Polytechnic Institute,
Decision Sciences and Engineering Systems, Troy, NY, USA
`leverd@rpi.edu, hsuc@rpi.edu`
[2] HEC Montréal, Information Technologies, Montréal, Québec, Canada
`Gilbert.Babin@hec.ca`

Abstract. On-demand business/service and other emerging models for service enterprise integration all involve on-demand information exchange. The best practices in the field tend to rely on homogeneous semantics, which is difficult to achieve for independent databases owned by independent enterprises. Otherwise, the exchange tends to be limited at the level of file transfers. To overcome these limits and achieve on-demand pull of information resources at the global database query level, we develop a new information exchange model to extend previous global query results and cover independent databases. The new model provides a four-schema architecture to allow enterprises to offer the information that they wish to share with others (query for users), as well as request what they want (query for data). A central element is information matching; which has to unify the representation and processing of both offering and requesting queries, and integrate them with traditional global database query results. We develop such a new information matching model in this paper. It employs a unique "query database" approach, an "export database" design, a "four-schema" architecture, and new matching algorithms that promise computing efficiency to achieve the purpose. The new method also allows for inclusion of constraints (rules) in the matching for information exchange.

1 On-Demand Enterprise Collaboration

A challenging problem in service enterprise integration is to drill through supply chains to coordinate schedules on a real time and online basis. To do this, trading partners need to go beyond the traditional "fixed" cooperation and develop on-demand information exchange for flexible collaboration. An example of fixed cooperation between partners is the effort that a retailer (e.g., Wal-Mart) links its demand forecasting databases with its manufacturer's production scheduling systems (e.g., Warner-Lambert) to shorten the replenishment cycle for certain products. This example is not new, but the collaboration still faces daunting technical limits. First, the information exchange was hard-wired rather than being on-demand; and second, the inter-operation mechanism was not easy to extend to other likely participants in the supply chain. In principle, these two problems can be facilitated by global database query results such as federated databases; however, these results run into problems, too, because they require the participating companies to open up their databases for direct inter-operation. Furthermore, the trading partners have a many-to-many relationship among them; therefore, even if the prime (e.g., Wal-Mart) could impose a single authority on the entire supply chain, the members still have to reconcile this particular Wal-Mart standard with their other primes or buyers (i.e., other federations).

The above example shows that supply chain integration is ***on-demand collaboration*** in nature, characterized by independent databases that control when and what information to offer and to whom, as well as issue queries for information. These databases may also project different images/personalities onto different global models in their inter-operation with other systems. There are other examples of independent databases, such as the participants in Homeland Security and industrial exchanges. The traditional global query assumptions do not cover them.

In this context, business process integration must rely on flexible information exchange mechanisms between systems. Then again, these exchange mechanisms must be able to evolve as local processes change and collaboration requirements increase. In this paper, we propose the Two-Stage Collaboration Model (TSCM) as a framework that support such flexible and evolvable information exchange mechanisms. The TSCM architecture and methods enable an enterprise to "offer" information it is willing to share with partners or to "request" information it needs to perform its internal processes. Information matching comes naturally when we realize that the unique characteristics of independent databases are actually their ability to make information offering bids to many potential users, as well as issue information requesting bids (with or without a pricing mechanism). As such, the TSCM constitutes an information service infrastructure, as it provides a matching service between offers of and requests for data. On-demand information exchange at the database level, as described herein, is an extension of the database query (SQL) capabilities that companies have come to expect of all enterprise databases. Queries provide ad hoc information for decision support, and hence complement the application level collaboration, and enhance/support service level collaboration.

