9 DESIGN OF THE FEDERATED INFORMATION MANAGEMENT ARCHITECTURE FOR PRODNET

H. Afsarmanesh♦, C. Garita, Y. Ugur, A. Frenkel, L.O. Hertzberger
University of Amsterdam, The Netherlands

In ESPRIT project PRODNET II, a federated architecture approach is adopted to support the information management requirements of the VE environment. This approach provides an innovative mechanism for information access rights and visibility level among enterprises, which are exploited through the federated query processing. In this chapter, the design issues of the federated information management components are represented.

INTRODUCTION

Enterprises involved in a VE must share and exchange a part of their information in order to jointly achieve the common VE goal, and to support each others functionalities. However, not all members of a VE play the same role and/or need or can request and acquire the same access level to the local information of other enterprises. It is clear that among competitive enterprises in a VE the amount of trust is limited, and that every enterprise needs to precisely define the specific access rights and visibility levels on its information for every other VE partner. As a result, within the VE, support for the security of shared data and provision of different rights to access shared data -based on other enterprise's role in the VE- are required to be provided and reinforced. For instance, partners involved in the supply chain may require to exchange data about the product, while the VE coordinator enterprise needs to investigate the accurate progress of work and job activities within the VE and at several individual partner sites.

Therefore, for the information management within a VE network, due to the competitiveness of pre-existing enterprises and their proprietary information, it is not realistic to assume that a single global schema defines all the information that is visible and exchanged by all partners. The approach for VE information

♦ Corresponding author address: University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands, tel.: +31-20-5257512, fax: +31-20-5257490, e-mail: hamideh@wins.uva.nl

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35577-1_37

L. M. Camarinha-Matos et al. (eds.), *Infrastructures for Virtual Enterprises*

management introduced in the DIMS (Distributed Information Management System) of PRODNET provides an innovative mechanism for defining both the group rights and the individual rights through the creation of export schema hierarchies on the VE interoperation layer database schema. The approach is based on the federated/distributed database principles, as it is detailed in (Afsarmanesh, 1999). A general description of the PRODNET reference architecture, the PCL (PRODNET Cooperation Layer), and the components of PCL are detailed in (Camarinha-Matos, 1999) and (Afsarmanesh, 1997). Figure 1 represents the PRODNET architecture. In the next sections, some design details of the federated information management approach adopted for PRODNET is presented.

The structure of this chapter is organized as follows. First, the DIMS federated

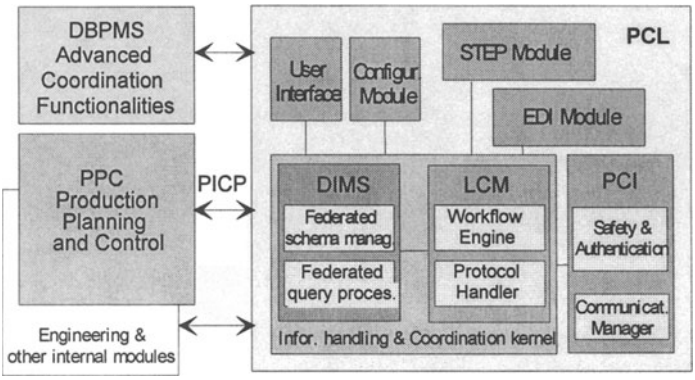

Figure 1 - Description of the PRODNET general architecture

schema is described, including a brief analysis of its components, as well as the general justification for applying a federated approach and some insight into related research areas and projects. Secondly, the mechanism for defining export schemas for other VE partners in order to define fine-grained visibility levels on local information is reported. After this, the DIMS federated query processing is also documented in details. Finally, some conclusions about the general federated design are drawn at the end of the chapter.

THE FEDERATED INTEGRATED SCHEMA OF DIMS

In this section, the DIMS classification of PRODNET information is addressed. Further, the federated schema for DIMS is described in details.

The complex set of requirements for PRODNET distributed information management can be supported through a federated database architecture. Analyzing the wide variety of shared and exchanged information among VE member nodes, and achieving a comprehensive description and classification of this information, are the main goals tackled within PRODNET. For this analysis, a step-wise approach is followed as summarized below:

1. Definition and study of several distinct focus areas. First, we divided the study domain into three focus areas that represent the main kinds of interactions and

exchange of information between different elements of the PRODNET architecture. The three focus areas consist of: a) the exchange of information within the PCL, needed by the PCL components, in order to operate properly and provide the expected functionality; b) the exchange of information between the Internal Module of an enterprise and its PCL cooperation layer; and c) the exchange of information between every two nodes in the virtual enterprise. The detailed study of the PRODNET application domain through the three focus areas mentioned above, has identified both a wide variety of types of information that need to be handled at every node, and a large set of requirements for information exchange among the nodes.

2. Identification of the local, imported, and exported information. As a first step towards modeling this diverse information for every enterprise, we divided the enterprise information into three categories of *Self, Acquaintance,* and *Virtual Enterprise* information with corresponding intuitive meanings. The details of this classification approach and their further division into private, restricted and public information, are outside the scope of this chapter and is covered in (Afsarmanesh, 1999). However, a summary of the steps taken in this study, and some results are enumerated below:
    2.1. Determine which part of the information is generated and stored in this enterprise, that thus becomes **local** (i.e. partially the *Self* and partially the local part of the *Virtual Enterprise* information).
    2.2. Determine which part of the information needs to be accessed from the other nodes, that thus needs to be **imported** from other enterprises (i.e. partially the *Acquaintance* and partially some other enterprises information related to the *Virtual Enterprise*).
    2.3. Determine which part of the local information needs to be shared with other enterprises, that thus needs to be **exported** to other enterprises (partially the *Self* information and partially some local part of the *Virtual Enterprise* information). However, the visibility levels of the enterprise information from external nodes must be carefully determined by every node in order to ensure its own autonomy and information privacy.

In order to support these specific information management requirements, and the proper interoperation among PCLs in different nodes, a federated database architecture is designed for the DIMS module in PRODNET (Afsarmanesh, 1997a), (Camarinha-Matos, 1997).

