

**RESOLVING HORIZONTAL PARTITIONING AND SCHEMATIC
VARIANCES USING METADATABASE APPROACH**

By

POON Koon-hei

M a s t e r O f P h i l o s o p h y T h e s i s

Presented to

The Graduate School

In Partial Fulfillment
of the Requirements for the Degree of
MASTER OF PHILOSOPHY

FACULTY OF BUSINESS ADMINISTRATION



Resolving Horizontal Partitioning and Schematic Variances Using Metadatabase Approach

Abstract

With the proliferation of PC-base database systems and advancements in communication technologies, computing powers are typically decentralized to the front line operations in modern enterprises. This nourishes the development of heterogeneous distributed database systems. Data objects are often segmented (horizontal or vertical partitioned) or replicated over different local systems. Horizontal partitioning is often overlooked although it is classified as a common phenomenon in these systems. It induces problems when users retrieve information from horizontally partitioned data objects. Without knowing that a set of data objects is horizontal partitioned, only partial data will be resulted and rendering query results incomplete or even incorrect. The problem in information retrieval from horizontal partitioned data objects can be further complicated when structural differences such as level of abstractions and schematic variances exist among these data objects. Many database integration methodologies or approaches have been developed since the 80s'. The main objectives for these methodologies are to facilitate information sharing (e.g. global query) in the heterogeneous distributed environment. These existing methodologies are either ignored or overly simplified the horizontal partitioning problem on retrieving information from local systems. In this study, we extend a system integration methodology, the Metadatabase approach, with the capability to retrieve and integrate information from horizontal partitioned data objects. The Metadatabase approach has many favorable features such as allows local autonomy and provides adaptability while facilitating interoperability among local systems. Specifically, we characterized horizontal partitioned data objects using metadata, which are subsequently modeled and included in the existing Metadatabase. Methods, which utilize this set of new metadata, for processing global queries against horizontal partitioned data objects are developed.

香港中文大學

決策科學及管理經濟學部
工商管理學系
哲學碩士論文

以Metadatabase 方法解決平行劃分及結構差異問題

撮要 (BIG-5)

網絡通訊技術發展及以微電腦為基礎的資訊系統的普及,利於現代企業把大量的計算資源下放到前線作業中。這無疑提供異構型分佈式數據庫系統一個理想的發展環境。在分佈式環境中,同義的資料物件難以避免地分別儲存於不同系統中,這現象稱為平行劃分。縱然資料物件的平行劃分被歸類為此等系統的基本問題,但在過去的研究中這問題往往被忽略。本文放棄了被平行劃分的資料物件皆具有相同結構的假設,進而把資料抽象層面和結構差異放在平行劃分的範圍內進行分析并揭開了這系列問題簡單的面紗。自八十年代開始至今,學者們開發了不同的數據庫整合方法,此等方法目的為提供分佈式操作環境,容許本地系統自主,增加系統的兼容性和改善資料保安等等。主流方法為:整體架構,聯合資料庫管理系統(FDBMS)及多資料庫語言。整體架構放棄了本地系統的自主。FDBMS得功能極為倚重其所選的共同資料模型。多資料庫語言則要求用戶對各個本地系統有所了解。其中Metadatabase放棄了慣用的整體架構而採用融合架構資料和總體知識整體模型,從而獲得更高的彈性及透明度。但其原型未有顧及資料物件的平行劃分問題,故本文所提倡的解決方案將以Metadatabase為實現對象。結果證明在有限改動下,解決方案可以融入Metadatabase的體制內并對資料物件的平行分問題實現較以往全面的解決方案。

TABLE OF CONTENTS

C H A P T E R 1 INTRODUCTION	6
C H A P T E R 2 LITERATURE REVIEW	13
2.1.BACKGROUND.....	13
2.2.EXAMPLE SYSTEMS	20
2.2.1.Multibase.....	20
2.2.2.Mermaid	23
2.2.3.The Metadatabase Approach	26
2.3.SUMMARY	29
C H A P T E R 3 THE METADATABASE APPROACH	31
3.1.TWO-STAGE ENTITY RELATIONSHIP (TSER) MODEL.....	31
3.2.THE GIRD	34
3.3.THE METADATABASE SYSTEM IN ACTION	36
3.3.GLOBAL QUERY FORMULATIONS AND PROCESSING IN THE METADATABASE SYSTEM	37
C H A P T E R 4 PROBLEM OUTLINES FOR HORIZONTAL PARTITIONING AND ITS VARIANTS	39
4.1. HORIZONTAL PARTITIONING	39
4.2. LEVEL OF ABSTRACTION.....	41
4.3. SCHEMATIC VARIANCES.....	42
4.4. SUMMARY	43
4.5. THE SCENARIO	44
4.6. POPULATING THE METADATABASE.....	48
C H A P T E R 5 THE ENHANCEMENTS FOR GLOBAL QUERY WITH HORIZONTAL PARTITIONED DATA OBJECTS	51
5.1. IDENTIFYING PARTITIONED DATA OBJECTS	51
5.2. ADDITIONAL METADATA FOR THE HORIZONTAL PARTITIONED DATA OBJECTS	52
5.3. COMPLICATIONS OF HORIZONTAL PARTITIONING PROBLEM	54
5.3.1.Level of abstraction.....	55
5.3.2.Schematic variances.....	57
5.4. GLOBAL QUERY WITH HORIZONTAL PARTITIONING DATA OBJECTS.....	59
5.5. HOUSING THE NEW METADATA.....	68
5.6. EXAMPLE.....	72
C H A P T E R 6 ANALYSIS.....	75
C H A P T E R 7 CONCLUSION AND FUTURE WORKS.....	78
REFERENCES.....	80
APPENDICES.....	84
A. GIRD DEFINITIONS.....	84
A1. GIRD Model.....	84
A2. GIRD/SER Contents.....	84
A3. GIRD/OER Constructs.....	87

<i>A4. Definition of Meta-attributes</i>	89
B. PROBLEMS REPRESENTATIONS IN RELATION ALGEBRA	96
<i>B1. Horizontal problem</i>	96
<i>B2. Level of abstraction</i>	96
<i>B3. Schematic Variance</i>	97
C. DETAILS OF LOCAL SYSTEMS	98

CHAPTER 1

Introduction

Background

In the past decade, information technology and the business world are entangled in a spiral relationship that fuels each other's development. In this information era, by knowing more and faster is the key to win the battle in the marketplace. As a result, information systems are the most important applications which businesses rely on. The decentralized structure of a modern enterprise and the proliferation of PC-based IS applications nourish the existence of distributed information systems. Hence, corporate database systems no longer are housed at the tip of the pyramid of the company hierarchy. Departmental databases are very common in modern multinational enterprises to allow flexible management and speedy accesses.

When the monarchic power of information system management is decentralized, individual systems often evolve independently. This is a natural way for individual systems adapting to the local requirements, especially for coping with the ever-changing business needs. Consequently, these systems are further away from the original unified database design schematically and semantically. Local systems inevitably become heterogeneous in terms of hardware, software, or even schematic database designs. Hence, management's privilege to have consolidated/summarized information at their fingertips expires because information is not easily shareable among the local resources. This kind of information is especially important for executives and/or decision-makers who would focus on the macroscopic picture of their business.

There are three directions to coordinate local systems in a heterogeneous distributed environment. First, re-establishing a *centralized regime* that eliminates the distributed and heterogeneous environment in the expense of foregoing distributed processing and flexibility. Second, implementing a *homogeneous distributed system*, which allows distributed processing but local systems are required to compile to a prescribed system design. In addition, this class of systems incurs higher costs on synchronizing local systems and refitting for new requirements. Both centralized and homogeneous distributed systems require enterprise-wide standardization sacrificing local information systems autonomy that handicaps flexibility. Third, use a framework with an extra layer between local systems and global users, which acts as a middleman for transactions on information exchange and queries from all local system. Therefore, a higher level of autonomy at the local system level can be sustained without preventing information sharing. The framework is called *heterogeneous distributed database management systems* (HDDDBMS).

Information retrieval and horizontal partitioning

Global query is an essential component in heterogeneous distributed database management system. Users often need to retrieve consolidated information from several local systems. Generally, an HDDDBMS, like an ordinary database management system, accepts queries from users and translates them into local queries for different local systems. Local systems will then process the queries and return results to the HDDDBMS for result integration.

In a distributed environment, semantically equivalent data objects are often segmented horizontally. Horizontal partitioning occurs when tuples of a relation (i.e. records in a table) are divided and physically stored in multiple sub-systems. Retrieving information

from these systems without knowing the data are horizontal partitioned, the query result could be incomplete or even incorrect. Despite the fact the horizontal partitioned data objects may hinder information sharing, there are reasons and benefits for data objects to be horizontal partitioned.

- (1) *Hardware/software limitations* – data objects are partitioned into to smaller segments to overcome system limitations and/or to reduce system overhead and recurring costs.
- (2) *Facilitate distributed processing* – production databases, which generate transaction data, are common on the front-line operations and data processing can be done at local sites when data object are horizontally partitioned.
- (3) *Enhance data availability* – users are more willing to utilize information resources if these resources become handier. Also, the level data availability can be assured when there is system failure and/or sabotages.

As a result, horizontal partitioning is concrete and an inevitable issue for information retrieval in a distributed environment. The heterogeneity of the local systems further complicates the issue.

As local systems alter and design their data objects with respect to the local situation, heterogeneity becomes a natural result. Hence, semantically equivalent data object can be schematically different. For instance, a data object, NAME, modeled and stored at different *levels of abstraction (detail)*, such as simply one item NAME in one system and two items (FIRST NAME and LAST NAME) in another system. Another type of design variation, which often exists among the horizontally partitioned data object, is schematic

variance. These two variations are caused by data modeling preferences and business needs. Query horizontal partitioned data with the prescribed variations is not a easy task.

Results from existing system integration methodologies

Four major heterogeneous distributed system integration approaches can be summarized from literature: (1) *integrated schema*; (2) *federated database management systems* (FDBMS); and (3) *Multidatabase languages*. Different systems developed from different methodologies have its strengths and weaknesses in solving the problems regarding information retrieval form horizontal partitioned data objects. Literature on systems employing integrated schema approaches mainly stress on the method to generate the global schema from the local system. Due to the structural conformity, the problem is expected to be easier to be solved in the integrated schema approach. However, the integrated schema approach are not favorable to dynamic business environment because its limitations on the local autonomy and adaptability. Multidatabase language systems require user to resolve the conflicts in each encounter. Intensive user interventions on the conflicts among local systems are required in this class of systems. As a result, it is regarded as a partial solution to the problem as it provides very limited user assistance and local system transparency. Various implementations of the federated database management system employ different common data models. Local systems with different data models must be mapped to the chosen common data model. So that, the schema integration is not done to the local systems and hence, a higher level of adaptability and autonomy can be obtained. Yet, loss might be incurred during the mapping processes due to modeling capability of different data models. Therefore, they impose assumptions on schema mappings to reconcile discrepancies in different data models, including mappings for horizontal partitioned data objects.

Both generic and methodology-specified methods for solving this problem were given in previous researches, they were overly simplified that without considering the combinations of the two complications, level of abstraction and schematic variances, to be addressed. The two variants are seldom put under the scope of the horizontal partitioning because of implementation assumption or limitations inherited from the methodology and/or data model. Therefore, it is a different perspective in analyzing horizontal partitioning with the two variants.

The Metadatabase approach

Metadata are not only used to describe information about data items like in a data dictionary. In the Metadatabase approach, the functionality of metadata is extended to convey information about information resources, like hardware and software resources information, and knowledge about relationships and interactions of information resources inside an enterprise. With the capability of modeling knowledge, concepts that are implicitly implied can be modeled. This knowledge can provide more intelligent assistance to users such that users are save from technical details of local systems. As a result, information resources can be systematically modeled in the Metadatabase system and, hence, information resources are more manageable. The approach has the following strengths: (1) local system autonomy – local systems have the rights to design, to operate and to evolve according to local needs; (2) system transparency – global users do not need to deal with sub-systems or not even realize their existence. As the global model has sufficient enterprise knowledge, the system can provide users' with technical details for global queries; (3) adaptability – adding new hardware/software and system migration do not require major re-build of the global system; and (4) interoperability – accommodating heterogeneity while resolving conflicts in data models, data item definitions and data manipulation languages.

Other methodologies might have different balances on the achievements due to limitations inherited from the methodology employed and/or from the system design and assumptions. For example, some of them might focus on information sharing by foregoing local system heterogeneity while the other try to preserve local system autonomy by giving up system transparency. With all the strengths from the approach, the prototype of the Metadatabase, however, has not been enabled for retrieving information from horizontal partitioned data objects.

The proposed research

This paper is to address challenges induced by the horizontal partition data objects in global query processing in a heterogeneous distributed environment. At the same time, we need to maintain high levels of local autonomy, transparency, adaptability and interoperability. Metadatabase approach has strengths that are required and these strengths are not acquired by other existing methodologies. We believe that the Metadatabase system can be extended with additional functionality to process global queries with horizontal partitioned data objects (and its variants). Hence, the proposed solutions will be materialized using the Metadatabase approach.

In this study, a case is set up, which contains horizontal partitioned data objects and its variants, for illustration purpose. New metadata are identified by going through a some exemplary queries with horizontal partitioned data object and/or its variants to the case system. Then, the current global query processing algorithms are enhanced to take advantage of these newly identified metadata such that the Metadatabase system can retrieve information from horizontal partitioned data objects. Enhancements are made with considerations that the prescribed strengths of the Metadatabase approach must be

retained. The set of newly identified metadata is, consequently, incorporated into the Metadatabase such that horizontal partitioned data objects are properly modeled.

The organization of this thesis

The flow of this paper follows: Literature reviews on common database integration methodologies and the respective methods for handling horizontal partitioning problem are given in Chapter 2. As the proposed solutions will be implemented to the Metadatabase system, additional information about the Metadatabase approach is given in Chapter 3. Then, the horizontal partitioning problem and its variants are revealed, together with a scenario, Chapter 4. Chapter 5 is centered on the proposed solutions for the problem and how they are incorporated into the Metadatabase system. Analysis on the advantage of the new methods in comparing to other methodologies will be discussed in Chapter 6 and, finally, followed by the conclusion.