The TSCM integrates market-style information matching with research results on global query processing. In a general sense, research has shown that market-style self-allocation of users/providers [4] is a promising solution approach that supports the collaborative paradigm of global database query. However, previous results of artificial markets (e.g., Covisint and CommerceOne [5, 6]) that support collaboration do not include global database query; and market style global query [12] has not afforded on-demand offering. More to the point of this paper, previous matching methods are focused on matching bids based on price and product definition, rather than matching database queries and views based on information semantics. While the matching on price could be academically intriguing [11], the basic matching on product is straightforward in practice since product definition can be standardized for particular commercial practice. Information semantics (including rules), on the other hand, is inherently more difficult to match. To solve the matching problem, one needs a common representation method to define the requests and offerings, to lodge the active information bids, and to match them according to their rules and data semantics. In this work, we employ a metadata representation method developed for multiple databases integration [9] as the basis for developing these new results. The basic logic is that both information requests and offerings are formulated as global queries (metadata) against the collaborating community, using the method as the query language. These queries are stored in a database (the query database) whose structure is also based on the metadata representation method, and hence is consistent with the language. The query database manifests the universe of community collaboration at any given time. Each request or offer is both a runtime database query against the query database and a persistent object to be saved (updated) in the query database. Therefore, the matching is performed directly when newly created queries are processed as ordinary database operations against the query database. Search algorithms determine the cases for complete matches and various partial matches, and take into account the participants' conditions.

The core of the paper is the new matching method: the metadata language, the query database, and the search algorithms. However, to properly present these results, we briefly review in Section 2 the context of the new matching method: a complete solution approach to the global query problem under the collaborative paradigm. The new results are briefly presented in Section 3. The last section, Section 4, concludes the paper with a summary of the status of this ongoing research.

2 The Information Architecture for the Collaboration

The overall information architecture of on-demand enterprise collaboration features a **four-schema** concept shown in Figure 1. which supports the following **Two-Stage Collaboration Model**. Local schemata (first schema) represent enterprise data as stored in their local databases. Export databases are used to store data that the enterprise wants to make available to partners. These export databases are described using a publication query, which acts as a schema of the data made available (second schema). Subscriptions describe data that is required by partners (third schema). All these are represented in the Blackboard which extends the Metadatabase (fourth schema). The Two-Stage Collaboration Model includes matching for collaborators and fetching of the required information, for on-demand enterprise collaboration. The technical problem involved is referred to as the participatory database query problem in [9]. The first stage matches information offerings with requests in a market style, and establishes optimal participation of databases for a search task. The second stage, then, executes the task in a traditional distributed database query manner. The model

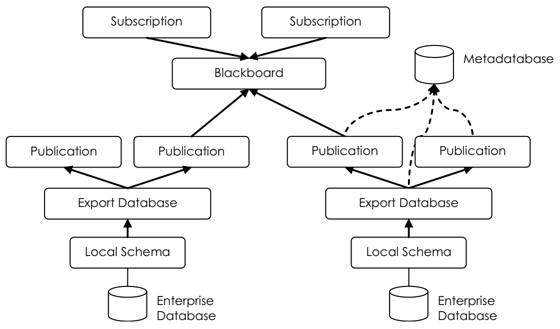

Fig. 1. Four-schema Architecture of the Two-Stage Collaboration Model

uses a global blackboard, but also distributes the blackboard to local sites as mini-blackboards to allow for peer-to-peer implementation. Requirements to the new model include that, at the first stage, it reconciles the different data semantics at the participating databases to allow determination of the "right" matches, and provides massively concurrent bidding, matching, and, if price is involved, auction/negotiation that lead to self-allocation of databases for particular tasks. The second stage needs to perform concurrent global queries for each task, according to its conditions and constraints, and, if necessary, join partial results from various sources.

The query database approach developed for matching integrates the usual market functions of bidding, matching, and auction/negotiation with global query processing. In this approach, the usually custom-designed **blackboard** of a market that conducts these market functions becomes an ordinary database management system, and can be implemented by using off-the-shelve DBMS technology. To implement this concept, the **query database** requires an integrated representation of data (information contents) and rules (task conditions and constraints) since the queries involve both. This requirement goes beyond traditional query processing. As a response, this research develops the dedicated databases collaboration language that we call exMQL (extended Metadatabase Query Language) to formulate the queries (and execute them). The semantics and syntax of exMQL are rooted in the Metadatabase model [8], which readily provides the schema for the query database — i.e., the queries are represented in and stored as semantic constructs of the Metadatabase. The exMQL is a metadata language based on SQL; thus, exMQL expressions are processed by the DBMS of the query database using the PL/SQL facility. The blackboard algorithms are also constructed in PL/SQL. The use of the PL/SQL facility affords the flexibility of adopting matching-negotiation-auction results from elsewhere. Comparing to concepts adopted in the literature, the query serves the purpose of a software agent on the market, and the query database satisfies the goals of the previous agent community.