The DIMS Federated Approach

The DIMS federated architecture approach has proven to adequately facilitate and support the sharing of information between enterprises in VEs, while providing the necessary information visibility levels to ensure their own autonomy and information privacy. The designed federated architecture for DIMS has its roots in the PEER federated database system, developed at the University of Amsterdam (Wiedijk, 1996), (Afsarmanesh, 1994), (Afsarmanesh, 1993), (Tuinman, 1993). The PEER system, in short, is a fully federated, object-oriented information management system, which supports the sharing and exchange of information among cooperating

autonomous and heterogeneous nodes, without the centralization and data-redundancy. A survey of other related work and architectures for federated database systems is described in (Sheth, 1990). Some approaches to distributed information management in the context of VE support platforms can be found in other related projects such as the NIIIP (NIIIP, 1996) and VEGA (Zarli, 1997).

In NIIIP, the information is uniformly modeled in an object-oriented framework. A number of object-oriented local conceptual schemas are defined; one for every individual VE member organization. The local schemas together with the definition of global VE resources, form the enterprise-wide global conceptual schema. The VE global conceptual schema is a mediated global schema rather than a predefined integrated schema, where some conflicts are resolved at the schema integration time. However, the detailed specification of NIIIP data management services is still under the development and so far there is still not enough information available.

In VEGA, the Distributed Information System (DIS) is supported by a "CORBA access to STEP" (COAST) layer. Through the CORBA architecture the physical distribution of the documents is handled. The DIS works with a single data model: the STEP data model. Therefore, documents like EDI orders, SGML documents and any other document if it can be modeled by the STEP language specification, can be stored in the DIS. The DIS is a distributed database and in comparison to the federated approach chosen for PRODNET, the concepts of export, import, and integrated schemas, and the support for information security and hierarchy of information visibility and access levels, are not directly addressed in this system architecture.

The design of DIMS in PRODNET consists of the definition of PCL integrated schema that is represented and handled in all nodes. Data can be imported/exported through this PCL schema, but the proper access rights are defined locally at every enterprise to precisely specify the rights of external nodes. Therefore, although the sharable data at every node can be accessed through the same schema structure representing the PCL information (as illustrated in Figure 2), it properly preserves the federated information access and visibility constraints by means of well-determined export schema definitions. For example, a given member of the VE_1 can access specific information of another node within VE_1, but not the information that the other node stores regarding its participation in other VEs. Furthermore, the information stored in a node about a particular VE is not made available with the same visibility level to all VE members; namely in one VE, the visibility can be defined differently for every node. But, what is important to notice is that regardless of these considerations, the PCL schema itself remains the same along the network of cooperating nodes. As a result of this design decision, the nodes do not need to physically import the schema descriptions from each other, in order to identify the *structure* of the data available in other nodes. Consequently, this design decision has simplified the set up of the PRODNET system for involved enterprises.

The DIMS component of PRODNET has been developed using C++. The underlying database manager system used to support the DIMS functionality is Oracle, and the low-level database operations were implemented using the Oracle OCI (Oracle Call Interface) and ODBC driver functions (through the MFC database facilities). The DIMS interface tools have been developed using GUI support of Microsoft Visual C++.

Design of the PCL Integrated Schema

The federated architecture of the DIMS handles the **local**, **import** and **export** schemas for the enterprise, while supporting the expected data location transparency for the user, the site autonomy, the access security, (associated to distributed query processing) and the reliability among other requirements. However, every enterprise needs to have access to both its local information and the information imported from other enterprises, and thus the two parts of information need to be **integrated** into one coherent schema for the sake of enterprise's convenience of access and retrieval of information. Then clearly, always a part of this integrated schema represents the local schema (of which a part is also exported), and the other part represents some imported schemas at the node.

Based on the initial analysis and identification of the wide variety of information described in the previous section, the ***integrated*** *object-oriented database schema* for enterprise information has been designed, and is depicted in Figure 2.

The integrated object-oriented database schema in Figure 2 represents a unified generalization hierarchy (subtype/supertype) and the association relationships among different type (class) definitions, while representing only partial high-level categories of information handled within the PCL, for simplicity reasons. Here, the DIMS schema diagram is described in the style of the Object Modeling Technique (OMT) notation (Rumbaugh, 1991). The connection used for this diagram is depicted at the bottom corner of Figure 2.

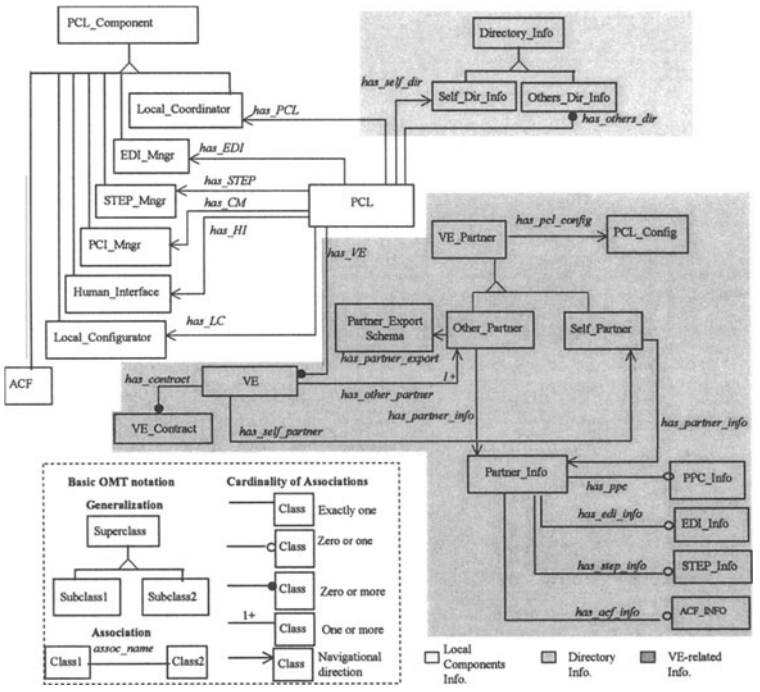

Figure 2 – Partial (high-level) integrated schema for the PCL of an enterprise

The class that represents the integrating focal point in this schema is the PCL class (appearing in the middle of this schema in Figure 2). The comprehensive PCL information is basically composed of three major parts:

- Local PCL components' information (represented on the top left side of this schema diagram)
- VE-related information (represented at the bottom right side of this schema diagram)
- Directory information (represented at the top right side of this schema diagram).