CHAPTER 2

Literature Review

2.1. Background

The premises of building heterogeneous distributed database management (HDDDBMS) are to address the distribution, autonomy and interoperability problems:

Distribution Distributed databases simply means that information is stored (distributed) in different database systems. Therefore, a distributed database can be homogenous if sub-systems share the identical design, software and hardware specifications. Otherwise, the distributed database system becomes heterogeneous.

Autonomy Local systems have full control over its system domain like system design, information sharing and process executions. In addition, merges and conversions should not be required for a local system level to participate in the HDDDBMS.

Adaptability Evolution of local systems is unavoidable when autonomy is granted. A HDDDBMS must be capable to adapt changes in local systems with a reasonable effort such that it does not require over-haul when there is any change in local systems.

Interoperability Dissimilarities emerge naturally in an automated environment. These dissimilarities might occur at any level from design to implementation of an information system and this is called heterogeneity. A HDDDBMS has to be catered for heterogeneous

environments and to make its member systems interoperable to each other such that they are benefited by their mutual existence.

In addition to these four dimensions we must address to, there are tactical issues to be resolved different kinds of conflicts among heterogeneous systems [SL90, ERS98]:

Horizontal partitions It is an inevitable phenomenon in a heterogeneous distributed environment that semantically equivalent data objects are horizontal partitioned in different systems.

Naming conflicts Synonyms map a real-world object to different names in different database systems. In contrast, homonyms map different real-world object into identical names in various systems.

Scaling difference Fields with identical names and real-world counterparts might be incompatible due to different in scale, unit and/or frequency. For instance, US dollar verse HK dollar in *price*; Zulu time verse local time in *time*.

Key equivalence Keys are the identifiers of data objects in database systems. Equivalent keys might be mapped to different real-world objects due to autonomy. This renders identifiers invalid.

Schematic variance With different modeling techniques, the domain of a data item a in system A might be divided into different subsets $\{b\}$ in the other system, B (or vice versa). At the same time, the data items $\{b\}$, each of which has a domain subset of a , belong to another data item c in A. i.e. in relational terms, the intension of a data object is in

the extension of the other one in different systems. In other words, fields/attributes of a data object is values of an attribute in other one.

Level of abstraction The same piece of information are captured at different level of detail, for example, customer name is stored as a single field in one system while it is stored as *firstname* and *lastname* in another one.

Many researchers had devoted their efforts on formulating methodologies and building HDBMS systems, which meet the premises, since 80s. This makes modern enterprises enjoy the synergy and flexibility of localized information systems given networking and communication infrastructures are entrenched in our society.

The bottom line of the integration process is to give users a uniform interface to those data stored in different databases. No matter a global schema or a model is put in place, it has to fulfill this need. There are three categories of heterogeneous distributed database management system architecture [ERS98]: global schema integration [BLN86, Mot87, BKDV92, SP94], federated databases management systems (FDBMS) [SL90] and multidatabase languages approach. In addition to that, data dictionary systems are also included in this analysis.

A ***global integrated schema*** is built by either a top-down or a bottom-up approach. The top-down approach enforces a globally defined schema to all component databases. The bottom-up approach performs by component databases pair-wise or group-wise unification and finally come up with the global schema. No matter which route is picked, every systems under the umbrella of the global schema are assimilated using a uniform representation in the global schema integration methodology. It provides, obviously, a

total integrated and conflict-free environment as of the integration date. Environmental changes, which are common in a dynamic environment, invalidate the global schema leading to reworks. It is also questionable whether a global schema can even be achieved when an enterprise spans over several industries and their business models are conflict to each other. A fatal drawback of this methodology is that it sacrifices local system autonomy totally.

Federated database management systems consist of 5-layer (from local to global) [SL90]: local schemata, component schemata, export schemata, a federated schema and external schemata. A common data model (CDM) is enforced to all these schemata except the local schema, i.e. local autonomy can be achieved to a certain extent. However, (a part of) the schema in each of local systems is required to be converted into a component schema as a premium to join the federation. A FDBMS effectively starts integration at the component schema level but not at the local schema level. An export schema is a restricted version of the respective component schema such that accesses from the global level can be controlled. The federated schema combines all or some of export schemata and it can be further partitioned into several external schemata for different groups of users. Besides the five schemata, a FDBMS performs auxiliary functions through different processors: transforming processors, filtering processors, a constructing processor and a federation dictionary. The transforming processors carry out bi-directional mapping (with help from the federation dictionary) between local schemata and the corresponding component schema. Filtering processors screen global service requests according to access control information specified in export schemata. The constructing processor distributes global service requests among export schemata and integrates results from local systems as a response to the request.

With the basic structure charted, FDBMSs are further branched out as *loosely-coupled* and *tightly-coupled* FDBMS. The administration rights of the global schema marks the difference between loosely-coupled and tightly coupled FDBMS [GSC95]. Tightly-coupled FDBMSs, Fig. 1, are capable to house one (e.g. MERMAID) or multiple (e.g. Multibase) federated schemata. The centralized administration of the global schema in tightly-coupled FDBMS facilitates the integration process by minimizing duplicated effort on setting up a unified data representation; nonetheless, the static nature of the federated schema(ta) handicaps tightly-coupled FDBMSs dealing with environmental changes.

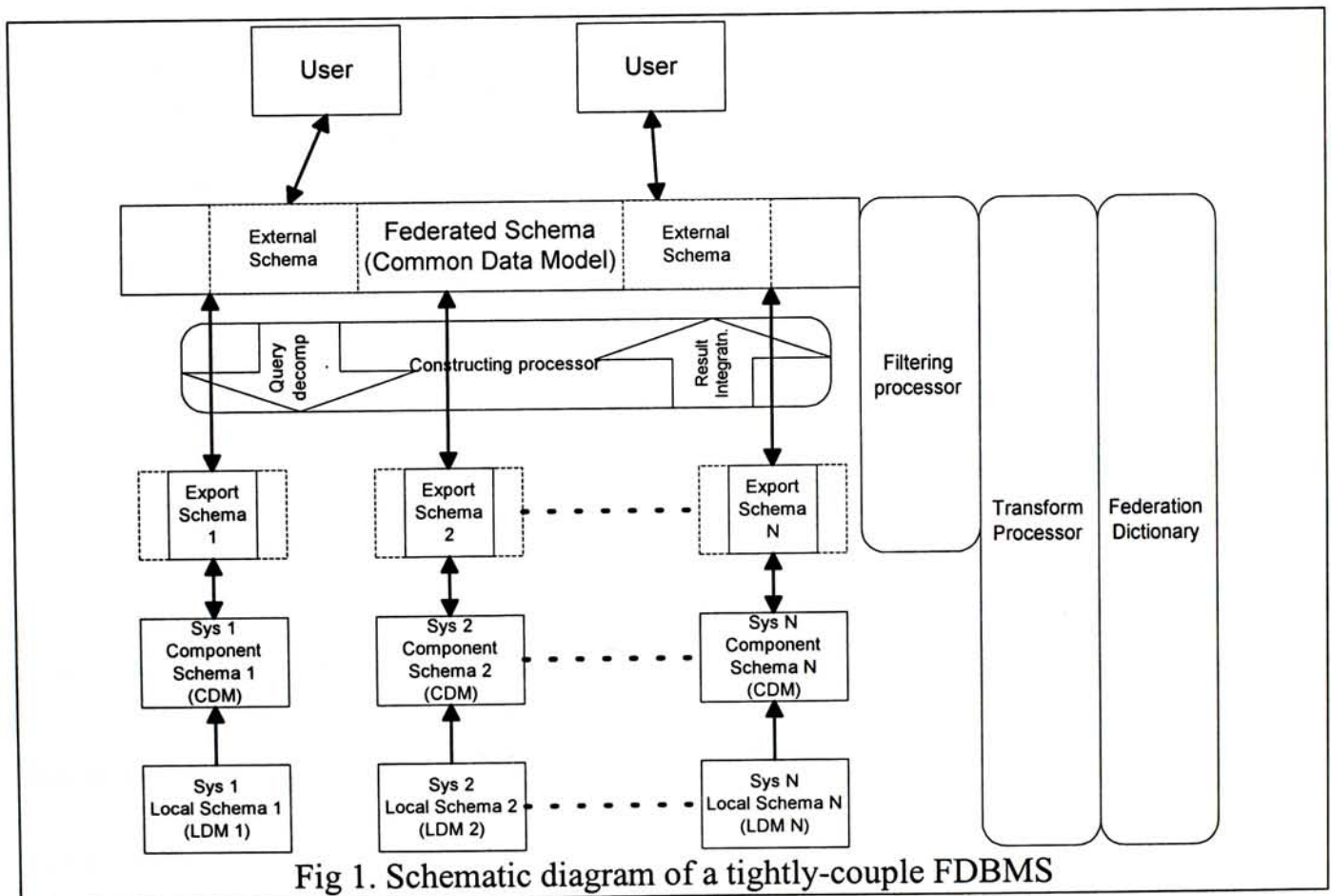


Fig 1. Schematic diagram of a tightly-couple FDBMS

On the other hand, the administrations of federated schemata are delegated to individual users in loosely-couple FDBMSs [CR93, YOL97, BCDE93, ZSC95]; as shown Fig. 2. Users are permitted to create their own federated schema through view integration of export schemata. View integration allows different semantic mappings in different federated schemata. Lossely-coupled FDBMSs are more adaptive to environment

changes as view integration induces less rigidity than schema integration does. Synchronizing customized federated schemata with local systems evolution become difficult because more administrators and semantic mappings are involved. Also, duplicated works on building similar federated schemata are inevitable.

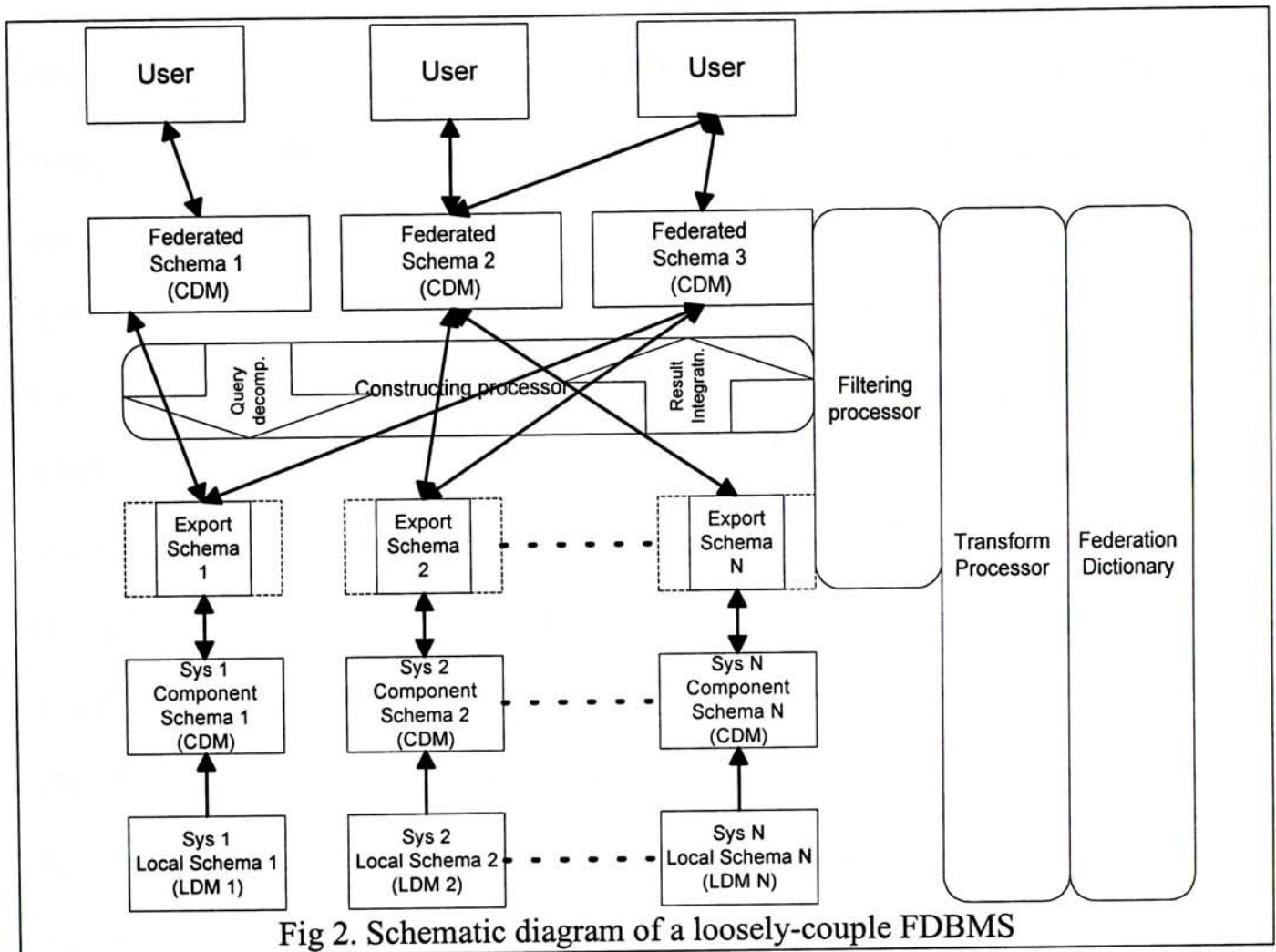


Fig 2. Schematic diagram of a loosely-couple FDBMS

Multidatabase language approaches [HBP94, Lit93, Lit94, LMR90] aim to provide basic constructs such that users can build their own global query over a cluster of databases systems without having a integrated schema built. These languages extend the capability of traditional query languages to handle multiple data base environment. The advantage of these approaches is that participant DBMSs can excises high level of autonomy, even higher than that in loosely-coupled FDBMSs. Yet, user might be required to re-learn and understand schemata of local systems every time start a new query session in such environment because changes have be made to local systems. Therefore, the major

challenge of this approach is to help users identifying relationship among interrelated schemata in addition to resolving schematic and semantic conflicts.