Alongside with the query database, a **Metadatabase** (a repository of metadata structured as a database) is created to include and integrate local data models (for the export databases, or the views that participants publish) into metadata tuples and tables, with the help of a registration process. As shown in the literature [1–3, 8], metadata entries (data models) are added, deleted, updated, and retrieved as ordinary database operations against the Metadatabase without interrupting the continuous operation of the market. Since the Metadatabase is also implemented using a standard DBMS, it allows for high intensity and concurrent updates. The Metadatabase facilitates the reconciliation of data semantics for query processing (matching) at the query database. A formal registration process with varying degrees of central coordination is required of all participants, through a CASE (computer-aided software engineering) tool. In the maximum registration regime, the Metadatabase integrates all published local views; while in a minimum regime, the Metadatabase contains only global equivalence information. The local sites are responsible for maintaining a global version of their respective database views in the minimum regime. In any case, the participants could opt to register a large, stable data model within which they publish their smaller, *ad hoc* offerings as often as they wish; or, they could frequently register different, small local data models that represent their published views and change them on the fly. The most demanding part of the registration is the establishment of a list of global data items across local sites, and the maintenance of the data equivalence among them. This is not an easy task and could easily become the bottleneck to any real-time, peer-to-peer collaboration. We stress, however, that this is a common problem in the field and the databases collaboration model does not add to the burden; if anything, the new model could ease the problem because the participants are now committing their proxies rather than production databases to a standard medium. We also wish to point out that the Metadatabase method affords a list that is automatically derived from all registrations, and can reveal any peer-to-peer correspondence of local data items. In this sense, the Metadatabase serves as an open common schema for the community.

The second stage, global query processing, is essentially an extension of the traditional database query against the export databases. The differences reside mainly in the *ad hoc* conditions and constraints under which the queries are processed; since the information requests and offerings may include additional requirements. The Two-Stage Collaboration model addresses this issue by extending established results from the Metadatabase research, including GQS (global query systems) [3], ROPE (rule-based programming environment) [1], and rulebase processing methods [2]. The new result becomes an integrated query processing and management system for the blackboard. The exMQL-based queries map into SQL expressions to be processed at the proxy servers of the local sites, which may inter-operate with the local enterprise databases depending on the local policies. Results are assembled at the blackboard according to the particular cases of joint implied by the original tasks.

The local sites are connected to the global site and to each other through a **proxy server** which is constructed and maintained according to a global design but is otherwise administered by the local site at which it resides. In addition to the export databases, the proxy server also replicates the blackboard, along with the capacity to maintain a query database and Metadatabase, into its functions for any local site that requires peer-to-peer information exchange capabilities. The proxy server, then, has the ability to initiate a virtual circle of matching among peers and serve as the "global" site of the circle during the life of the match. In this way, the global blackboard co-exists with many concurrent, distributed local blackboards and their virtual sub-communities of information exchange. This design promises to reduce the possible computing bottlenecks in the community, enhance overall system reliability (against system or network failures), and support maximum flexibility of the collaboration regime. The proxy server may reduce its requirement on a full Metadatabase when peer-to-peer metadata interfacing or interchanging is included. Similar to the minimum regime of registration, a minimum set of metadata at proxy servers includes only the list of global data equivalents. A partial Metadatabase ranges between the minimum and maximum contents. A continuing challenge facing this model is the maintenance of the distributed copies of global equivalents. It represents an upper bound to the real-time performance of peer-to-peer collaboration.