Local PCL components information. All the information that different components of the PCL need to handle in order to function properly and perform their activities is embodied in the PCL_Component class. These components consist of, for instance, the Local_Coordinator, the EDI_Mngr, STEP_Mngr, Comm_Mngr, the Human_Interface, the Local_Configurator, and the ACF module. Every one of these PCL components is represented as one "class" in Figure 2, where clearly, all the component's information is defined as extensions to this class.

VE-related information. A given PCL node can be involved in any number of different VEs. The VE information (VE class in Figure 2) encompasses the information about the structure for this involvement, such as their members' composition, identification of the VE coordinator, etc. as its attributes. Furthermore, it covers the detailed information about the partners of the VEs. For instance, within each VE, for the enterprise itself, a set of local enterprise information (e.g. PPC) is managed and stored at the PCL. This information encompasses different kinds of data at the enterprise that can be potentially shared with the other partners. All this information is kept locally for every enterprise. The information of other partners is captured through association defined between the Other_Partner class and Partner_Info (has_partner_info association). Please notice that this information is not physically stored locally at this node for other partners, rather this information is distributed over the corresponding other-partners' nodes. In other words, this part of information will be **imported** from other partners when there is a federated query at this node that needs to access it and if the node is authorized to access it. Now, from the point of view of the users and the applications issuing such queries, this association between Other_Partner and Partner_Info is treated as any other local relationship that is also represented in the integrated schema. This means that the DIMS provides transparency on data location, that supports the queries on other nodes' local information through the distributed/federated query processing on the integrated schema at every node. It is important to notice that with the design of DIMS integrated schema structure for the nodes, the schema for this node's Partner_Info and other nodes' Partner_Info is the same. This design reflects the fact that the schema for what a company makes available to others, is in turn the same as the schema of the information that is imported from those companies.

The Other_Partner class is also associated with the Partner_Export_Schema. This class provides the facility to make the Partner_Info partially available to be ***exported*** to other partners. The Partner_Export_Schema class is introduced as a mechanism to support these different levels of visibility to local information for every other partner in the VE. Through this concept an export schema for every

other partner is defined. The details of the export schema definition mechanisms is described later in this chapter.

Directory information. The descriptive information that every enterprise is willing to make available to public, regardless of cooperating with them or not, is represented by the directory information (Directory_Info) in PCL schema for both the enterprise itself (Self_Dir_Info) and for other enterprises (Others_Dir_Info), in which this enterprise is interested. The directory information can include a general description of the products, services, and activities of the company.

VISIBILITY LEVELS AND EXPORT SCHEMA MANAGEMENT

In this section, the information visibility levels in virtual enterprises are addressed and then an approach based on export database schemas to support them is introduced.

The Concept of Visibility Levels in VEs

Since different enterprises must have different visibility levels and access rights to other nodes' information, every node in the federation must decide what part of its local information to make available to every other partner in every particular VE that it is involved. In other words, the level of visibility and access that other partners have on the local PCL schema of a given node, must be clearly determined. To accomplish this objective, every node can protect its autonomy and privacy by defining one detailed *individual export schema* based on its PCL local schema, for every other node with which it shares information. Every export schema defines the corresponding access right of other node to this node's local data (see also views defined on objects in (Abiteboul, 1991)). For every class defined in the local schema, the individual export schema determines which instances (i.e. horizontal class partitioning) and which attributes (i.e. vertical class partitioning) of those instances will be made available to every other node. Through this mechanism, different granularity can be preserved, from the value of one specific attribute of a determined class instance, to all PCL information accessible from another node.

Although the approach of individual export schema definition on the local schema for every external "user", is the basic idea, we have generalized this basic idea to the definition of a complete *hierarchy of export schemas.* This hierarchy allows the grouping and classification of common export schema characteristics, facilitating the task of individual export schema definitions.

To explain a situation where this hierarchy becomes highly valuable and convenient for the VE creator and configurator, let us assume that for a VE there are three different kinds of roles that a given enterprise can play: the *coordinator*, *supervisor* (subordinated to coordinator but enabled to monitor certain VE activities), and *regular* VE partner. Clearly, for every role, different information items must be made accessible from other nodes. For example, a VE coordinator needs to know information, which for another VE partner may even be a secret. The support for a fine-grained visibility level mechanism is required to model this situation.

Using the concept of hierarchical export schema definition, different visibility levels can be properly defined for every partner. Figure 3 represents an example of a general export schema hierarchy definition. At the first level of visibility it is needed to extract the data that corresponds to every given VE from the PCL local schema. Namely, a horizontal partitioning of information that chooses all objects related to one VE is achieved. Further, a second level of visibility is defined for every VE export schema, through a vertical partitioning that supports the proper information access rights for the different groups of users. For instance, the VE coordinator will obviously have more access visibility to partners' data than a simple regular VE partner due to its inherent control, monitoring, and possibly auditing responsibilities. Similarly, a supervisor in charge of the proper accomplishment of a specific subtask inside a VE, will have more access visibility to partners' information than other regular VE partners, but certainly more limited than the coordinator. However, all these three kinds of export schemas are defined on top of the initial export schema (for a VE) as the base. Furthermore, at the third level of visibility, the definition of the export schema, even at the level of partners of the same VE, can still be different for every particular partner, since a node needs to exchange different information with every other VE member. Clearly, when further layers for visibility levels become necessary to define, the export schema hierarchy can be extended and modified.

Please notice that the export schema definitions on the PCL schema, represent the derivations from that PCL's local schema in the federated database architecture of the DIMS. Also remember that no schemas need to be physically imported due to the fact that the schema structure is the same for both the exported and imported information at every PCL, but not all the "information" is available to every given node, thanks to the visibility levels defined by their export schema definitions.