Data dictionary systems have been used for supporting life-cycles of database systems as well as database integration like the Information Resources Dictionary System (IRDS) [DK87]. They have shown their effectiveness in management information resources by using metadata. However, these systems do not incorporate knowledge of interaction among database systems inside an enterprise. The Global Information Resource Directory (GIRD) tools a step forward that wields schematic, semantic and knowledge together. The traditional ER data model is criticized to be incapable capturing semantic information, which is essential to model interaction among systems in a multi-database environment. The Two-stage Entity Relationship (TSER) model incorporates semantic information and knowledge into the traditional ER data model. Hence, the Metadatabase approach uses TSER to model the GIRD, which is employed as the unified metadata model of the Metadatabase. The total absence of a global schema and availability of knowledge model distinguish the Metadatabase from methodologies/approaches mentioned before. Local systems and their interactions are model in the GIRD. A global model can be customized over heterogeneous database systems in an enterprise that masks all heterogeneity from the users' perspective like a tightly-coupled FDBMS. On the other hand, user-defined views can be materialized and stored in the Metadatabase similar to a loosely-coupled FDBMS. It further provides a uniform query language that operates on multiple databases environment with on-line assistance analogous to functionality provided by multi-database languages approaches as mentioned. Yet, users are saved from knowing schematic details of local system as the knowledge model will compensate these efforts. With all the strength embodied in the Metadatabase, this paper will be using it as the base approach to solve the horizontal partitioning and schematic variances problems.

2.2.Example systems

Multibase and MERMAID are chosen to be example systems because both systems are mature and functionality address to the problems discuss in this paper has been developed or discussed dedicatedly. Other literature on HDDBMS is either not touched on the horizontal partitioning problem [SY96] or just claiming the problem is solved in a limited scope [PRSL93, LSS94, Chu90] and/or without substantial supporting literature about the process and methodology [Hua94, PRR91, GSC96, Sou93]. Sketches of the outlook of the two example systems are revealed here as corner stones for comparison in later section.

2.2.1.Multibase

Multibase performs database integration by means of schema and language translation using the functional data model (FDM) as the global model. It employs a three-tier structure using functional data model [SBD+81, SP94, CH96] as shown in Fig. 3. The three tiers are (1) the global schema; (2) the integration schema and local schema; (3) local host schema. The global schema provides users a uniform interface to those local systems. Multiple global schemata can be defined to materialize different views of local systems and to satisfy different integration requirements. The local schema are the receptors of the global system which translate schemata of a local system (local host schemata) into the global data model. The traditional Multibase [SBD+81] used the functional data model as the CDM. (There is a modified approach using relational model in [Hua94].) The schematic design of Multibase is shown in Fig. 3.

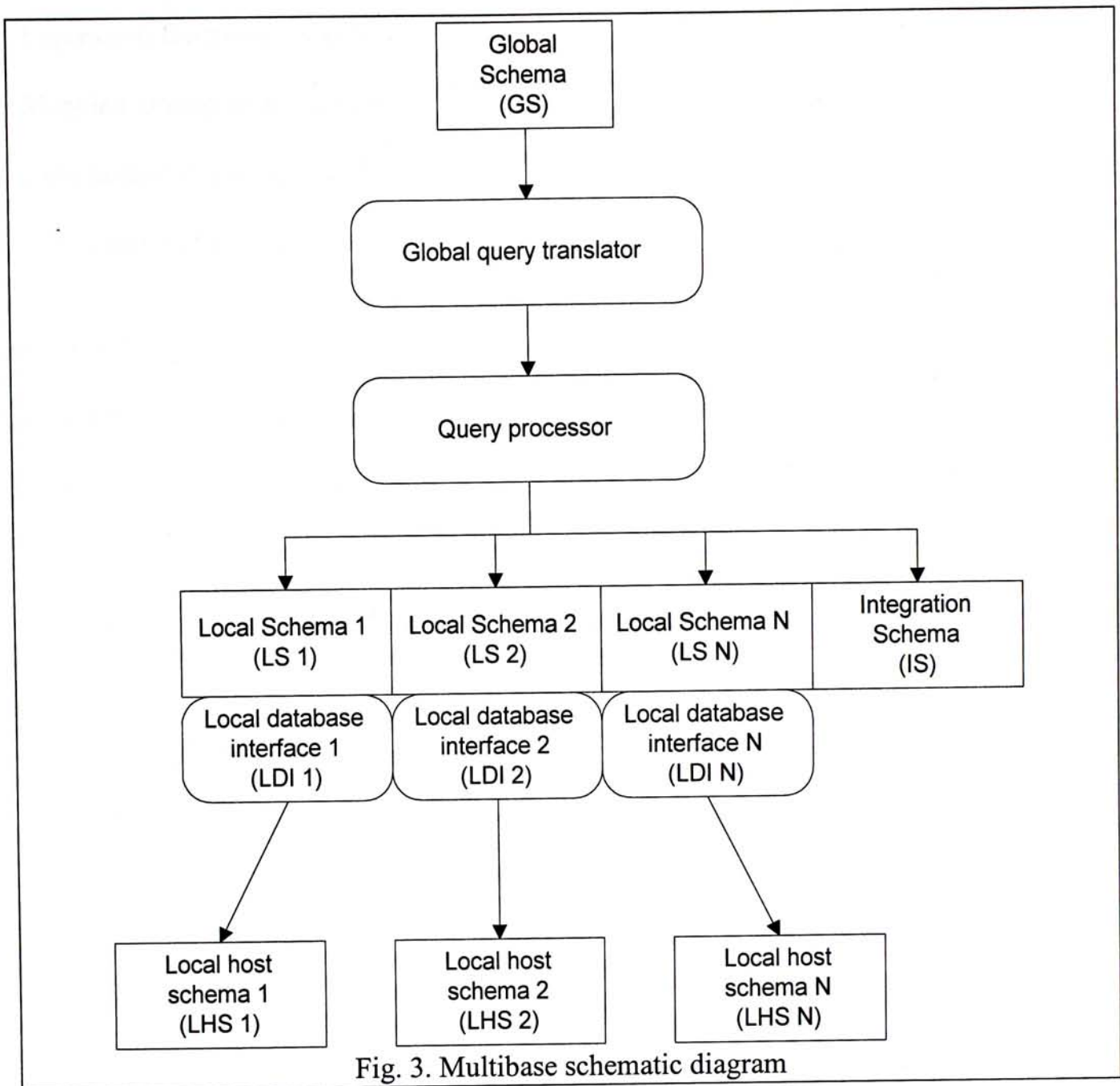
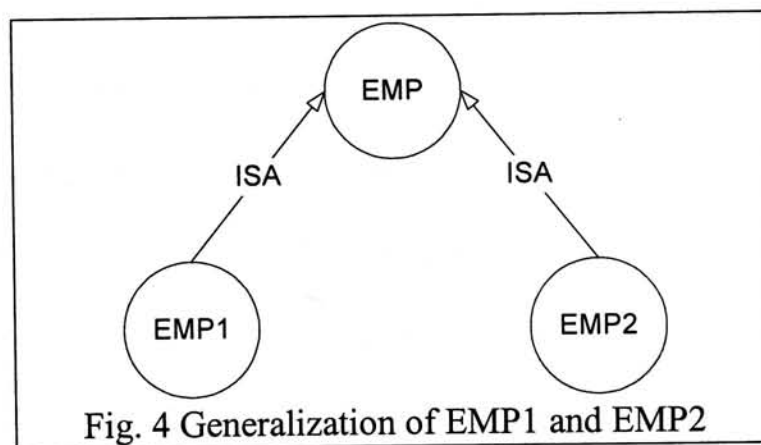


Fig. 3. Multibase schematic diagram

The integration schema in which contains information for resolving conflicting data objects in local systems. Conflicts are resolved by a two-step process: schema and domain integration. Data integration language (DIL) – DAPLEX [Shi81] is provided to map local data objects into the local schemata so that they are represented using CDM before schema integration. Local schemata are further mapped into the global schema using the DIL. Inconsistency and conflicts either resolved by altering LHSs into uniform representation or by mapping information stored in the integration schema (IS). Functions can be written in DIL to handle vertical, horizontal partitioning and schematic variances [SBD81+]. Domain conflicts are resolved by conversion and aggregation functions.

Expression functions are used for representing mathematical relations among data objects. Mapping among string, textual, information can be made by enumerative functions. More complicated mappings that do not have a direct functional relationship can be derived by procedural and aggregation functions.

Horizontal partitioning problems are resolved by generalization such that a generic data object is used to represent the horizontal partitioned set of data objects. Suppose there are two local schema for keeping employees' data: EMP1 (SSNo, Name, Sal, Age) and EMP2 (SSNo, Name, Sal, Address). A generic entity, say EMP, will be created in the global schema for generalizing EMP1 and EMP2. There are total three entities in the global schema: EMP(SSNo, Name, Sal), EMP1(SSNo, Age), EMP2(SSNo, Address). In order to show EMP1 and EMP2 are "subclasses" to the EMP, two ISA-relations are imposed to signify relationship among these three entities as shown in Fig. 4.



By using the generalization method, horizontal partitioned data objects are partially represented by a "superclass". Attribute (functions in FDM) set of the superclass is determined by the intersection of that of all partitioned entities. Any residual attribute in the partitioned entities that cannot be represented by the superclass has to be represented as a separate entity with a ISA-relation pointing to the superclass. As a result, both EMP1 and EMP2 still exist in the global schema after generalization. Therefore, only those common attributes of a set of horizontal partitioned data objects can be retrieved when the

superclass is targeted, hence, the global schema cannot make all local partitions transparent to users.

All the functions defined in the functional data model are entity-specified that make complex global data object difficult to be maintained. When there is any change to local schema, all related functions and methods have to be rebuilt. For huge global schema where there are a lot of horizontal partitioned data objects, efforts paid in updating old functions will be enormous. Under such circumstance, necessary changes are inhabited or prolonged system life-cycle invalidate changes such that the system adaptability is crippled.

2.2.2.Mermaid

Mermaid is a front-end interface on a group of heterogeneous database systems [TBD87]. Its schematic diagram as shown in Fig. 5. The primary objective is to allow is retrieve data from different local systems with a standard interface. Updating local databases in Mermaid is a trivial case, only one database at a time, as data entries are assumed taking place at local system. There are two global query languages are currently supported in Mermaid: SQL, for relational schema, and ARIEL, for semantic schema. A highly structured distributed intermediate language (DIL) is used as the communication media among all parties in the system. However, a translator for each node of the system is required to translate between the global and local query languages. New global query languages can be adopted if a corresponding translator is developed. A basic set of query function is defined in DIL. Therefore, the system only support local systems have the defined function set, i.e. DIL can be translated into local query language; otherwise the local system will be dropped. The global schema is/are stored in the *Data Dictionary/Directory* (DD/D) using relational model in Mermaid. There might be

multiple global schemata defined to fulfill different integration needs but they must be stored in different DD/Ds. Besides schematic information, system and physical information of local systems are stored in the DD/D as well to facilitate query optimization process. Data translation is required if there is any discrepancy between the data representation in the global level and the local systems. For instance, unit and scale, grouping, etc. Two types of mapping can be applied on the conflicting data: functional and enumerated mappings. In order to create the global schema, local schemata are converted into relational model; and then, local schemata are unified by schema translation methods given in Mermaid.

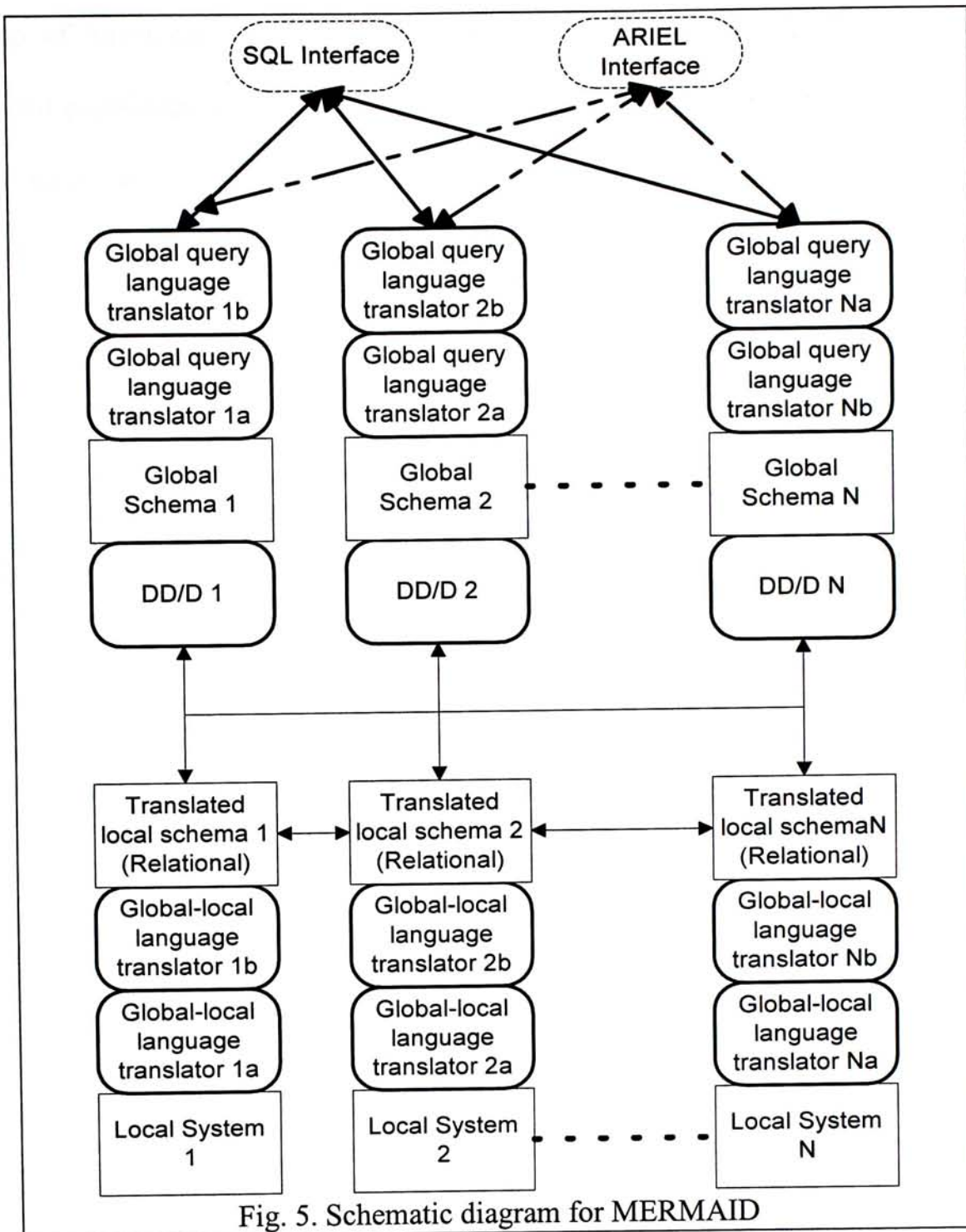


Fig. 5. Schematic diagram for MERMAID

At the relation level, vertical and horizontal partitioning, projection, one-to-many and many-to-one relation mappings can be resolved by methods provided.