3 The Information Matching Method

We briefly present below the new query database method developed in this paper for the first stage, to provide the matching for information exchange. The metadata language that defines queries for the matching also defines the queries for final global query processing at the local sites. In a similar way, the query processing algorithms that perform the matching also extend to include functions pertaining to the second stage, the fetching of information for users.

3.1 The Query Language: exMQL

The Extended Metadatabase Query Language (exMQL) is designed to provide a uniform query format for the various query operations that are required in the Two-Stage Collaboration. The structure is derived from the original MQL specification in [3], which in turn is based on the TSER representation method [10] and the GIRD metadata model [8]. However, it extends significantly from the original MQL to support information publication and the inclusion of constraints/conditions in the queries. The full specification of exMQL is provided in Figure 2. Each query operation is performed via the extended Metadatabase Global Query System (exMGQS) and so the query specification described below is provided for illustrative purposes only.

3.2 The Query Database Schema

The conceptual structure of the Blackboard in general is based on a derivative of the GIRD, the integrated representation of the Metadatabase [8]. The requirements of the GIRD are relaxed for the purposes of the Blackboard, and so a

```
<QUERY> ::= <COMMAND> <ITEMS> [’DO’ <ACTIONS>]* [’FOR’ <CONDITIONS>]* ;
<ITEMS> ::= ITEM [, ITEM ... ] ;
<COMMAND> ::= ’GET’ | ’PUT’ ;
<CONDITIONS> ::= <CONDITION> <conjoin> <CONDITION> ;
<conjoin> ::= ’AND’ | ’OR’ ;
<CONDITION> ::= <SC> | <JC> | <NC> ;
<SC> ::= ITEM <bound> VALUE ;
<JC> ::= ITEM <bound> ITEM ;
<NC> ::= ATTRIBUTE <bound> VALUE ;
<bound> ::= ’<>’ | ’=’ | ’<’ | ’>’ | ’< =’ | ’>=’ ;
<ACTIONS> ::= action [, action ... ] ;
<DELETE_QUERY> ::= ’DELETE’ query_name [’CASCADE’] ;
<DELETE_RULE> ::= ’DELETE’ rule_name [, rule_name ...] ’IN’ query_name ;
<DELETE_CONDITION> ::= ’DELETE’ condition_name [, condition_name ...] ’IN’ query_name ;
<UPDATE_QUERY> ::= ’UPDATE’ <ITEMS> [’DO’ <ACTIONS>]* [’FOR’ <CONDITIONS>]* ’IN’ query_name ;
```

Fig. 2. The Syntax of exMQL

number of the elements of the GIRD are unused but are not removed in the information model. We will highlight the changed features in any discussion of the Blackboard. The intent of the Blackboard is no different from the Metadatabase, in that it is used for the management of metadata, specifically the schema of the queries. The new structural model of the Blackboard is illustrated in Figure 3 using the TSER modeling methodology [10], and we discuss the conceptual structure of the query database and rulebase in the sub-sections that follow.

The Structural Model of the Query Database. The addition of the SYSTEM, QUERY, and VIEW meta-entities, which replace the APPLICATION, SUBJECT and ENTREL meta-entities, are the primary differences between the structural model of the Blackboard and the GIRD. The remaining meta-entities, and meta-relationships: Functional Relationships (FR), Mandatory Relationships (MR) and Plural Relationships (PR) that connect them, retain their original definitions as outlined in [2, 8]. We briefly illustrate the most relevant elements of the query database in the paragraphs below.

The SYSTEM meta-entity identifies the proxy database servers that are currently participating in global query, and so represents a dynamic view of the Two-Stage Collaboration System. Each proxy database server is defined by a unique identifier, which is determined at runtime. However, this is not provided to the Blackboard unless a query has been submitted by the proxy database server.

The QUERY meta-entity identifies the queries submitted by proxy database servers. Every query is unique, and each proxy database server can submit more than one query at any time. A timestamp attribute is generated automatically by the Blackboard and associated with the query, to facilitate the measures utilized during the query assignment process. The **components** meta-MR identifies the queries that belong to each SYSTEM.