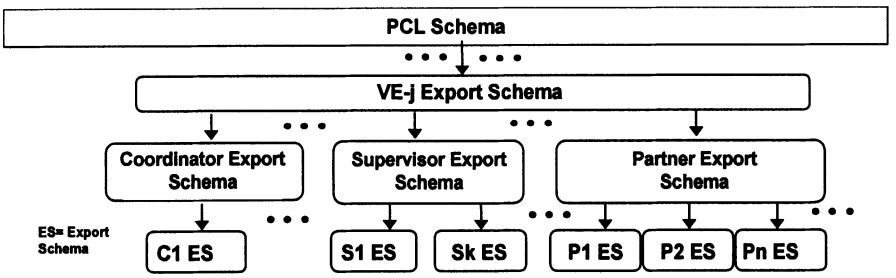

Figure 3 - Example of Export Schema hierarchy definition on PCL information

Design of VE Export Views

In Figure 4 the design of the database schema related with export schema management is presented. Through this schema, the recursive definition of elements of the export schemas hierarchy is supported. For every VE partner, an external schema set (Export_Set) is defined, which corresponds to the partner's export schema. Through the Export_Set, the proper visibility levels for the partners on the local schema of the enterprise are specified. An external export set can be either a

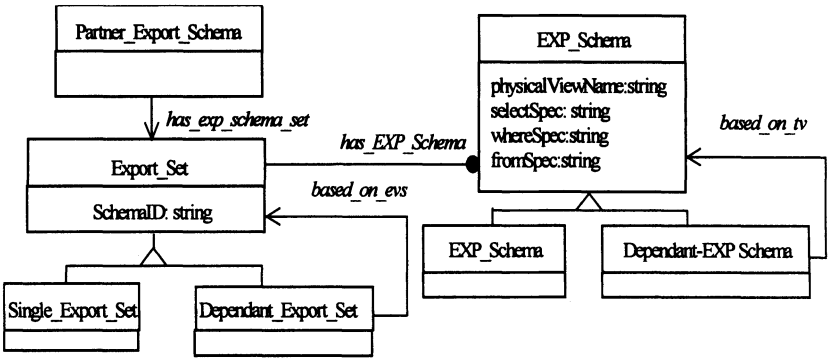

Figure 4 - Schema definitions for partner export schema's management

single or a dependent export set, as depicted in Figure 4. Single export sets are not based on other export sets. A dependent export set however, is defined in terms of another partner export schema, based on the export schema hierarchy concept introduced previously in this section. With this approach, on one hand, support for the general export schemas definition is provided, where not only the pre-defined export schema definitions at the level of VE, coordinators, supervisors, and partners, are considered, but also other hierarchies can be defined and supported as necessary. On the other hand, a partner export schema consists of a set of schemas, which in turn can be single schema (EXP) or dependent schema (Dependent-EXP) following this definition strategy. Every EXP schema has as attributes: the identification of the physical schema definition for the exact table (e.g. an Oracle view name), a list containing the attributes selected from that table (selectSpec), a condition on the specified attributes (whereSpec), and the original table on which the schema is defined (fromSpec). If a schema is a Dependent-EXP schema, then the fromSpec refers automatically to the corresponding EXP schema.

To operate on the described schema, a "Export Schema Manager" (ESM) application is developed. The export schema manager is used to create a basic PCL export schema, and then, to define dependent PCL partner export schemas based on it. The ESM will ensure that the export schema hierarchy remains consistent, and that the schema definitions for every dependent partner export schema are properly created. In general, the ESM will be responsible for accomplishing the following specific operations (among others):

- EXP schema creation based on the actual database table
- Dependent-EXP schema creation based on an EXP schema
- Dependent-EXP schema creation based on another Dependent-EXP schema
- PCL export schema creation based on a set of EXP or Dependent-EXP schemas.

The ESM incorporates advanced user interface graphic elements to achieve a comprehensive, easy-to-use, and friendly application, as a part of the PRODNET Cooperation Layer human interface. An ESM designed and implemented this way will highly facilitate the fine-grained access level definitions at each node for every VE partner.

Methodology for definition of VE Export Schemas

In this section, a specific methodology for the definition of the VE Export Schema is

```
class CLASS_DECLSPEC Export_Manager {
  int Create_Schema(char *schemaname, char *selectspec, char *wherespec, char
  *fromspec);
  int Create_Compound_Schema(char *schemaname, char *selectspec, char *wherespec,
  char *fromspec);
  int Create_BaseOn_Compound_Schema(char *schemaname, char *selectspec, char
  *wherespec, char *fromspec);
  int Create_PCL_Schema(int schemas_id[50], char *s_role, char *ve_id, char
  *enterpriseid);
  int Add_Schema(char *schemaname,char *selectspec, char *wherespec, char
  *fromspec, int type_dep, int base);
  int Create_Schema_Instance(char *schemaname,char *selectspec, char *wherespec,
  char *fromspec, int type_dep, int base);
  int Check_SchemaID(int schema_id);
  int Add_PCL_Schema(int schemas_id[50], char *role, char *veid, char *enterpriseid,
  int type_dep, int base);
}
```

Figure 5 - Class definition for Export Schema Manager

presented. The Export Schema, that will define visibility levels and access rights to other nodes' information, consists internally of a set of export tables (EXP) and dependent export tables (Dependent-EXP). The EXP refers to an export schema defined on one database table, and the Dependent-EXP is a schema defined on an EXP. Such schemas need to be defined in several steps:

1. Create EXP schema on every database table. The EXPs determine which instances (i.e. horizontal partitioning) will be for instance, made available for the Coordinator level, in every other node. In this case, it is necessary to define which database tables will be accessible by another node.
2. Define one or more Dependent-EXPs for every EXP table defined above. The Dependent-EXPs determine which attributes (i.e. vertical partitioning) will be for instance, made available for the Supervisor level, in every other node.
3. Define one or more Dependent-EXPs for every Dependent-EXP defined above. This operation is equivalent to the previous one, except that the reference is another Dependent-EXP. The Dependent-EXPs determine which attributes (i.e. vertical partitioning) will be for instance, made available for the Enterprise level, in every other node.
4. Define the EXP/Dependent-EXP Set, a set of EXP and Dependent-EXPs that will specify the proper visibility level.
5. Define the Role EXP Schema hierarchy, a complete hierarchy of the roles that a given node can play in a VE, based on the for instance three basic kinds of roles: coordinator, supervisor, and regular partner. Each role will be associated with one EXP/Dependent-EXP Set defined above, so that every defined role is able only to access predefined information. The advantage to define the role

hierarchy and associate every role with a EXP/Dependent-EXP Set is that then it is easier to define different visibility levels for every actual VE partner.