Horizontal partitioned data objects in MERMAID are classified into four categories: local, replicated, fragmented, dependent fragmented. Local data objects exist in one and only one site are *local*; exist in multiple sites in form of duplicates are *replicated*. Those *fragmented* are horizontal partitioned data objects with disjoint partitions, i.e. it is assumed that there is no overlap among partitions. Dependent fragmented data objects are

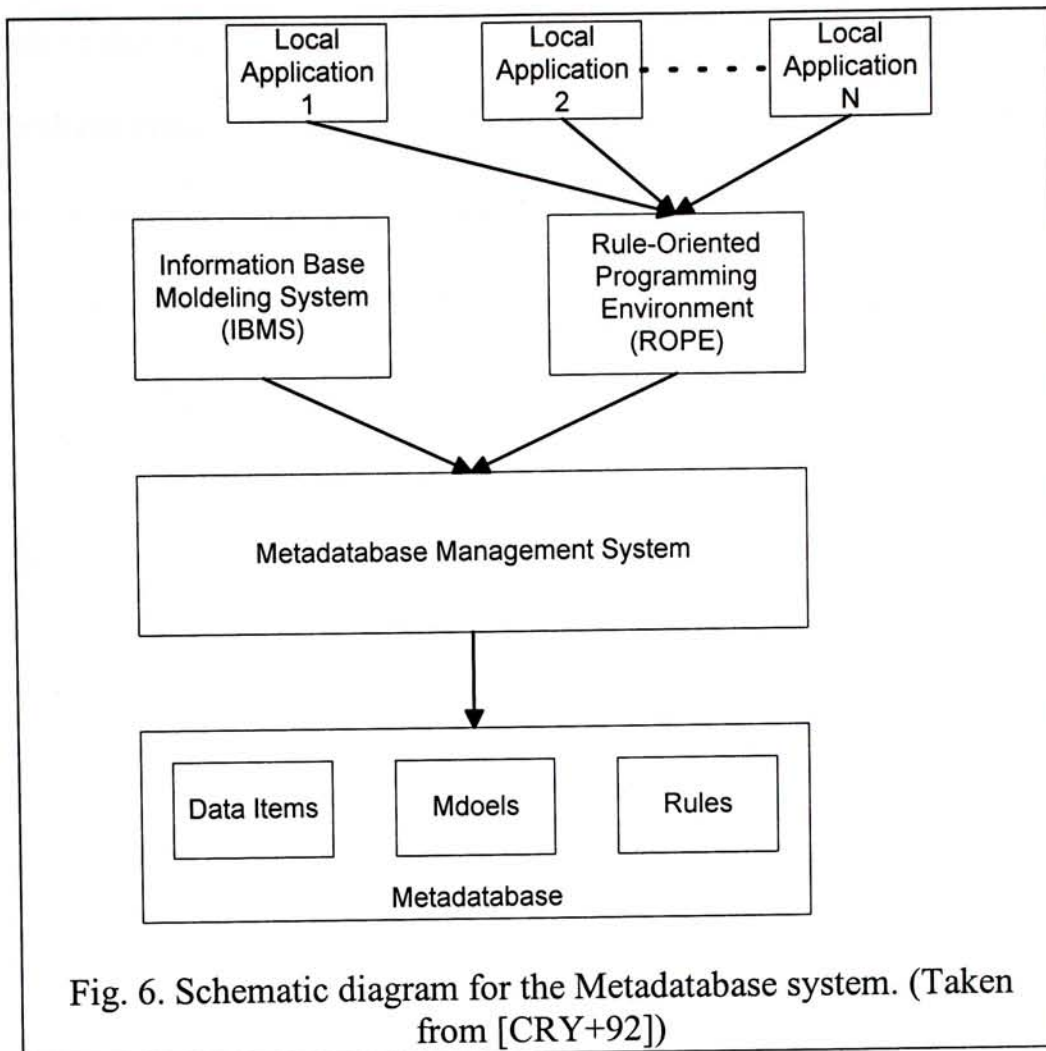
a group of functional dependent fragment data objects. Query optimization regarding horizontal partitioned data objects are discussed in [CBTY89]. The proposed algorithm tried to move “fragments” involved in the query into one processing site, where the query result will be generated, and perform union before further processing [RPR88, CTBY89]. The algorithm took processing and transmission times (costs) into account and tried to minimize them giving shortest query respond time. There are seven steps in processing queries targeting horizontal partitioned data objects. (1) The algorithm first applies all selection criteria to all fragmented relations wherever applicable. As a result, the data volume to be transferred can be reduced. (2) A heuristic is applied to select fragmented relations to process sites so that the sum of transmission and processing times is minimized. Replicating some of fragmented relations to the processing sites might be required. (3) Semijoins are applied to relations located in the same site to further reduce data transmission time. (4,5) Fragmented relations are move to respective processing site determined in step (2) and process the query in parallel. (6) Results from different processing sites will be assembled at the result site where the query was first issued. (7) Aggregation, eliminating duplicated records and formatting for output will be done at the final step at the result site.

2.2.3.The Metadatabase Approach

Metadatabase employs Two-Stage Entity (TSER) Relationship model as the data model. TSER is developed as a modeling tool for complex information modeling. It is capable to model both functional and structural model of information resources in an enterprise. Both models are modeled separately and integrated in the later part of the integration process. After local systems modeled using TSER, they are transformed into metadata and stored in the Global Information Resources Dictionary (GIRD). The GIRD combines semantic, schematic, facts (static) and knowledge (dynamic) information of all the local

systems. (Refer to appendix A for the GIRD model) Facts and knowledge are used to synthesize rules regarding inter- or intra- local systems operations. Metadata about local systems together with knowledge and rules stored in the GIRD become the Metadatabase. Therefore, global schema does not exist in Metadatabase. Local systems and knowledge about their interactions are fused together in forms of metadata, and hence, a global model about information resources inside an enterprise is established. Without the constraints of schema transformation a higher degree of transparency and autonomy in local system can be obtained.

Besides, there are three addition components in the Metadatabase system other than the Metadatabase itself [HRY+92]. (1) The Metadatabase management system (MDBMS) provides required functionality to maintain and utilized metadata stored in the Metadatabase. In addition, it acts as an interface for the two following components to the Metadatabase. (2) The information base modeling system (IBMS) is a CASE tools helping users model and maintain the information resources in an enterprise and finally the GIRD is populated with proper metadata. (3) The rule-oriented programming environment (ROPE) interface the MDBMS and local systems. It helps implement and enforce knowledge on interaction among local systems. It is implemented as a software shell that monitors any local system activity that is significant to the enterprise as a whole. As a result, overall system consistency can be ensured. The schematic diagram as shown in Fig. 6. The Metadatabase can also be deployed in a distributed environment for large enterprise.



The Metadatabase system has three modes of operation. They are passive, semi-active and active mode. In passive mode, the Metadatabase is as simple as an information direction of local systems. The Metadatabase system is not necessary connected to the local systems. Rules for resolving conflicts among sub-systems are not required because only information regarding sub-systems, not data stored in them, can be retrieved by users. A global query system is an addition feature of the Metadatabase system in semi-active mode. That allows the Metadatabase system interacts with local systems and processes users' queries on data stored in sub-systems with on-line assistance on query formulation. Operations are carried out and monitored by *shells* built around each of local systems and the Metadatabase system itself. Shells are built on top of respective systems using local development tools and pose no intervention on local operations. Rules for resolving conflicts on data items and assuring consistency across systems are

implemented in this mode. These shells also provide a foundation for the active mode of the Metadatabase system. In the active mode, knowledge of sub-systems is formulated as rules in the system that enhances the control of the overall operations. There are two sources of rules: (1) from direct inference of the model and (2) from users. The system provides utilities for rule generation from both sources. In addition, utilities for rule implementation and maintenance are also provided. As a result, adaptive ways for sub-system interactions are achieved because rules change automatically as the model changes or as per users' requests.

For the objectives listed at the beginning of this chapter, the Metadatabase system provides comparably better solutions than other methodologies as discussed in previous sections. However, the system does not cater for the horizontal partitioning problem. It is a natural consequence for the original idea of the Metadatabase is for computer integrated manufacturing environment that horizontal partitioning problem is not common.

2.3.Summary

The federated database management systems approaches have substantial improvement from the integrated schema approaches. Schemata of local systems are preserved in FDBMS approaches in most cases. This increases the level of autonomy. As mappings from the local schema to the component schema are required, there are occasions that functional and data model incompatibilities among the local systems and the federated systems. To resolve such situations, local systems have to be changed accordingly or leave those systems out of the federation. Loosely-coupled FDBMS approach is in turn more adaptive and flexible than the tightly-couple counterpart. The adaptively and flexibility are given by the user-defined federated schemata. Yet, without coordination, duplicated efforts are inevitable. The formulation process, however, required extensive

knowledge about local systems and the common data model. For actively changing federations, recurrent efforts must be paid for updating user-defined federated schemata. Hence, users cannot avoid dealing with local systems repeatedly. Drawbacks of FDBMS is rooted in schema mapping from the local schemata in local data model to the federated schema(ta) in the common data model. The mapping mechanism cannot create additional concepts that do not schematically exist in local schemata. In other words, functionality of the federation is limited by that of local systems.

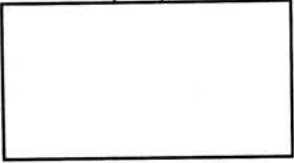
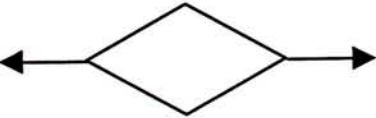
Different methodologies develop their own methods to recover information scattered in horizontal partitioned data objects. With different methodologies and data modeling techniques employed, assumptions and limitations are inherited. MERMAID defines a set of basic functions of the global system help identify if a local system can be fully incorporated into the federation. Horizontal partitioned data objects are assumed to be either replicas or disjoint set despite that does not true for all business models. On the other hand, Multibase handles horizontal partitioned problem in a more flexible manner by creating a generic global entity that capture the common attributes of a set of horizontal partitioned data objects. For those attributes that cannot be generalized, they are retained in the global schema as separate entities. As both FDBMS in general and its instances have deficiencies in handling horizontal partitioning problem, new methods for the problem are worth investigating. The Metadatabase approach is selected to materialized the improved method on solving horizontal partitioning problems given that its strengths in achieving objectives for a HDDBMS.

CHAPTER 3

The Metadatabase Approach

3.1. Two-Stage Entity Relationship (TSER) model

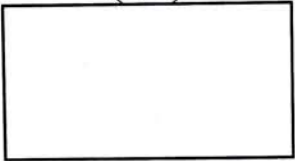
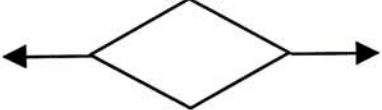
TSER is the vehicle to convoy information resources inside an enterprise to the global model in the Metadatabase system. It is developed as a modeling tool for complex information modeling targeting both functional and structural models of information resources in an enterprise. Both models are modeled separately and can be integrated in the later part of the integration process with at least 3NF [HBRY91]. TSER models the functional model with two constructs: SUBJECT (SE) and CONTECT (SR). A SE stores database views and their functional dependencies while a SR keeps relations and interaction among SUBJECTs.

Construct	Primitives	Description
<p style="text-align: center;">SUBJECT (SE)</p> 	<p>Contains data items (attributes), functional dependencies (among data items), intra-SUBJECT rules (characterized by data items belonging to a single SUBJECT), and class hierarchy (generalizes and aggregates SUBJECTS).</p>	<p>Represents functional units of information such as user views and application systems, and is analogous to frame or object. Triggers and dynamic definition of items are examples of intra-SUBJECT rules.</p>
<p style="text-align: center;">CONTEXT (SR)</p> 	<p>Contains inter-SUBJECT rules (characterized by items belongs to different SUBJECTS), typically includes directions of flows for logic (decisions and control) and data (communication, etc.)</p>	<p>Represents interactions among subject and control knowledge such as business rules and operating procedures and is analogous to process logic.</p>

Construct	Primitives	Description
<p>Note:</p> <p>(1) The full contents (as applicable) must be specified for all SUBJECTS at the leaf level of the SUBJECT hierarchy. The class hierarchy implies integrity rules for applications, but its presence is not required.</p> <p>(2) Rules are constructed in the form of (a subset of) predicate logic where all clauses must only consist of the logical operators and the data items that have been declared or defined in the subjects, excepting certain key words such as <i>do</i> and <i>execute</i>. A data item may be defined to represent an executable routine, algorithm or mathematical expression</p>		

Table 1. TSER functional model (SER) constructs¹

At the structural level, SRs are mapped into Operational Entities (OE), Plural Relationships (PR), Functional Relationships (FR) and Mandatory Relationships (MR). An OE is characterized by a single primary key and is analogous to an entity in ER or an object in object-oriented framework. A PR has composite keys and connects to either other PRs or OEs. A FR signifies the functional dependence among OEs and/or PRs. The arrow is pointing to the **determined** and the **determinant** is on the normal end. The arrow side of a MR called the **owned** while the normal side is called the **owner**. A MR also depicts functional relationship among OEs and PRs with condition that if the **owner** does not exist, there will not be any **owned**.