The VIEW meta-entity provides a mechanism to associate data items with multiple queries, in various arrangements.

The ITEM meta-entity remains unchanged from its original definition, and represents the data items specified in each query. The **belongto** meta-PR rep-

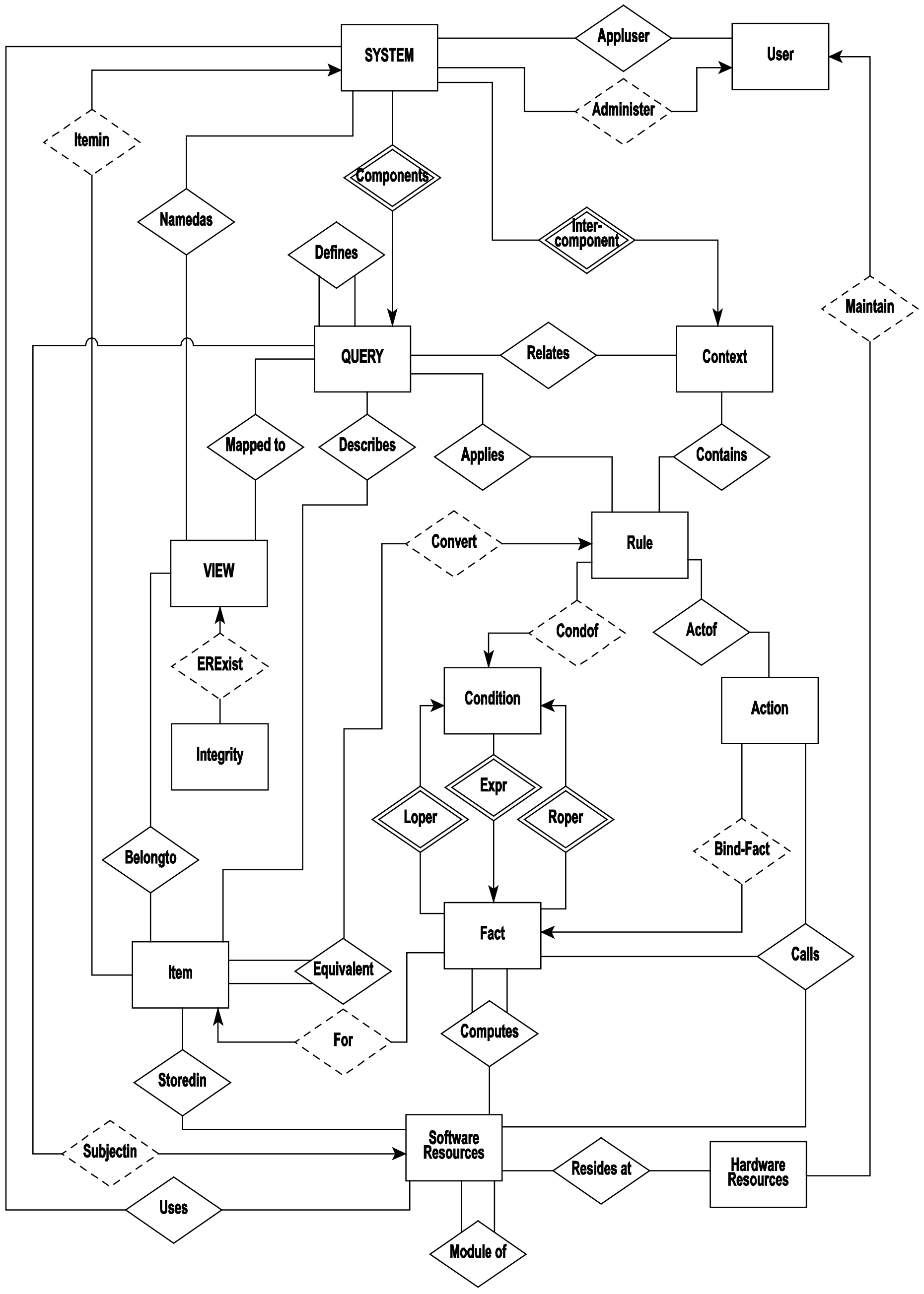

Fig. 3. Conceptual Structure of the Blackboard

resents the assignments of items to a specific VIEW. The **describes** meta-PR specifies the data items that are aggregated into each QUERY.