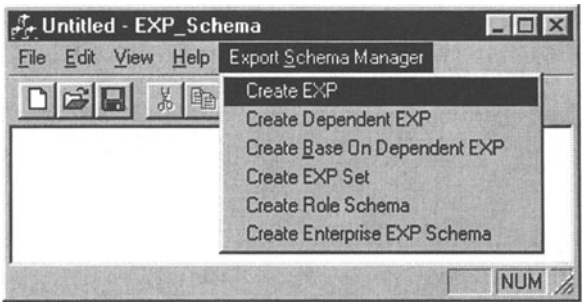

Figure 6 - Export Schema Manager Interface

6. Create the export schema for an enterprise, this will associate an enterprise with one of the Role EXP Schema defined above, so that every node will be able to access different information according to role it plays in the VE.

Following this methodology, every VE will be able to create and define the export schema in an easy and systematic way.

ESMT - VE Export Schema Manager Tool

The ESMT (Export Schema Manager Tool) developed for DIMS supports the definition and creation of the export schemas as part of the local information that is shared with other enterprises. It helps the human operator of PCL to define and create the export schemas, during the configuration phase of the VE. In Figure 5, the definition of the class Export_Schema_Manager is presented. In this class the operations mentioned above are supported. There is a basic operation, Create_Schema_Instance that creates an EXP or a Dependent-EXP, according to the parameters received from the functions Create Schema, Create Compound Schema, Create BaseOn Compound Schema. The function Create PCL Schema creates the export schema set.

The main window of the ESMT interface tool contains a menu bar that enables the user to perform different operations. The main functionalities are accessed through the Export Schema menu displayed in Figure 6. The Export Schema menu supports the definition of different schemas. There is also a status window where control messages generated during the creation of the export schema are displayed. The error and warning messages generated during the process are displayed as a Visual C++ message box.

The main actions associated to the ESM menu are described below:

- The Create EXP Window (Figure 7) supports the creation of the EXPs. The user has to enter a unique name for the EXP name. Then, he/she has to specify the attributes that define the export schema: the original table on which the schema is based on, the list of attributes selected from the table, and the conditions on

the specified attributes. In that way the user is restricting the records the other partners can access.

- The Create EXP/Dependent-EXP set Window (Figure 8) supports the creation of the export schema set. This window displays the EXP/Dependent-EXP hierarchy at the top left. It represents the specific instance diagram for the export schema hierarchy, it shows how EXP schemas are defined for each database table, and how the Dependent-EXP schemas are based on the EXP schemas. At the top right, it displays the EXP/Dependent EXP Schema. This "edit box" shows how the EXP Set is taking form according to what the user has selected or removed from the hierarchy. Finally, at the bottom of the window the SQL description of the chosen database table, EXP schema or Dependent-EXP schema is showed.
- The Create Enterprise EXP Schema Window supports the definition of the export schema for an enterprise. It helps to associate a specific enterprise (Enterprise ID) with a specific Role (Enterprise Role ID). The Schema Specifications shows the Schema Set, the EXP schemas and EXP-Dependent

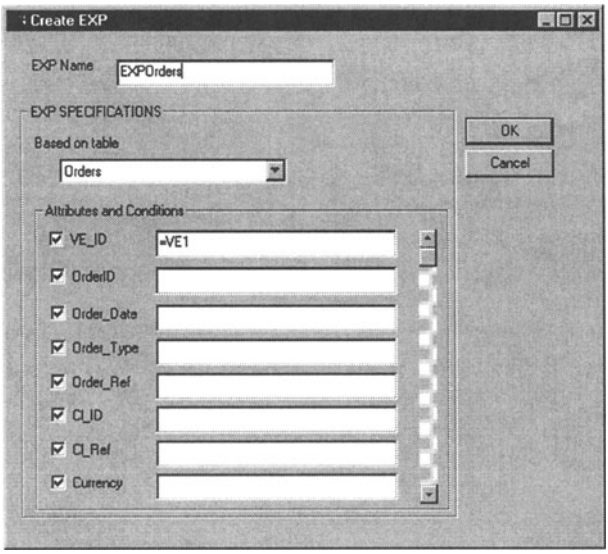

Figure 7 - Create EXP

schemas that the particular enterprise will be able to access, and the SQL description of the chosen EXP schema or Dependent-EXP schema.

Examples of VE Export Schemas

In this section, some specific instance diagram examples for the Export Manager classes are presented in Figure 9. Two database tables (Order and Product) are shown, with first a single EXP schemas (tv1 and tv2) for each table, containing only the information about orders and products related to the VE identified by "ve12". The single EXP schema construction establishes the first visibility level, to filter

only this information. Then, these schemas are associated with a single export schema set instance called "pv1", which represents the general export schema for ve12. Based on this VE export schema, a VE coordinator export schema is defined. To do this, two dependent table schemas are defined (tv3 and tv4), and associated to a dependent export schema set pv2. Please notice that the definition of the dependent table schemas hide those fields that were present in the previous visibility level, but that for example are not made available to the VE coordinator.