CONSTRUCTS	DEFINITION AND DESCRIPTION
<p>OPERATIONAL ENTITY (OE)</p> 	<p>Objects identified by a singular primary key, (optional) alternative keys, and non-prime attribute</p>
<p>PLURAL RELATIONSHIP (PR)</p> 	<p>Association of entities characterized by a composite primary key and signifying a many-to-many and independent relation.</p>

¹ Taken from [HBRY91]



CONSTRUCTS	DEFINITION AND DESCRIPTION
<p data-bbox="170 226 688 309">FUNCTIONAL RELATIONSHIP (FR)</p> 	<p data-bbox="776 219 1398 376">A many-to one association that signifies inheritance relationships. FRs represents the referential integrity constraint implies by the existence of foreign keys.</p> <p data-bbox="776 413 1382 687">The arrow side is called the <i>determined</i> and points to either an OE or a PR, while the other side is called the <i>determinant</i> and is also linked to either an OE or a PR. The primary key of the <i>determined</i> side is included as a non-prime attribute (i.e. a foreign key) of the <i>determinant</i> side.</p>
<p data-bbox="188 700 680 782">MANATORY RELATIONSHIP (MR)</p> 	<p data-bbox="776 692 1360 807">A one-to-many <i>fixed</i> association of OEs. MRs represents the existence-dependency constraint.</p> <p data-bbox="776 844 1403 924">The "1" side is linked to the <i>owner</i> OR while the arrow side points to the <i>owned</i> OE.</p>
<p data-bbox="126 941 212 974">Note:</p> <p data-bbox="126 974 1333 1106">(1) In both top-down design and reverse engineering, the OER model is typically derived automatically from the SER model by using the TSER normalization algorithm.</p> <p data-bbox="126 1106 1377 1228">(2) While there usually are multiple SER representing different views or application systems of an enterprise model, there always exists only one integrated OER model for the global system</p>	

Table 2. TSER functional model (SER) constructs²

Although TSER is in a form of Entity-Relationship representation, it is not restricted to model ER-based systems. Among all information resources in an enterprise, each of applications is modeled as a functional view. Each of those functional views is further broken down into sub-views as an application might provide multiple functions. Then, the corresponding structural views are derived from functional sub-views. Finally, views are integrated into the global model, which represents information resources of the entire enterprise under the scope of the system. The building the meta-model of an enterprise is shown in Fig. 8.

² Taken from [HBRY91]

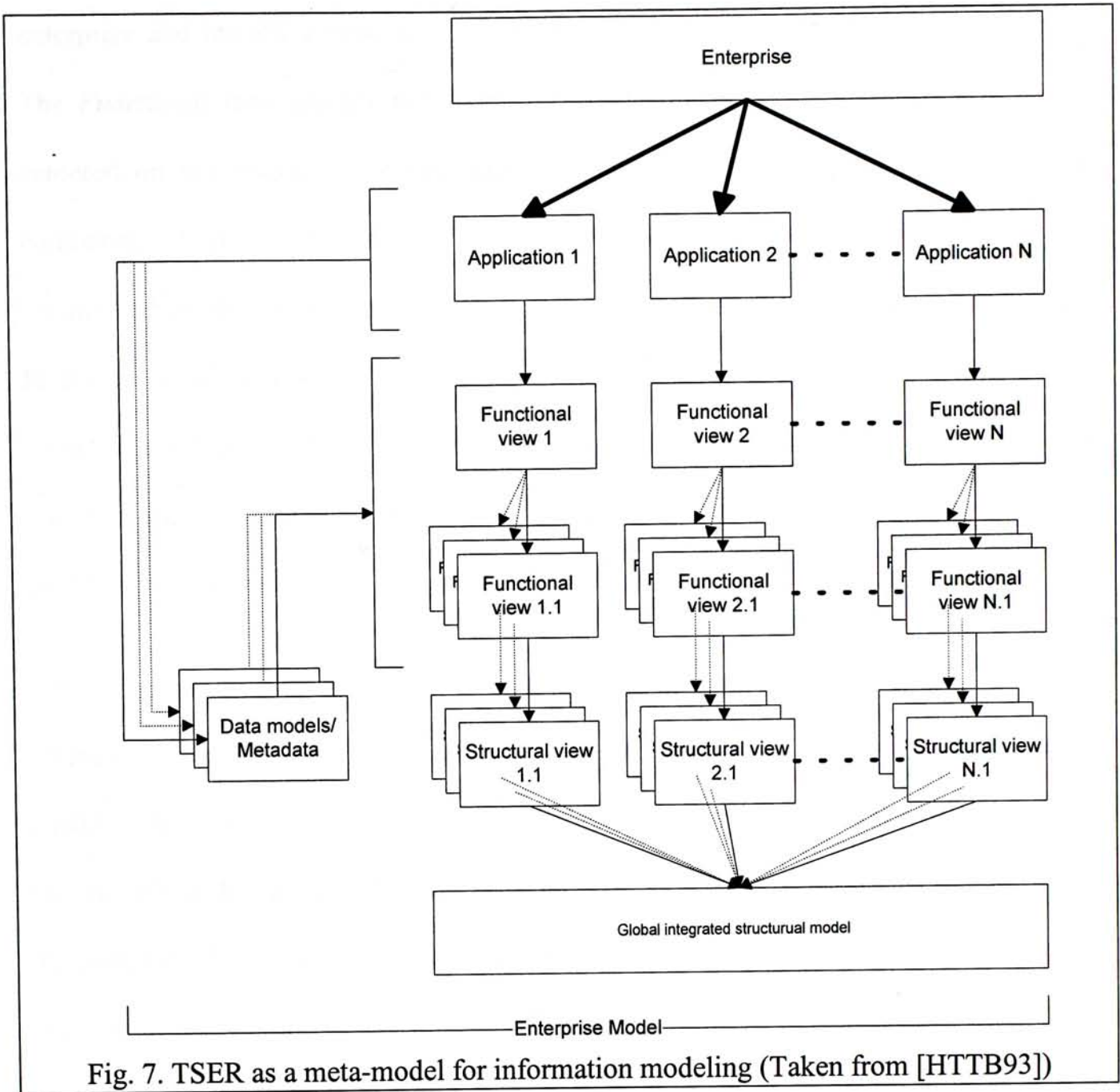


Fig. 7. TSER as a meta-model for information modeling (Taken from [HTTB93])

3.2. The GIRD

Functionally, the meta-model comprises of four views: (1) Application; (2) Functional; (3) Structural and (4) Resources. Among these four views, contextual information, inter-relations of views, is modeled according to literature, target modeling paradigms and empirical studies. The four views are built by TSER constructs and that streamlines the consolidation process for forming the GIRD.

The *Application view* gives a high-level abstraction of **applications** employed in an enterprise and models interaction among applications and **users** like users' privileges. The *Functional view* models the enterprise at the functional or semantic level. It is centered on the interaction among **context**, **subjects** and **data items**. Subjects are component of applications and described by data items. Data items form local systems are masked by their global counterparts and hide from users' view in the global model. **Rules** can be applied when conversion and reversion is needed. Context is the dynamic knowledge among subjects. The interactions are modeled by rules, as well. Rules are pooled in the rule base in form of logical expressions built by **Facts** (static knowledge) in the Metadatabase system and they can trigger external functions and procedures.

The *Structural view* is the structural model for the global model of the enterprise. Metadata about global **entity-relationships**, OE and PR in TSER terms, are stored in the GIRD for binding local data objects logically. Local integrity constraints are modeled by FR and MR as **integrity** of global data objects so that those constraints are preserved at the global level. Global data object can be mapped to multiple local instances in different application and **names**. Each of subjects in the *Functional view* is mapped to a set of global data objects to which data items are assigned. The *Resource view* layouts how **hardware and software resource** are utilized and maintained in an enterprise. Applications, data items and computations occupy software resources. Software resources consist of modules and distributed among information infrastructures.

Consolidating the four views forms the GIRD. It is done by merging common constructs in different views. The GIRD is the schema of the Metadatabase. Metadata of local systems are populated into the Metadatabase with required knowledge to make the model functional. The details of the GIRD are given in Appendix A.

3.3. The Metadatabase system in action

As the Metadatabase is aiming for complex information resource modeling, extra effort have been paid for providing assistance to users to model operate and maintain the global model. **Information base modeling system (IBMS)** is a CASE tool help user model the enterprise information resources accordingly. Views are defined and manipulated visually and represented in metadata, which is ready to be populated into the Metadatabase. The prepared metadata are migrated to the GIRD by the **Metadatabase management system (MDBMS)**. Like a conventional DBMS, MDBMS interface the Metadatabase with outside components helping user operate and exploit the metadata stored. Users can navigate themselves among models, which represent the respective local system, and discover the details of system(s) without the hassle of physically dealing with local system(s). It also serves as a **global query system (GQS)**, which can target the metadata as well as the actual data located in the local systems. On top of the functionality as multi-database language, with the help of knowledge regarding local system, extensive on-line assistance can be provided making global query formulation easier. Knowledge in the Metadatabase are modeled by rules. Rules are either directly inference of the global model or user-defined. They are modeled and programmed by a **rule-oriented programming environment (ROPE)**. The ROPE interacts with local system and monitors their operations so that rules modeled inside the Metadatabase can be enforced.

The Metadatabase system can be implemented in three modes. (1) The passive mode serves as the enhanced data dictionary of the enterprise that the system does not actively interact with local system. It is also the basic step in building the enterprise model that enable the system fully functional in the following two modes. (2) The semi-active mode provides additional functionality on global query system such that users are allowed to

interact with local systems through the system. (3) The active mode of the Metadatabase serves as a coordinator of information resources inside an enterprise that facilitate interactions among local systems. Global rules are enforced making information are consistent through out the enterprise; i.e. the same identifier gives the same real-world object with consistent data items describing it. As a result, users can enjoy the synergy of integrated information resources at the global level.

3.3.Global query formulations and processing in the Metadatabase system

For the solution proposed in later chapters, modifications are mainly on the global query processing, a process in the global query system. A summary on the current global query processing will help us pinpointing what and how modification(s) to be made. Enhancements to the current Metadatabase for solving the problems will be based on the outlines given in Chapter 4.

A formal global query is specified as the following [CH96]:

$$\text{GQ} ::= (\text{A}, \text{D}, \langle \text{C} \rangle \mid \langle \text{M} \rangle)$$

Where,

A is the set of data items specified by user and determined automatically by the MDBMS (if necessary).

D is the domain of the global query. The MDBMS will determine the required data objects for the global query if the user does not specify all the required data objects during global query formulation.

$\langle \text{C} \rangle$ is a set of selection conditions and/or join conditions.

$\langle \text{M} \rangle$ are system metadata that minimally required by the global query.

The existing Metadatabase GQS has following steps:

- Global query formulation** Users can transverse through models in the enterprise visually, select items and specifies selection conditions with assistance from the Metadatabase.
- Join condition determination** The Metadatabase management system will determine joins if joins are not specified. A minimum set of data objects will be determined according to the items specified by the user. Then, the shortest path connecting the minimal set of data will also be identified.
- Global query processing** The formulated global query in the previous step is decomposed into a minimal set of local queries that will sufficient for the required result. Similarly, a minimal set of file, data objects in local systems, will be determined.
- Local query generation** Based on the set of files determined and metadata about the respective system, local queries formulated using local manipulation language will be generated and be dispatched.
- Query execution** Dispatched queries will be processed by the local systems. Query results will be returned to MDBMS for result integration.
- Result integration** Local query results returned are converted and assembled to the global format as the global query result.

These steps will be further elaborated in later chapters when modifications are made.

CHAPTER 4

Problem Outlines for Horizontal Partitioning and Its Variants

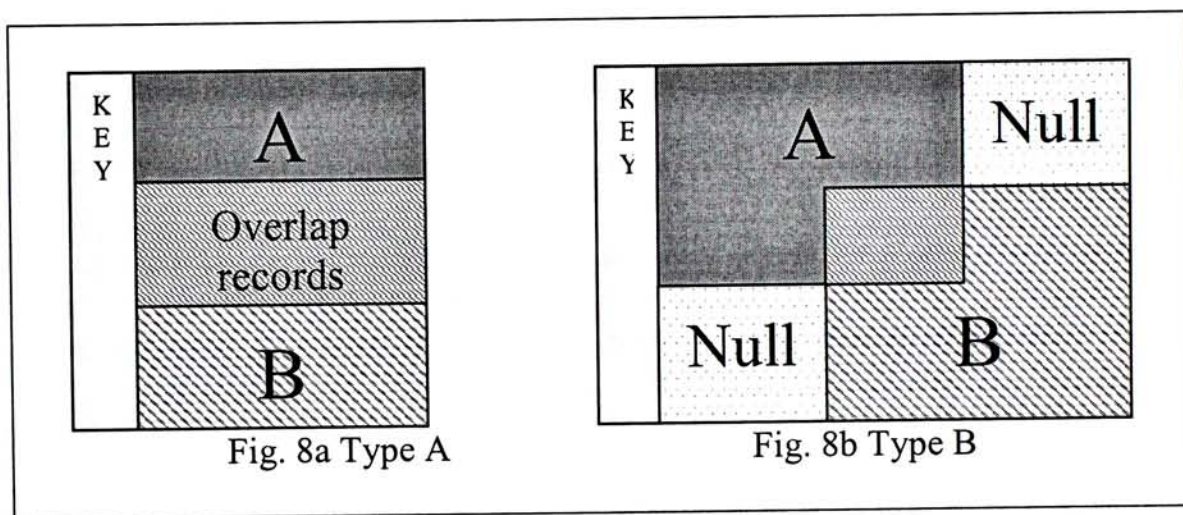
4.1. Horizontal partitioning

When semantically equivalent records are segmented in different in data objects, they are horizontal partitioned. It can be caused by distributed operation environment, replication and/or hardware/software limitations. Regardless of the causes, horizontal partitioned data objects can be beneficial. Horizontal partitioned data objects *facilitate parallel processing* as data objects can be distributed into different systems. Especially for production databases, which generate transaction data, front-line operations can be easily distributed into several systems and, hence, improve efficiency. When data objects are horizontally partitioned, the size of each partitioned can be controlled and alleviate hardware/software burden *making systems more agile and flexible*. Horizontal partitioned data objects can be *distributed and becomes more “reachable” for users*. “Supply creates its own demand”³ is a classical economic theory and its also apply to information utilization. Users are more willing use information resources that are in their proximity because of responsiveness and availability. After all, using the information resources during decision-making processes is the only way extract the real business values out of the information system. *Data become more secured* when they are horizontal partitioned. Remote-site replication, a form of horizontal partitioning, allows quick recovery from fatal system clashes. Other forms will prevent hackers and insidious users from having the full set of sensitive data at one spot. Horizontal partitioned data object can be created by local systems due to local autonomy. In other words, horizontal partitioning problem

³ Say’s law – Say J.B. (1821) A Treatise on Political Economy, London: Longmans.

is inevitable in and for a heterogeneous distributed environment. It increases the value of such environment but, at the same time, it poses difficulties in composing the global views for those partitioned data objects.