The Structural Model of the Rulebase. The rulebase inherits the rule modeling paradigm of the GIRD as illustrated in Figure 3. The RULE, CONDITION, and ACTION meta-entities, and their associated meta-PR's, meta-FR's and meta-MR's. retain their original functionality, however they now must provide for the management of data objects as opposed to metadata objects. Specifically, rulebase facilitates the modeling of all constraints (conditions) and associated actions.

3.3 Query Processing

Following the formulation of a query, the participant submits the query for processing. This processing involves three distinct phases: (1) match, (2) assign, and (3) execute.

Search for Matching Queries. In the match phase we identify queries (which we denote as target queries) that contain any number of data items specified in the submitted query (which we will denote as the source query). Note that either type of query (source or target) can represent a information request or offer. The search process for a given query (S) can produce four classes of results; (1) an *exact* match, (2) a *closest subset* match, (3) a *subset* match, and (4) a *closest* match. Exact match is not illustrated.

Definition 1. *In an* exact match*, the number of data items, the syntax, and the semantics of the data items (in other words, the items that match) in each target query are equivalent to those in the source query.*

Definition 2. *A* closest subset match *indicates that the number of items matched is less than that specified in the source query but equivalent to those defined in a target query.*

Definition 3. *A* subset match *indicates that the number of items matched is equivalent to the number of items in the source query, but less than that the number of items specified in the target query.*

Definition 4. *A* closest match *is the case where queries contain common items. That is the number of items matched is less than the number of items specified in both the source query and the target query.*

As a first step, we take a source query as input and compare it with a set of target queries. For each target query that has common items with the source query, we count the number of data items that intersect and classify these according to the aforementioned definitions. The result of this step is the set of queries that match the source query.

Permute Queries to Identify a Feasible Solution. If we are unable to identify target queries that contain all the data items in the source query, then we enter into an additional step of processing. Here, we permute the results found (most likely closest subset and closest) and evaluate each of the resulting permuted queries to determine if they contain the data items found in the source query. The resulting permuted queries are classified as, (1) *permuted exact* match, (2) *permuted closest subset* match, (2) *permuted subset* match, and (4) *permuted closest* match, where each, respectively, is the analogue of the four classes posited above.

A permuted exact or permuted subset match does not indicate that a solution for the source query has been found; rather, only that the disparate combined queries contain data items common to the source query. For the permuted query to be a feasible solution, the disparate queries that constitute the permuted query must be logically connected. We illustrate this problem with the following example:

Example 1

$$\boxed{\begin{array}{l} S = \{item_1, item_2, item_3, item_4\} \\ T_A = \{item_1, item_2, item_n, item_{n+1}\} \\ T_B = \{item_2, item_3\} \\ T_C = \{item_4, item_m, item_{m+1}\} \end{array}}$$

Let S, T_A, T_B, and T_C correspond to the source query and target queries illustrated in Example 1. Obviously, T_A, T_B, and T_C match S on particular data items. The union of T_A, T_B, and T_C, which we denote as T_{ABC} is the permuted query that contains all data items in the source query; however this is not necessarily a feasible solution. If T_{ABC} is to be a feasible solution for S then there must be logical relationships among T_A, T_B, and T_C. The Metadatabase is employed to evaluate the permuted queries. In this case, the Metadatabase will reveal that T_A contains the primary key $item_1$. Accordingly, we find that there exists a transitive dependency between T_A and T_B, as demonstrated below, but this will only be true if $item_2$ is the primary key for T_B. Since, $T_A : item_1 \rightarrow item_2, item_n, item_{n+1}$, and assuming $T_B : item_2 \rightarrow item_3$ then, by Armstrong's inference rules (transitivity), $item_1 \rightarrow item_3$. Therefore, T_A and T_B are logically connected. If $item_2$ was not the primary key for T_B then we could not postulate this result.