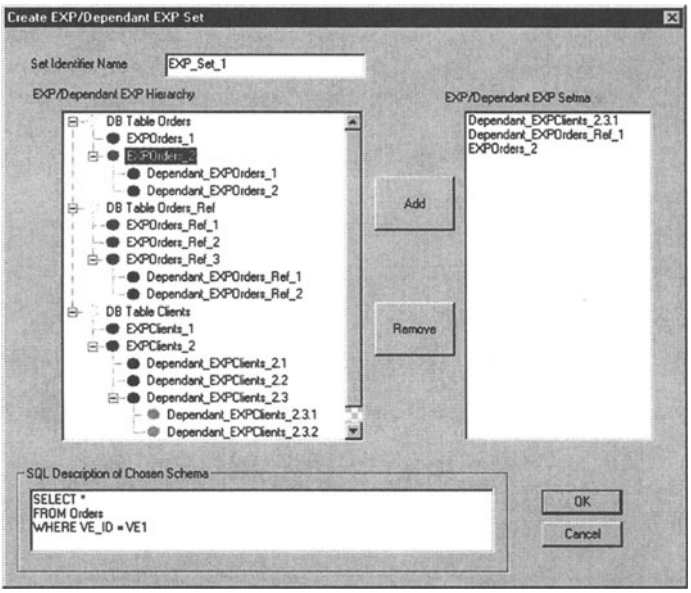

Figure 8 - Create EXP/Dependent-EXP Set

Another example is provided in the same figure, and in this case first a general partner export schema (tv5,tv6) is defined based on the general VE export schema (tv1,tv2). In this case the fields which can be made accessible to any given partner are provided (in this example the order price information is removed in tv5 and tv6 definitions). On top of this generic partner export schema, a specific partner export schema is defined for partner 1 in ve12. For partner 1, only the information related to the partner (clientId=1) is made accessible through tv7 and tv8. The specific partner export schema definitions provide examples of how the export schema hierarchy can be defined to support a flexible visibility level definition approach for every partner, based on its function in the VE (coordinator,regular partner, supervisor, etc.).

DIMS FEDERATED QUERY PROCESSING

The goals of general federated query processing are tailored and extended in DIMS considering the PRODNET architecture and requirements. The main objective is to

support federated access to authorized data, which can be distributed over the nodes of the VE network. In other words, the federated query processing functionality of DIMS enables end users, for instance, the VE Coordinator to query the privileged proprietary VE related information for which the coordinator is authorized, while hiding the data location details. Furthermore, considering that the information in internal modules (e.g. PPC) of the enterprises are updated independently, the federated query processor in the DIMS is also responsible for retrieving the most up-to-date generated data, if it is necessary. This necessity is determined depending on the needs of the query requester.

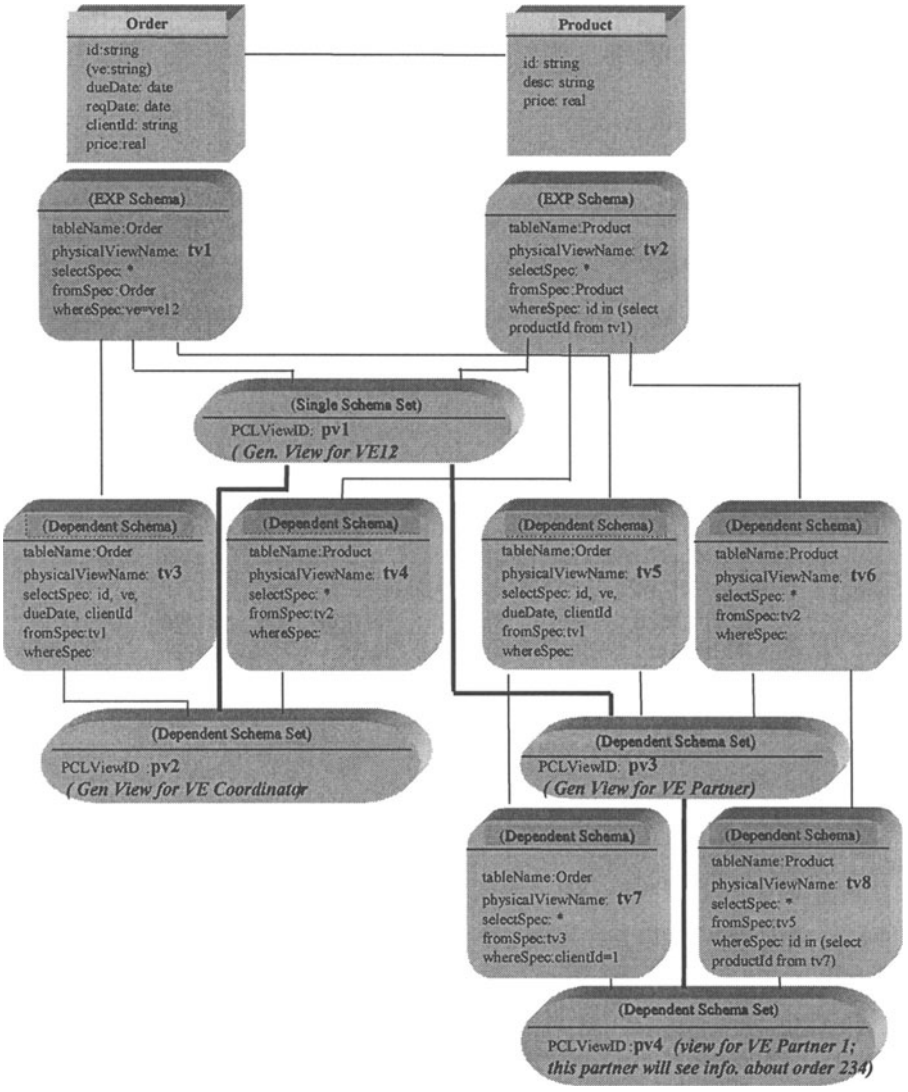

Figure 9 – Instance diagram of view hierarchy for regular VE partners

In the general case, two kinds of federated queries may arrive at a DIMS of an enterprise, an *internal query* (a query arriving from the PPC or other PCL modules in the node, such as DBPMS) or an *external query* (a query arriving from another node's DIMS). Furthermore, there are two forms of internal queries that need to be handled. The internal query can be a *local query,* which will be answered without acquiring any information from other nodes, or a *distributed query*, which involves retrieving of information from one or a specific set of other nodes. In this way, the distributed query is processed against the integrated schema, through which local and imported data is seamlessly accessed from any higher-level application interacting with the DIMS.

There are two main components of the DIMS query processor that handle all these types of queries: the *local query component* and the *federated query component.* The local query component is a high level interface on top of the PCL local database for the PCL modules and the PPC, when they are accessing and manipulating their VE-related local information to support the VE operation. Some examples to the high level functions that are provided for PCL modules and PPC are: the Get/Put of purchase-order for EDI module, Get/Update of publickey/privatekey for PCI module and Update of production orders for PPC.