Currently, the Metadatabase treats all horizontal partitioned data objects as identical replicas. Schematically, data objects shares equivalent identifiers or having the equivalent identifiers as a part of composite key are considered as horizontal partitioned regardless of the attribute set they functionally imply. There are three patterns of horizontal partitioning:



Three types of *clusters*, a set of horizontal partitioned data objects, are depicted in Fig. 8. Type A horizontal partitioning is shown in Fig. 8a and that is the basic form of horizontal partitioning that attribute sets of the cluster are equivalent, not necessary identical. Overlapping occurs when two or more equivalent keys exist in the cluster assuming records are consistency in a cluster. If both A and B perfectly overlap to each other, that is a replication, a special case in horizontal partitioning. Fig. 8b shows the Type B horizontal partitioning. It is a generalized form of the Type A problem such that members in a cluster have equivalent identifiers but the sets of data items are only partially overlapped.

There is one more important piece of information has been often left out. That is the implicit concept that partitions a cluster of horizontally partitioned data objects. In our physical world, the tags on a file cabinet reduce our searching time dramatically. The way that the tags are put on is the implicit concept that partitions the files in a file cabinet. By the same token, the partitioning concept helps global query services minimizing query processing time. A partition concept might not exist at local system level. For example, there is not a field storing which sales center a customer belongs to in both local sites in our case. It is because the partition concept only matters at the global level. Therefore it is natural that the concept is not explicitly modeled in local schema. As a result, there is at least one partition concept that can be applied to a cluster of horizontal partitioned data objects. With all the overlapping partition concepts, it is important to assist users to sort out complex horizontal partitioned situations especially when the global model/schema is huge and complicated. However, these require complete knowledge of the enterprise and its sub-systems and are beyond the modeling capability of an integrated schema. The problem can be represented in relational algebra as shown in Appendix B1.

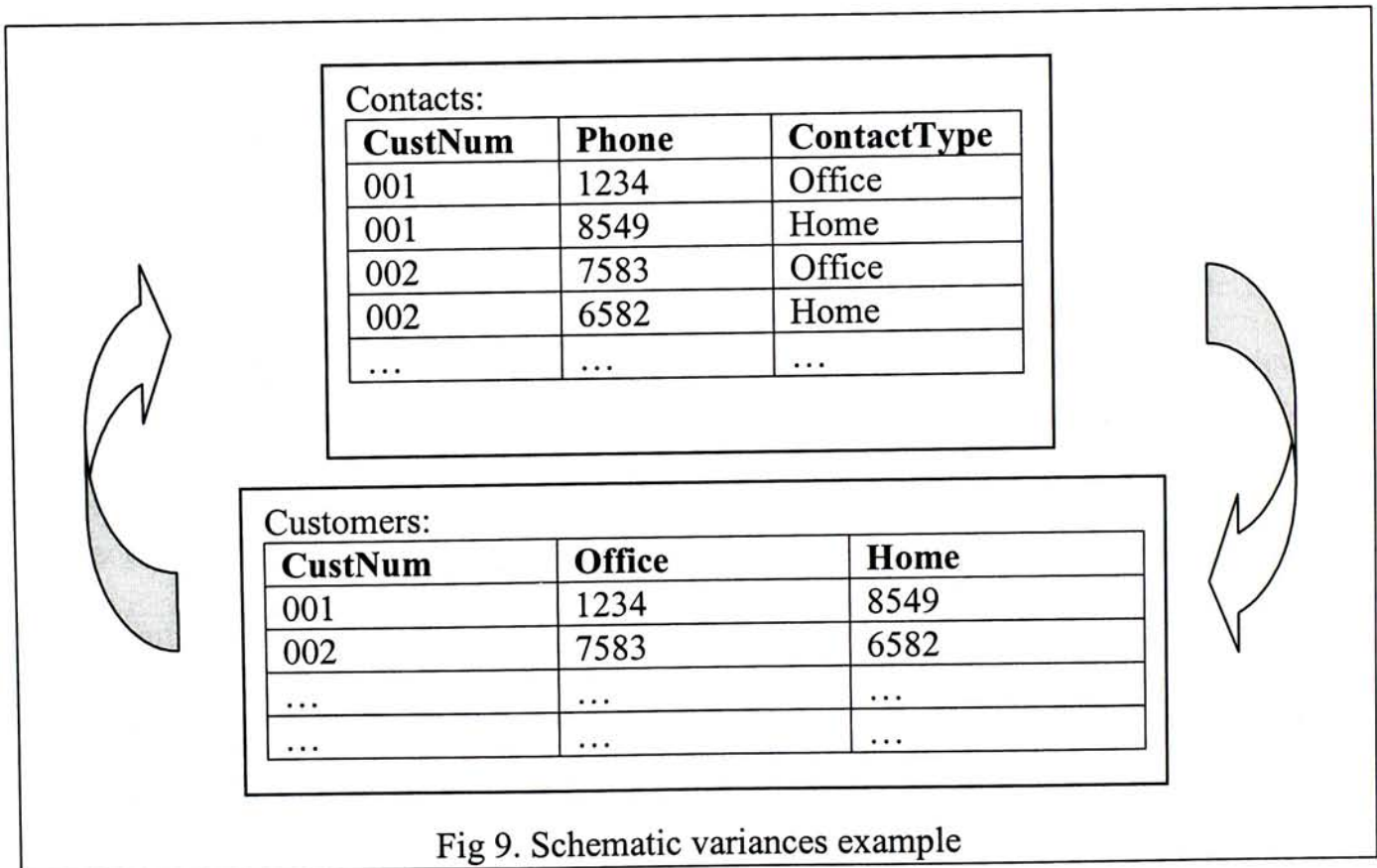
4.2. Level of abstraction

Different systems have different reasons for keeping the same information at different levels of detail to best fit the business environment. It has been called as schema isomorphism [ERS98] because it only induces structural changes at data object level but does not affect the overall schema of a system. These variations occur possibly because of business nature, user preferences, etc. Typical examples are on recording names, addresses and telephone numbers. Keeping addresses in different segments is more manageable than in a single field but there is no standard way to break down an address into different segments and that would cause confusion to users. After all, the information

to be stored does not change due to different structures of the data object. Yet, this complicates horizontal partitioning problems because a Type A problem will be treated wrongly as other types of problems. (Refer to Appendix B2 for the problem represented in relational algebra.)

4.3.Schematic variances

The schematic variances problems are characterized by difference in partitioning of the domain of a data item inside a data object. As a result, in relational terms, a subset of attributes (intensions) of a data object is in the domain (extension) of the other semantically identical data item in other data object. Visually, we can put customers' contact numbers in two ways (Fig. 9):



Both **Contacts** and **Customers** are virtually carrying identical information. *Office* and *Home* belong to the domain of *ContactType* in **Contacts**. (*ContactType* might be determined by an external data object.) On the other hand, the domain of *Phone* in

Contacts is split into *Office* and *Home* in *Customers*. When either one is chosen to be the global model, there will be field that never can be recovered or that without local counterpart. For example, if **Contacts** becomes the global representation, none of data item in **Customers** will match the domain of *ContactType*. Therefore, schematic variances introduce a complex domain mapping problem to database integration [SCG93, SK93, DR93]. (Refer to Appendix B3 to relational algebra representation)

4.4. Summary

Horizontal partitioning problem is one of the basic functionality of an HBDBMS. However, it is often overlooked by previous researches because of its apparent simplicity. The simplicity is rooted at the assumption that horizontal partitioned data objects are having identical structures. This assumption will be released in the next chapter. There are three types of horizontal partitioning identified in the chapter. A Type A problem is the basic form with equivalent attribute set and possible overlapping in a cluster. A Type B problem is a generalized form of a Type A problem with different attribute set and possible overlapping. The Type C problems are equivalent to vertical partitioning problem. By introducing the partition attribute to model the concept that partition a horizontal partitioned cluster, we can optimize global query processing by eliminating the irrelevant cluster members. In addition, horizontal partitioning problem is further complicated by introducing two variations: level of abstraction and schematic variances. These two variations only exist in the scope of horizontal partitioning as shown in Fig. 11. It is not different from a single database operation if a query does not target multiple partitions in a partitioned cluster. Data objects with different levels of abstraction cause confusion in distinguishing a Type A horizontal partitioning problem from another and rendering query result invalid. Schematic variances occur at the schematic level such that semantic equivalent data objects are having different schematic representation. Hence,

data objects cannot simply be cascaded; complex schema mapping is required when schematic variances happen in a horizontal partitioned cluster.

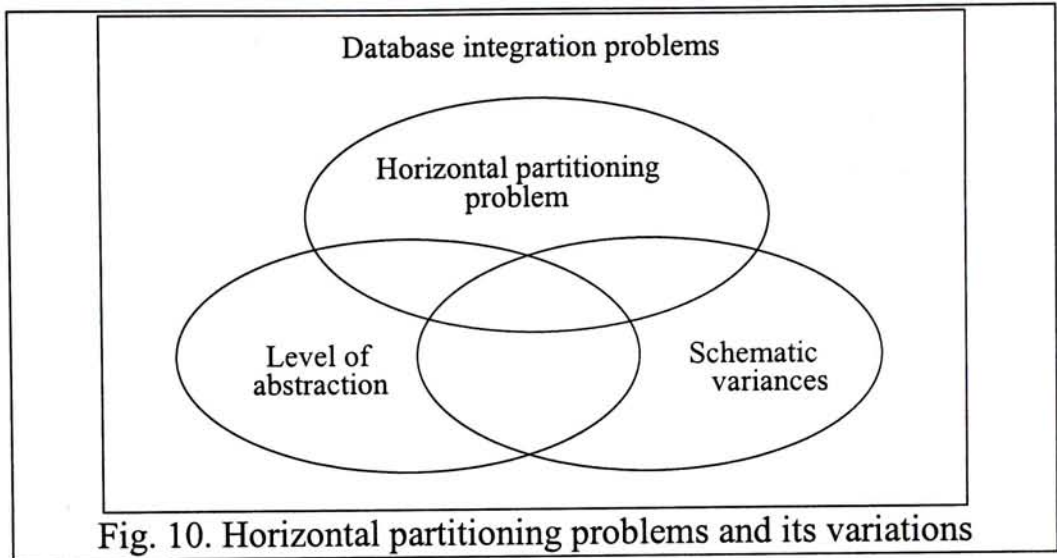


Fig. 10. Horizontal partitioning problems and its variations

4.5. The Scenario

For illustration purpose, discussion in this paper will be made around the sales settlement system of an enterprise. The system consists of 3 local systems: Sales Center A (Site A), Sales Center B (Site B) and the Head Office (HO). There are modeled using TSER constructs as the following:

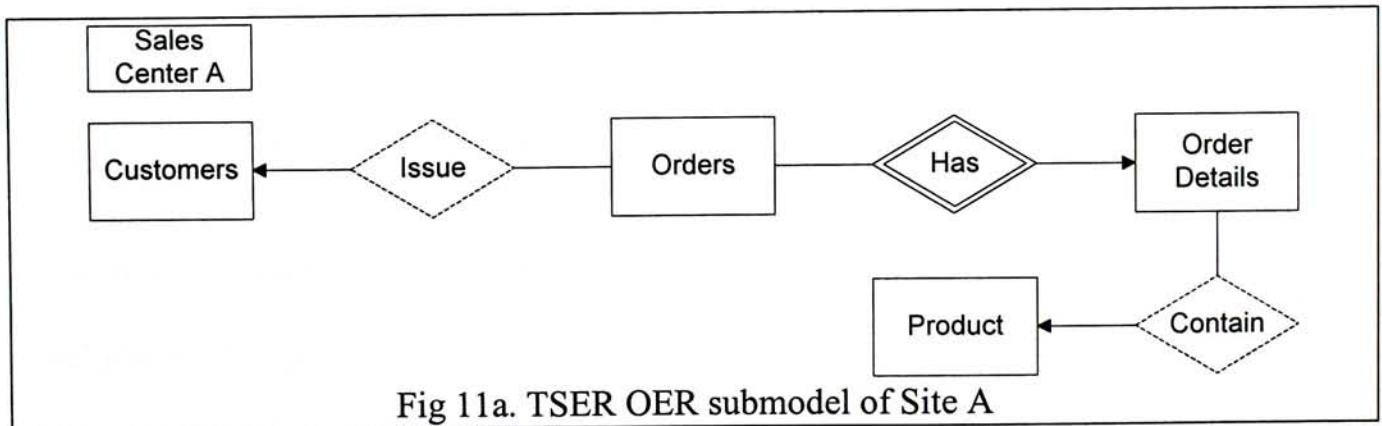
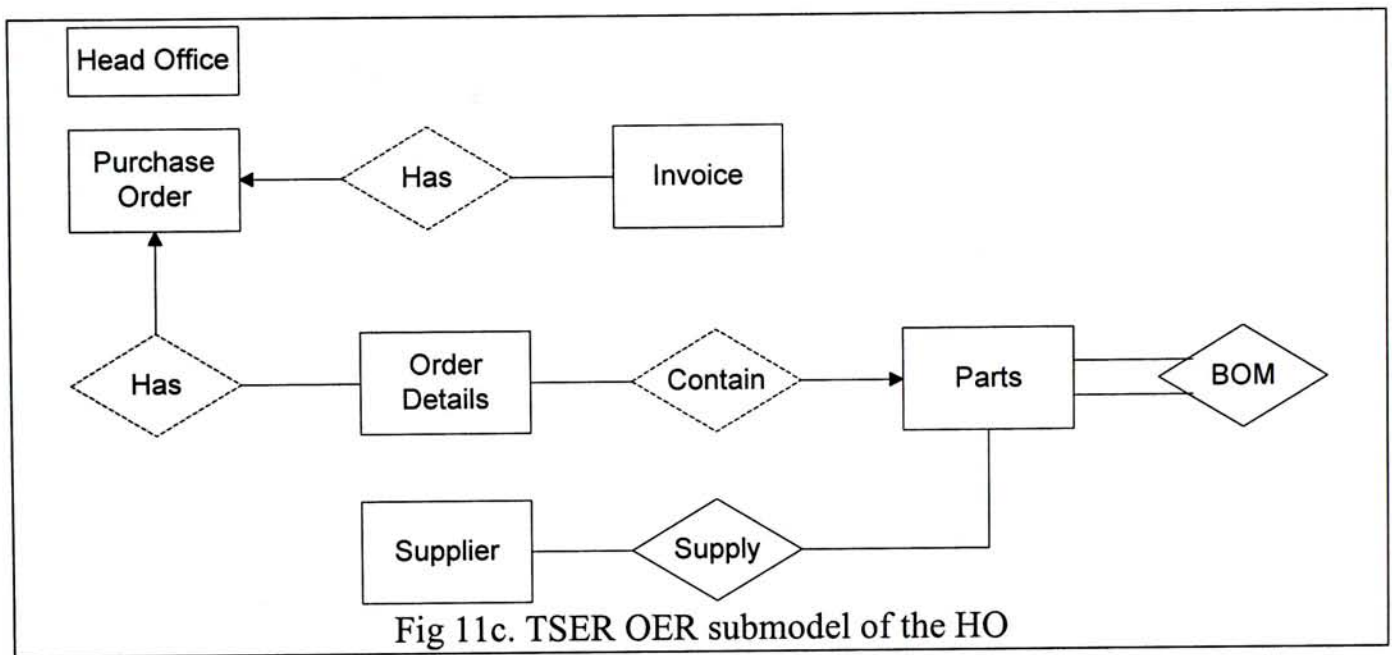
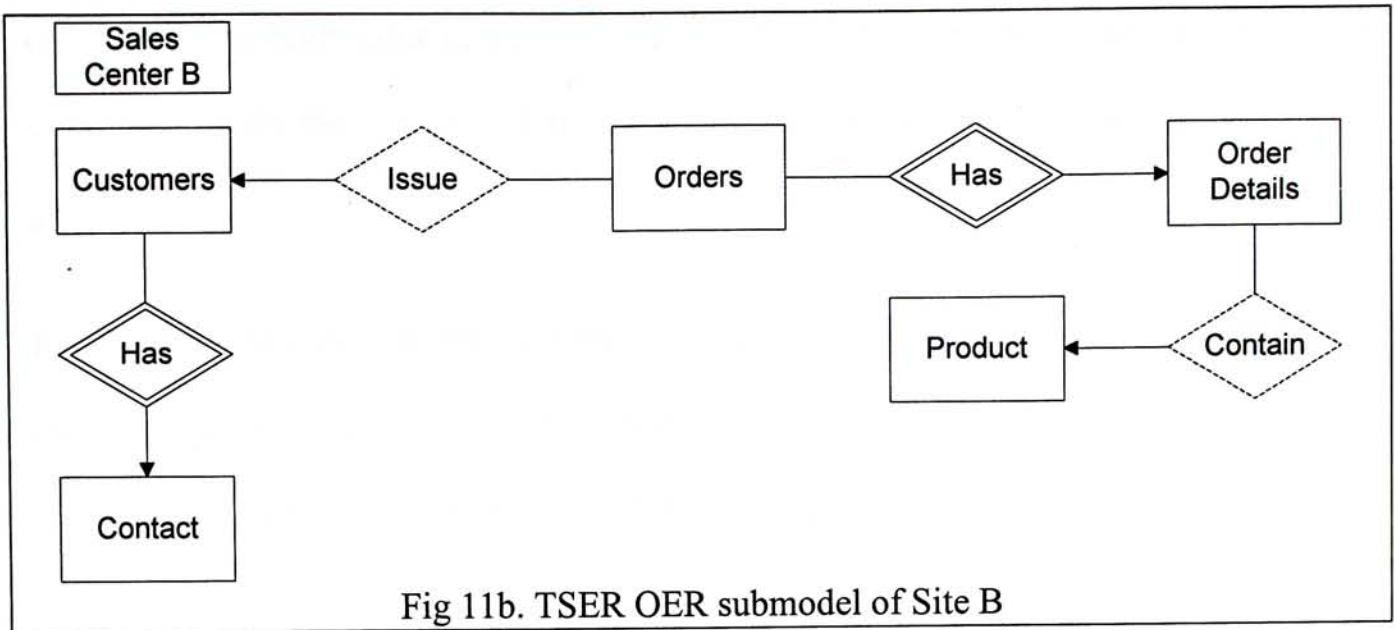


Fig 11a. TSER OER submodel of Site A



The two sales centers, A and B, (Fig. 11a and 11b respectively) keep their own customer and product bases and take customers' orders. The order processing sections are identical in both sales centers. Nonetheless, the schema of the two customer bases are different. Customers are kept in a single entity with all related information in Site A. Site B keeps customers' contact information separate from the customer entity so that Site B allows multiple contact numbers under a customer. The Head Office (Fig. 11c) performs central administrative processes. Product and supplier information is maintained for logistic reasons. The HO collects sales orders from the two sales centers also such that invoices can be issued. Therefore, Purchase_Orders and Order_Details are the unions of their own

counterparts in both sales centers. As a result, there are multiple options to retrieve information from the systems. (Refer to appendix C for details of local system and the global model.)

The three systems are then consolidated into a global OER sub-model. The consolidation process is done by merging constructs with identical primary identifiers as a single construct entity in the global OER sub-model [HBRY91].

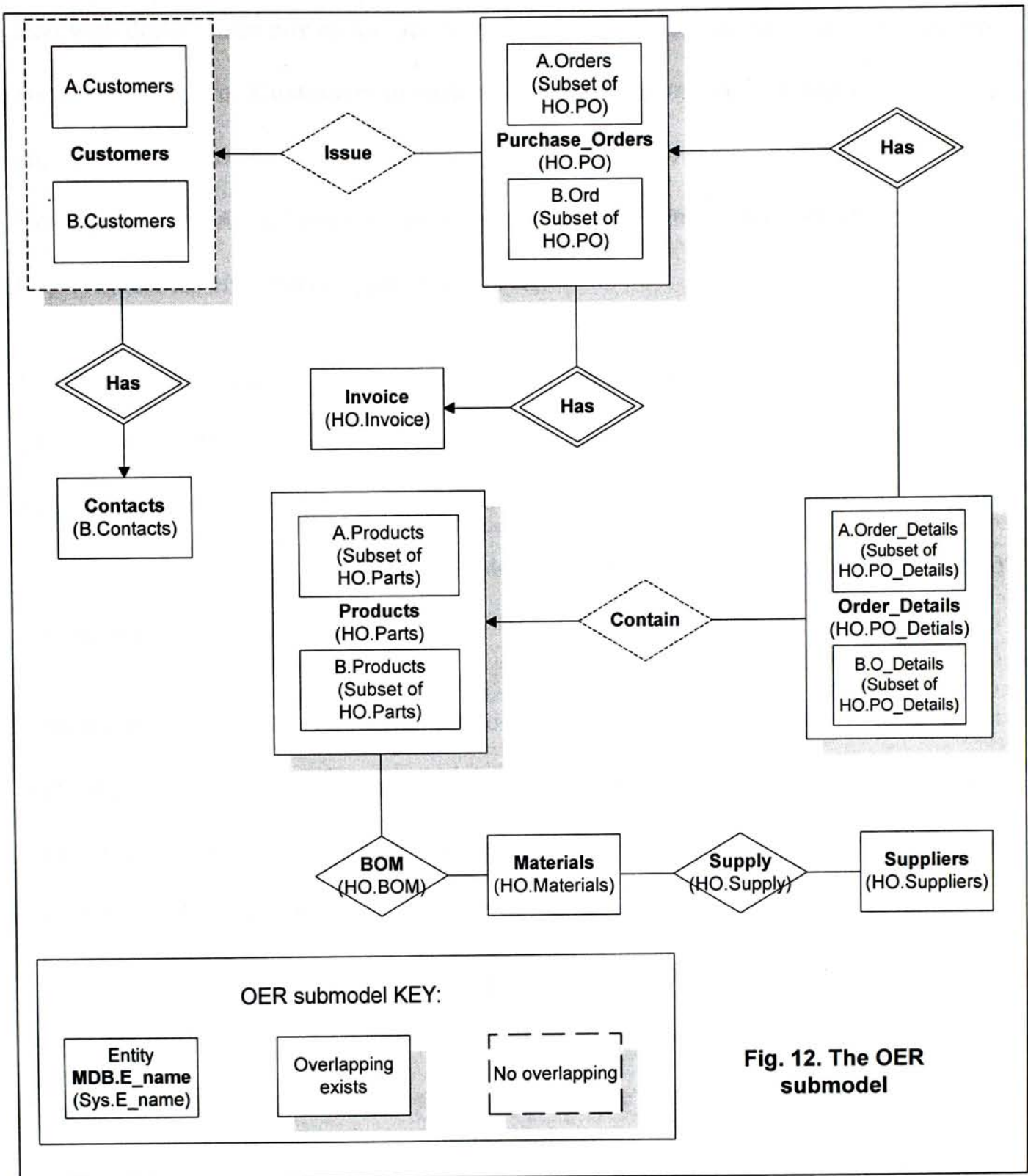


Fig. 12. The OER submodel

Constructs used in Fig. 12. are TSER constructs as shown in Table 1 and 2 in Section 3.1.. There are normal and floating icons for OEs to distinguish different types of horizontal partitioning patterns. All the normal icons represent that the local data object is not horizontally partitioned. There are two types of floating icons showing those horizontal partitioned local data objects. A floating icon with solid outer box shows that local data objects (partially) overlap with each other in subsystems. On the other hand, a floating

icon with dotted outer box means all the local partitions are disjoint. For example, there are two partitions of **Customers** in each of Site A (**A.Customer** – customer in Fig. 11a) and Site B (**B.Customer** – customer in Fig. 11b). And, there are three partitions of **Products** in Site A (**A.Product** – product in Fig. 11a), Site B (**B.Product** – product in Fig. 11b) and the HO (**Parts** – parts in Fig. 11c).

In our case, **Purchase_Order**, **Order_Details** and **Products** are Type A horizontal partitioned, as the attribute sets of the all members of the cluster are equivalent. **Customers** in our local systems are Type B horizontal partitioned. It is because apart from the equivalent primary keys, attribute sets in the both local systems are partially overlapped.

Customers are not only horizontal partitioned but also process both level of abstraction and schematic variance simultaneously. Customers' names are modeled as multiple and single fields in Site A and Site B respectively. The problem occurs at the data object structural level and is not shown at the schematic model level. In our global model, we choose to use multiple-field representation for addresses and names in our global model regardless of there are different level of abstraction in Site A and Site B. The different way to model customers' contact number is an example to the schematic variance problem and it can be identified at the schematic model of the systems. Types of contact numbers in Site A are modeled by three different fields in the customer entity while contact information is detached from the customer entity as a separate entity, **Contacts**, in Site B. **Customers** and **Contacts** has a one-to-many fixed relationship, i.e. a contact number only exists in **Contacts** if it is belongs to a customer exists **Customers**.

4.6. Populating the Metadatabase

Given global OER sub-model, we can populate the related parts of the Metadatabase with metadata. Application (subsystem) information is kept in **APPLICATION**. Each of the OE and PR in the global OER sub-model will be stored in **ENT-REL** as the global representation of local data objects and the respective local names will be in **NameAs**. Although there are multiple horizontal partitioned data objects exist in subsystems, users will see one and only one global representation of each of partitioned objects in the global model. Therefore, users are saved from struggling through system from system for the desired information. Metadata regarding data items in the global and the local OER-models are populated into **ITEM** with their respective equivalence metadata in **EQUIVALENT**.

However, the assumption behind is that all local objects modeled by an entry in **ENT-REL** are identical, i.e. assuming there is no horizontal partitioning exists in the local systems. When query is launched through the global query system in the Metadatabase, algorithms are set to minimize number of data object accessed with lowest accessing cost (time). During the process, horizontal partitioned data objects are treated indifferently [CH96].

Examples:

Given the global query formulation interface, users can easily launch a query:

```
SELECT CustName from Customer
```

The query result should give all customer name inside the customer base should be returned. However, the current GQS will only return the set of customers either in Site A or B, but not all of them.

Considered that the existence of horizontal partitioning data objects is so pervasive in a heterogeneous distributed environment and the strengths of the Metadatabase discussed,

the problem of processing global queries with horizontal partitioned data objects will be further investigated following the Metadatabase approach. The problems will be characterized using metadata. The set of new metadata will be included and modeled into the system along with necessary methods to enable the Metadatabase to resolve the problem.

CHAPTER 5

The Enhancements for Global Query with Horizontal Partitioned Data Objects

By analyzing queries on horizontal partitioned data objects in the coming sections, additional metadata will be identified. Then, methods are formulated to resolve problems during query processing with horizontal partitioned data objects. Metadata identified during the two phases will be summarized and modeled into the Metadatabase. Finally, results from different sections will be summarized in an example to conclude this chapter.