By consulting the Metadatabase we find that the attributes of T_C are dependent on the primary key $item_0$. Even if a logical relationship exists between both T_A and T_B, we cannot determine T_{ABC} since $item_0$ is not included in T_C. This leads us to the following definition that each permuted query must respect to be considered a feasible solution for a source query.

Definition 5. *To be considered a* feasible solution*, a permuted query must be logically connected, and all logically connected data items must exist in the target queries.*

Example 2 illustrates the permutations of the three target queries, T_A, T_B, and T_C. It is safe to ignore Round 1, since these correspond to the singular target queries.

Example 2

Round	**Permutations**
1	T_A, T_B, T_C
2	T_{AB}, T_{AC}, T_{BC}
3	T_{ABC}

The feasibility of a permuted query is determined by the Shortest Path algorithm [3], which determines if the entities and relationships (which in the Metadatabase are referred to as oe/pr, respectively) in a query are connected. However, since permuted queries do not contain oe/pr's, then these must be determined as well. The result of this second step is the permuted query (or queries) that contain the greatest number of data items common to a source query.

Evaluation of Constraints. A successful match process is also contingent on the agreement of preferences that have been associated with queries. Each query can be optionally qualified with constraints that restrict or limit a query to specific values, or restrict or limit the query in general to particular system-defined variables. These constraints are manifested as rules, adhering to the Event-Condition-Action paradigm [1] where each rule is processed subsequent to an event occurring, which will execute an associated action if the evaluation of the related condition returns true. In general, all events correspond to a successful query match.

As a last step, we evaluate the rules associated with the query from the steps above. If multiple target queries are determined, we retrieve all rules and combine them into a single fact table. We then iterate through each condition in the source query, and tests the associated facts with those of the target query (or queries) in the fact table. If we identify a matching attribute in the fact table, we evaluate the condition, i.e., comparing the values in the input condition with the output condition. If a match is **not** found, we then mark the specified conditions for additional processing. We repeat this process until all conditions in the source query are exhausted and return the set of conditions that do not match.

Assign Queries. After a match (or matches) has been found, then we enter the assignment phase where the source query is assigned for processing. The basic task is to determine the proper space of query processing, i.e., what subscribers and what publishers are to participate in this particular global query processing. It is trivial if there is only one match (exact or joint) to be assigned for processing; but it becomes complicated when multiple matches must be considered together. There are a number of criteria by which these ties may be broken if the "competing" queries are equivalent in every respect, i.e., they are of the same class of match, and the conditions that match the source query are equivalent.

A basic, and default, criterion we consider in this research is to permit multiple subscriptions of a single publication, and the single subscription of the multiple publications. That is, we would execute all equivalent multiple matches and present all the results for the users to interpret. We call this the **maximum collaboration** principle, which makes sense for information exchange (but not for merchandise trading) since information can be shared without loss by multiple users and fused freely from multiple sources. However, if this principle is not true for whatever reasons, then either a round of auction/negotiation could be conducted or the system could apply some automated decision rules. The selection of these criteria is essentially a matter of system design.

The list below provides a few basic functions that facilitate the choice of automatic tie breaking among multiple matches. The users could specify their preference of any of these choices during query construction as actions of a query. Note that multiple actions can be specified for a single query.

Simple heuristics – uses the system-defined timestamp of each query to select a winner. The query with the oldest/latest timestamp, i.e., the query that has been registered with the Blackboard for the greatest/least amount of time is selected.

Network Performance – considers the geographical location of the associated proxy database server and the load of the server, assuming that further distances and loads contribute to performance latency, and chooses the match that is in closer proximity to its location.

Participant History – the proxy database server that has reliably provided solutions for many participants and received good ratings will be chosen over a proxy database server that has not been similarly prolific.

User Preference – the winner will be chosen from a list of attributes of the proxy database servers that have been specified during query construction, such as the locality, organization, and other indicators of the site that the users prefer.