The federated query component is the one which exploits the imported/exported data definition between enterprises to support the secure access to the federated information, which is defined in the integrated schema, for every VE (e.g. asking requested order information in VE by the coordinator module). In order to identify if the requested data is privileged for this user, this component must work together with the VE Export Schema Manager described earlier in this chapter. We will focus on the federated query component in the rest of this section.

At present, there are two main usages foreseen for the federated query component in VE paradigm. One usage is generated by Advanced Coordination modules, that requires to access information from many enterprises. The second usage is generated by the human operator of the enterprise. For PRODNET the following usages are foreseen :

1. The coordinator enterprise in the VE needs to ask distributed queries to its DIMS to gather the information distributed over the nodes, so that it can monitor the status and performance of VE nodes. DIMS supports a set of *high-level functions* based on PCL interface standards for module interoperation, so that the coordinator does not need to worry about low level database access mechanism/language details. Then high-level functions will be transformed by DIMS for instance to some stored procedure calls or to SQL queries using the parameters specified in the functions.
2. Through the Human Interface , the end user at the enterprise, may also want to ask for distributed information.

The steps of the federated query processing will be addressed in details in the following section.

DIMS Federated Query Processing Steps

When a query arrives at the DIMS, it must first be determined if it is a local query or a distributed query. To make this distinction, the query must be analyzed in search of specifications of partners IDs corresponding to Other_Partners. In summary, a set of subqueries will be deduced from the original query (one subquery per partner), and the subqueries will be sent, and finally the received results will be merged.

The deduced subqueries, which are sent from one DIMS to another DIMS in another node, comply with a specific format to facilitate the processing at the target node. An initial subquery format is presented in Figure 10. At the first field, the query message contains a tag field (msg_type) to specify that it is actually a "query message", since there are several types of messages which can be exchanged between DIMSs. Thus, this field will contain a code to determine that the DIMS message is a query message. The second field is a unique identifier for the query inside the DIMS of query issuer. The third and fourth fields correspond to the identifications of the origin and target nodes, respectively. These fields are actually composed of the node identifier and the identifier of VE from which the information is requested. Finally, the last field is reserved for the content of the query itself. What should be noticed here is that DIMS to DIMS communication is not done directly, but rather via two other modules, the local coordinator module (LCM) and the communication interface module (PCI). This strategy is consistent with the general PCL architecture, while at the same time the PCI advanced communication functionality is exploited as explained later. The inter-DIMS messages are always embedded in PCI messages.

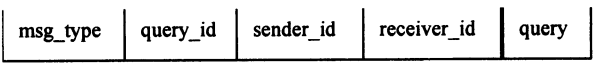

Figure 10 - Structure of an inter-DIMS query message

In summary, in all cases that an internal query is issued by a PRODNET user or application to the DIMS, the federated query processor of the DIMS involves the following simplified steps at the query issuer site.

1. Determine whether the query is local query or distributed query
2. If the query is a local query
   2.1. *If* the query issuer asks the most up-to-date PPC generated data
        Invoke the workflow to ask PPC store most recently updated data into the DIMS
        else, Evaluate the subquery and prepare the result
3. If the query is a distributed query
   3.1. Identify all the nodes (internal/external) that the subquery should be sent
   3.2. Decompose the query into subqueries at every one involving only one partner
   3.3. For every node that the subquery should be sent
      3.3.1. If the node is this node itself
           Same as step 2.1

3.3.2. If the node is external node
3.3.2.1. Prepare inter-DIMS query message including the information specifying if PPC should be asked for the most up-to-date information
3.3.2.2. Invoke the workflow to send inter-DIMS query message to every partner node containing the corresponding subquery message
3.4. If the internal PPC is asked to update data, wait for PPC to finish the task
3.4.1. When response arrives to DIMS from PPC, evaluate the subquery and prepare the result
3.5. Wait for the results of the subqueries evaluation from the external nodes
3.5.1. Interpret the inter-DIMS query answer message and extract the result of the subquery
3.6. Merge the partial results and prepare the final result
4. Return the result

When a subquery arrives from another node (an external query), the query is evaluated against the export schema defined for the query sender node. When the result is obtained, it must be returned to the sender of the query, using the node identification that is contained in the inter-DIMS message. This process is described in the following steps:

1. Interpret the inter-DIMS query message
1.1. If the query issuer asks the most up-to-date PPC generated data
Invoke the workflow to ask PPC store most recently updated data into the DIMS
else, Evaluate the subquery and prepare the result (described in details below)
2. If the internal PPC is asked to update data, wait for the PPC to finish the task
2.1. When response arrives to DIMS from PPC, evaluate the subquery and prepare the result
3. Prepare the inter-DIMS message including the result of the subquery and invoke the workflow to send the result back to the query issuer node.

The evaluation of the subquery at the external node is very crucial from the secure data access' point of view. There are number of steps that must be followed to convert the incoming subquery to a local query. As described earlier, the PCL schema definition is the same in all nodes. Therefore, any node can issue a query against its "imported" part of the schema. But clearly, the access rights of every node to the data that it can import from a second node are precisely specified in the export (individual) view defined for it in the target node. Therefore, the arriving query will be evaluated against the corresponding export schema. However, before this query evaluation step, first the query specification needs to be altered according to the particular export schema definition. For example, the class names in the individual views will definitely be different than the actual class names. Since the actual class names are the ones used by the incoming query, these names need to be replaced by the names for the corresponding local export schema. After this step, the query can be evaluated locally, and all the visibility access constraints are preserved.

msg_type	query_id	sender_id	receiver_id	result

Figure 11 - Structure of an inter-DIMS query-result message

More specifically, the sequence of steps that must be followed to evaluate a subquery is:

1. Identify the VE(s) involved in the query (from the class VE in the local PCL schema)
2. Identify the Other_Partner instance corresponding to the sender partner associated with the VE
3. Check query issuer enterprise has specific export schema definition in the VE.
   3.1. *If* it has, get the export schema associated with this partner
   else, find the role of the enterprise and get the export schema for this role
   (At this step, hierarchy of visibility levels is used to avoid the user access unprivileged data)
4. Get the names in the export schema associated with the partner for the table names in the subquery
5. Modify the incoming subquery replacing the specified table names with the table names of export schema associated with this partner
6. Evaluate the modified subquery (it is evaluated on the local database)

The format of a DIMS query-result message is represented in Figure 11. The node which answers the query, places the corresponding query_id (the same that arrived with the DIMS query message), places its ID as sender_id, places the ID of the query sender node in the field receiver_id, and appends the result of the query. The tag field will be "query answer message" in this case.

Please notice, that all the messages exchanged between PCL nodes are properly encrypted and secured by the Network Communication Layer Component of the PCL, applying algorithms such as the DES, RSA, and digital signature for message authentication. The communication protocol parameters of exchanged low-level messages are also handled by this PCL component. The Communication Component is also in charge of handling large messages, which may result from the evaluation of a query. This component determines the best way to send large amounts of information along the network, embedded in PCL messages. DIMS federated query processor relies on the communication layer of the PCL when it is passing the inter-DIMS message.

The DIMS also embodies the mechanism to handle the errors/exceptions which can occur during the query processing. For example, the DIMS timeout mechanism properly handles PPC delays when updating data, or delays in other nodes when resolving an external query. In the case of a failure in a single site or the communication links, some data will be unreachable to the end user module. However, the end users should be allowed to access partial information, which is reachable. In the DIMS federated query processing approach, the end user is informed about the incompleteness of the data and the incomplete query result is provided. Later the module or user can try to access o this unreachable part of information again.

CONCLUSIONS

The federated design of the DIMS properly supports the cooperative information sharing and exchange, node autonomy, visibility levels and access rights for exchanged data among the VE nodes. The described view hierarchy management features are used to adequately support the definition of information access rights based on the existing legal contracts among VE partner enterprises. The federated database architecture of the DIMS has been specifically tailored to handle the complex interoperability and information management functionality requirements for VEs, set by the PRODNET application and its target SMEs. The view hierarchy definition and the distributed query processing described in this paper are the novel features of the DIMS that provide for import/export of information among the federated nodes in virtual enterprises.

Acknowledgements
This work was funded in part by the European Commission, Esprit programme. The authors also thank the valuable contributions from their partners of the PRODNET II consortium: CSIN (P), HERTEN (BR), MIRALAGO (P), University of Amsterdam (NL), Universidade Federal de Santa Catarina (BR), Universidade Nova de Lisboa (P), Lichen Informatique (F), ProSTEP (Germany), Uninova (P), ESTEC (P).

REFERENCES

1. Abiteboul, S; Bonner, A. - Objects and Views, in Proc. ACM SIGMOD91, pages 238-247, May 91.
2. Afsarmanesh, H; Tuijnman, F.; Wiedijk, M.; Hertzberger, L.O.- Distributed Schema Management in a Cooperation Network of Autonomous Agents. Proc. of the 4th Int. Conf. on Database and Expert Systems Applications (DEXA'93), Lecture Notes in Computer Science 720, pages 565-576, Springer-Verlag, Sept 93.
3. Afsarmanesh, H. et al. Flexible and Dynamic Integration of Multiple Information Bases, Proceedings of the 5th IEEE Inter. Confer. on "Database and Expert Systems Applications DEXA'94", Athens, Greece, Lecture Notes in Computer Science (LNCS) 856, Springer Verlag, Sep. 1994.
4. Afsarmanesh, H; Camarinha, L. - Federated Information Management for Cooperative Information - in proc. of the 8th Int. Conf. on Database and Expert Systems Applications (DEXA'97), Sept. 97.
5. Afsarmanesh, H; Garita, C; Hertzberger, L.O.; Santos, V. - Management of Distributed Information in Virtual Enterprises : The PRODNET Approach – in proceedings of the Int. Conf. on Concurrent Enterprising (ICE'97), October 97.
6. Afsarmanesh, H.; Garita, C.; Ugur, Y., Frenkel, A.; Hertzberger, L.O. - "Federated Information Management Requirements for Virtual Enterprises". In the Pro-VE'99 Book: "Infrastructures for Virtual Enterprises" (L.M. Camarinha-Matos and H. Afsarmanesh, Editors), Kluwer Academic Publishers, 1999.
7. Camarinha-Matos, L; Afsarmanesh, H; Garita, C; Lima, C - Towards an Architecture for Virtual Enterprises; accepted in The Second World Congress On Intelligent Manufacturing Processes & Systems, Budapest, Hungary, June 10-13, 1997.
8. Camarinha-Matos, L; Afsarmanesh, H; Garita, C.; Lima, C - Towards an Architecture for Virtual Enterprises. Special issue of the journal of Intelligent Manufacturing with the focus on agent based Manufacturing, Volume 9, Number 2, Pages 189-199, Chapman and Hall publications, March 1998.

9. Camarinha-Matos, L; Afsarmanesh, H. "The PRODNET Infrastructure". In the Pro-VE'99 Book: "The PRODNET Architecture" (L.M. Camarinha-Matos and H. Afsarmanesh, Editors), Kluwer Academic Publishers, 1999.
10. NIIIP, The NIIIP Reference Architecture, 1996, http://www. niiip.org.
11. Rumbaugh J. et al - Object-Oriented Modeling and Design; Prentice Hall, 1991.
12. Sheth, A; Larson J. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACMCS, v. 22, n. 3, September 1990.
13. Tuijnman, F.; Afsarmanesh, H. - Management of shared data in federated cooperative PEER environment. Int. Journal of Intelligent and Cooperative Information Systems (IJICIS), 2(4): 451-473, Dec 1993.
14. Wiedijk, M.; Afsarmanesh, H.; Hertzberger, L.O. - Co-working and Management of Federated Information-Clusters. Proceedings of the 7th Int. Conf. on Database and Expert Systems (DEXA'96), Lecture Notes in Computer Science 1134, pp 446-455. Springer Verlag, Sept. 96.
15. Zarli, A. et al – Integrating Emerging IT Paradigms for the Virtual Enterprise: Ghe VEGA platform. In proceedings of the Int. Conf. on Concurrent Enterprising (ICE'97), Nottingham, England, October 97.