Execute. The execution of the query on the publication databases constitutes the final phase of query processing. It is necessary however, to perform some data conversion and translation on the queries before they can be processed on the publication databases. Briefly, all data processing that exists beyond the boundaries of the publication databases are defined in global terms, and so queries must be converted to local values before they can be processed on a proxy database server. Conversely, when the publication query is created the parameters are converted to global values. In general, the subscription query is always created in global values, since the exMGQS provide the interface of the global information model for the construction of information requests.

The conversion algorithm employs the global equivalence functionality of the Metadatabase to (1) identify equivalent data items in the query, and (2) convert to the data items to the local/global value depending on the perspective. In that event, a unit conversion is required, which the Metadatabase allows for; the algorithm in concert with the Metadatabase will address this as necessary.

4 Beyond the new matching method

We might submit that the global database query capabilities are a key to achieving service enterprise integration under virtually all conditions. The problem is only that how much results the field affords. We develop a new approach to the problem and thereby extend the previous results to support independent databases, which are a basic characteristic of on-demand enterprise collaboration. A new enterprise integration architecture is developed for the approach, which includes the two-stage collaboration model and the four-schema design, as well as the new information matching method.

The query language and processing algorithms span both the matching (the first) stage and the global query processing (the second) stage of databases collaboration, and hence provide the integration of the two for the final system. The algorithms constitute a major part of the Blackboard. The new matching method is currently being tested in a laboratory setting for the purpose of performance evaluation as well as for the verification of the Two-Stage Collaboration model. Beyond the matching method itself, and also beyond the technical verification of the complete solution, the ongoing research is seeking to improve the registration methods and the maintenance of global equivalents, as a means to enhance its efficiency in practice. In this direction, the research also investigates novel ways of defining and managing global equivalents. The goal is to eventually allow proxy servers to discover data equivalents from direct data or metadata interfacing with peers (that certain data items from peers are in fact equivalent to their own), on an on-the-fly basis. This way, local sites would be able to add data items without having to update the global list — i.e., without much delay. This field remains challenging and widely open to new ideas, despite the new results obtained [7].

References

1. G. Babin and C. Hsu. Decomposition of knowledge for concurrent processing. *IEEE Transactions on Knowledge and Data Engineering*, 8(5):758–772, 1996.
2. M. Bouziane and C. Hsu. A rulebase management system using conceptual modeling. *J. Artificial Intelligence Tools*, 6(1):37–61, March 1997.
3. W. Cheung and C. Hsu. The model-assisted global query system for multiple databases in distributed enterprises. *ACM Transactions on Information Systems*, 14(4):421–470, 1996.
4. S.H. Clearwater, editor. *Market-Based Control : A Paradigm for Distributed Resource Allocation*. World Scientific Publishing, River Edge, N.J., 1996.
5. Covisint. http//www.covisint.com.
6. R. Glushko, J. Tenenbaum, and B. Meltzer. An XML framework for agent-based e-commerce. *Communications of the ACM*, 42(3):106–114, 1999.
7. A.Y. Halevy, Z.G. Ives, J. Madhavan, and P. Mork. The piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):787–798, July 2004.
8. C. Hsu, M. Bouziane, L. Rattner, and L. Yee. Information resources management in heterogeneous distributed environments: A metadatabase approach. *IEEE Transactions on Software Engineering*, 17(6):604–625, 1991.

9. C. Hsu, C.D. Carothers, and D.M. Levermore. A market mechanism for participatory database query: A first step of enterprise resources self-allocation. *Information Technology and Management*, forthcoming.
10. C. Hsu, Y. Tao, M. Bouziane, and G. Babin. Paradigm translation in manufacturing information using a meta-model: The TSER approach. *Ingénierie des systèmes d'information*, 1(3):325–352, January 1993.
11. T. Sandholm. Making markets and democracy work: A story of incentives and computing. In *IJCAI-03*, 2003. Computers and Thought Award Talk Abstract.
12. M. Stonebraker, P. Aoki, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide area distributed database system. *International Journal on Very Large Databases*, 5(1):48–63, 1996.