5.1. Identifying partitioned data objects

Currently, the MDBMS is not enabled with the power of detecting horizontal partitioned data objects. Global queries are formulated by browsing through the enterprise model with the current query engine. Users might specify required data items in a global query and left the Metadatabase to complete the query with technical details. The global query completion phase is to determine necessary components, which are not specified the user, for the global query. During the global query process phase, a minimal set of local files will be identified. Once the required set of local files is determined to be accessed, system-specific selection criteria and join conditions are inserted. Then, local queries can be formulated accordingly and be dispatched to the designated local systems for execution. In these processes, if multiple data sources are found they are treated as replicas. In our case model, a global query likes:

```
SELECT Ord_ID, Amt  
FROM Purchase_Orders;
```

It would only result contents from one of three partitioned *local files*, where data are stored in local systems. Yet, in our example, there are three partitions exist for **Purchahse_Orders**, namely: Order (Site A), Ord (Site B) and Purchase_Order (HO). It is because none of existing metadata is carrying information of horizontal partitioning regarding data objects. Therefore, it is the first step to accommodate metadata that would indicate if the local counterparts of a global data object are horizontally partitioned instead of replications of each other. **The additional metadata is to indicate the global object(s) selected in a global query is horizontally partitioned.**

New metadata required:

Metadata name	Data type	Description
Partitioned	Boolean	Indicates if a global data object is horizontal partitioned at local systems

5.2. Additional metadata for the horizontal partitioned data objects

As there are different partitions of data objects inside the enterprise when data are horizontal partitioned, users probably want to specify the partition(s) of data to be accessed. For this purpose, a global query will be formulated like this:

```
SELECT Prod_ID, ProdName
FROM Products
WHERE Sales_Ctr = 'A';
```

The user expects the query to result what are being sold in the sales center A as stated in the selection criteria. In other words, selection criteria are the hints for the Metadatabase to locate the required partition(s). With the new metadata identified, we know that **Products** is horizontally partitioned in three different sites. Yet, it is not sufficient to pinpoint whereabouts of a designated partition in local systems. Hence, additional metadata are needed for locating particular partition of data a local files.

In the above example, selection criteria with **Sales_Ctr** only valid at the global level as **Sales_Ctr** does not exist otherwise. It is because **Sales_Ctr** at the local level is implied by the local system itself. The Metadatabase is capable to model global items that do not exist in any of local systems as non-persistent items. The global data item **Sales_Ctr** models the partition concept that partitions the local files of the global data object **Products**. Global data items that model partition concepts are called partition attributes. Partition attributes are not restricted to be non-persistent items when the required information is modeled in local systems. The Metadatabase is transparent in making users not realizing that the additional partition attribute might not exist in the local data objects. The following query will possibly be launched against horizontal partitioned data objects:

```
SELECT *  
FROM Ord_Details;
```

As all data items are selected including partition attribute, it has to be screened out during the global query process phase to prevent issuing invalid queries to local systems as it does not exist in local systems. Then, the designated value of a partition attribute to respective subset of local query results is required to be appended as a derived data item during the result integration phase.

Partition attributes alone is not sufficient to limit local files will be accessed for a global query because there is no information inside the Metadatabase that matches the corresponding selection criteria. Partition conditions are needed to describe which partition of data is contained in a local file. They are modeled a new metadata using logical expressions in disjunctive normal form (DNF) because logical inferences can be easily made with selection criteria.

The advantages of having partition attributes and partition concepts in place are two folds: (1) it limits the potential search space of a global query making the query process more efficient; (2) it preserves the autonomy of local system. With these metadata stored in the Metadatabase, users can identify how a global data objects are partitioned and where its partitions scatter in the enterprise by launching queries on the Metadatabase instead of on local systems. In order to identify data sources, **additional metadata are required to associate local files to partition attributes and the designated values that validate the relevant selection criteria.**

New metadata required:

Metadata name	Data type	Description
Partition attribute (PA)	Numeric Synonym to Itemcode	Signifying an global item is a partition attribute tat models a partition concept
Partition condition (PC)	String	To store a set of well form formula (WFF) in DNF to valid selection criteria with PA involved

5.3.Complications of horizontal partitioning problem

In terms of solving the horizontal partitioning problem in previous database literature would come to an end here as any global query issued against horizontal partitioned data objects can be handled. The following two sections about the level of abstract and the schematic variance problems were dealt with as separate issues or simply ignored in the context of horizontal partitioning.

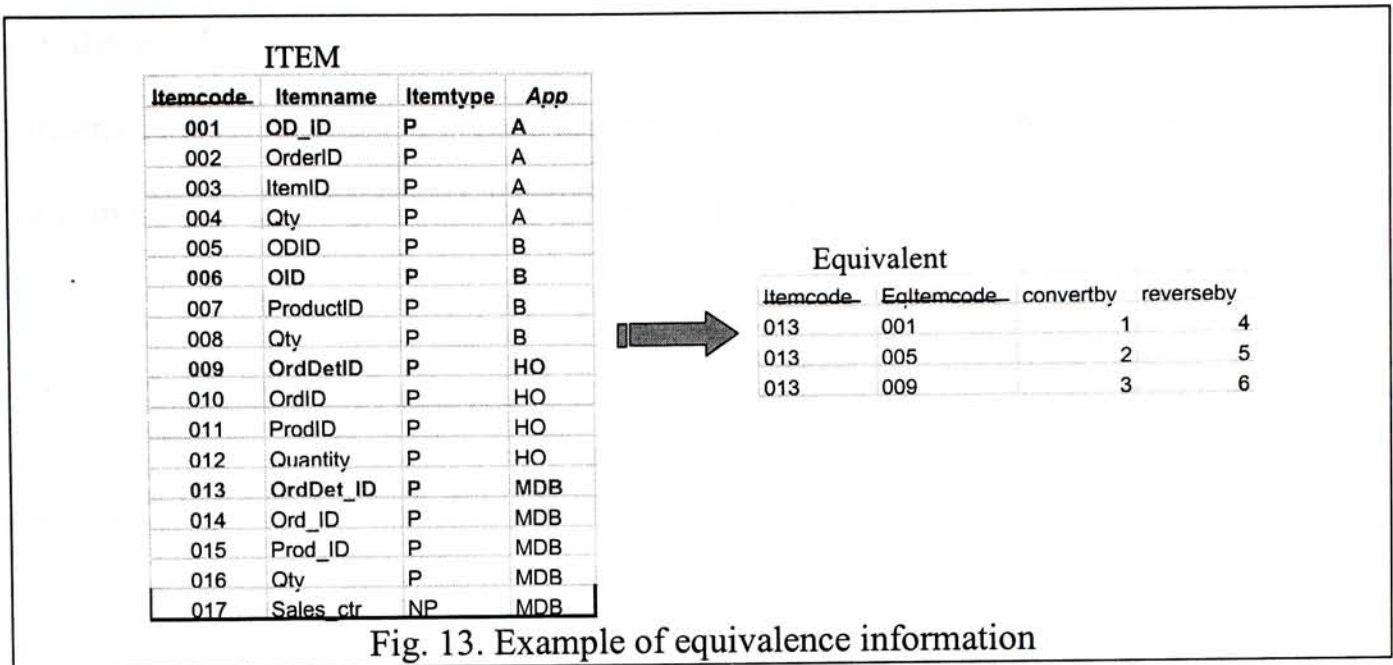
If the two variants are not resolved under to scope of the horizontal partitioning problem, it leads to the assumption that all partitioned data objects are identical in structure. For production database, those produce transaction data, data objects are often identical structure. Yet, this assumption is not necessary true in a heterogeneous distributed environment. For example, customers' names are modeled different in two of the sales

centers in our example system. That is very reasonable to model customer names using a single field for corporate accounts while using multiple fields for other retail customers. As a result, we need to concatenate *firstname* and *lastname* fields in A only when we want to cascade that to B in which the first and last names are put in a single field, or vice versa. Similar reasons are valid when we model data object with schematic variances.

5.3.1. Level of abstraction

Different levels of abstraction are the differences in how detail is information stored in data objects. Using less number of data items to capture the same information is said to have higher level of abstraction. For instance, customers' name in Site A are stored in a single field called **name** while the same information is broken down into two fields, **firstname** and **lastname** in Site B. That is, Site A has a higher level of abstraction on recording customers' name. The level of abstraction problem is matter of mapping between one and multiple attributes.

The current Metadatabase has equivalence information in converting from global items to local items, or vice versa, stored in the Meta-PR **Equivalent**. Nonetheless, the Meta-PR **Equivalent** that contains equivalence information takes only one parameter when converting/reversing items, i.e. a global item can be only mapped to a single local item at a time. The current structure of **Equivalent** follows:



In Fig. 13, the global item OrdDet_ID (013) has three local counterparts: OD_ID (001) in Site A, OID (005) in Site B and OrdDetID (009) in HO. All of them are 1-to-1 mappings.

When a global query like the following is launched:

```
SELECT firstname, lastname, street, district
FROM Customer;
```

There are problems when formulating the local queries for both Site A and Site B. It is because only parts of the global query find a one-to-one mapping **Equivalent** for the both local sites: **firstname** and **lastname** in Site A and **street** and **district** in Site B. However, customers' name is stored as a single field in Site B making only one of **firstname** or **lastname** can be mapped to name, a **many-to-one** mapping required. The similar mapping problem occurs for address information mappings to Site A. It introduces inconsistency due to incomplete mapping when only the one with loose end is selected in the query. On the other hand, the global model can model customer names using a single field and, in turn, we need one-to-many mapping instead.

The current equivalence information need to be enhanced so that a global item can be mapped to multiple items in a local object when the level of abstraction is higher in the global level. Otherwise, when the level of abstraction at the global level is lower than that

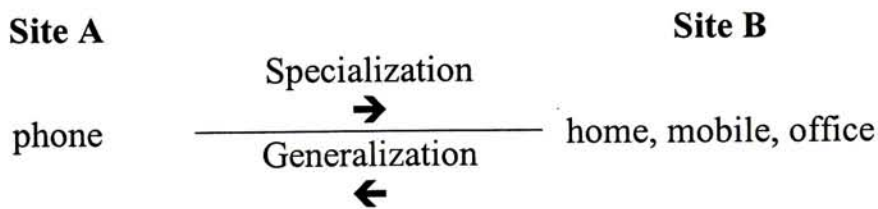
at the local level, multiple global items will be aggregated as one local item. To incorporate the additional equivalence information to the Metadatabase, a set of sequence numbers that signifies a set of data items with one-to-many or many-to-one relation. It also discloses the relative order when the set is passed into a conversion/reversion rule.

New metadata required:

Metadata name	Data type	Description
Convert_order Reverse_order	Numeric	Relative orders of global and local items when passing into conversion or reversion rules.

5.3.2.Schematic variances

Schematic variances occur when a domain of a data item in a system is split into several subsets and stored in different data items in other system – domain specialization – or vice versa – domain generalization. In our example global model, customers’ contact numbers are modeled as an entity in Site B so that a customer can has a (an unlimited) number of contact numbers (contact numbers are stored in **Phone**). Customers in Site A, however, customers are limited to have three contact numbers, which are modeled as three fields: **home, mobile** and **office**. The relationship among data items in both **Customers** in our example model can be shown as the following:



Normally database modelers would choose the Site B model as the better one. It is because structural changes, which induces more effort to do so, have to be made to the **Customers** entity in Site A when additional contact number is required for customers. As

database models are to fit the business environment, both modeling techniques are considered to be valid and we need to cater a solution for both cases.

By using TSER modeling method, identical keys are consolidated when integration OER submodels. This ensures that the global model will be modeled as Site B in case of both modeling techniques are used in local systems (as shown in our global model in Fig. 12). It is because both *local Customers* share the same key and will be consolidated while **Contacts** becomes an orphan attaching to the *global Customers*. As a result, the new metadata proposed in this section provide information on the complex relationship of domain mappings among local data objects to the global model. When a user issues a global query to obtain customer names and whose contact number(s):

```
SELECT firstname, lastname, phone
FROM Customers
WHERE Customer.CustID = Contact.ContID
AND ConType = 'Office' OR ConType = 'Mobile';
```

The global model can be directly mapped to Site B with the help of current set of metadata plus the additional metadata for level of abstraction specified in the previous section. We will focus on new metadata required for the mapping from the global model to a local system like Site A.

The complication, in the example, is that phone numbers are stored in three different data items in the **Customers** of Site A while we have only one data item in the global model. That is, the domain of the global data item **phone** is specialized into 3 fields in **Customers** of Site A. Schematic variances problems are problem of domain mapping (1-to-many or many-to-1) of data items among systems.

In addition, the three field names, **home**, **mobile** and **office**, or their equivalent forms, are in the domain of the global item **contype**, which distinguishes types of contact numbers

stored from one to others. With the help of **contype**, a specific subset of data in **phone** can be identified. If there are specialized domains in the local level match the subset of data identified, the local query should be formulated accordingly instead of combining all specialized domains indifferently for any global query issued. The required domain mapping information is not included in the existing equivalence information in the Metadatabase. As a result, the **meta-PR Equivalence** is further modified with a new metadata, which carries domain mapping information.

New metadata required:

Metadata name	Data type	Description
Domain mapping	Numeric	It indicates the domain relationship among data items.

5.4.Global query with horizontal partitioning data objects

To illustrate the impact of introducing horizontal partitioned data objects, we will go through the global query processes described in Section 3.3 again.

Global query formulation. This phase is to allow user issue global queries. Users are given a visual interface to transverse models and to select data items and the data objects from which data items are drawn. Data objects for a global query is optional when the users do not care or not know where data items are actually stored. The Metadata system will determine data objects required answering a global query automatically. Therefore, users are not necessary to specify the technical details of the query. Due to this feature of the Metadatabase, this formulation process is not affected by horizontal partitioned data objects because (1) they need not deal with data objects particularly and (2) horizontal partitioned data objects are bound by a corresponding global representation. (As shown in Fig. 12) In this phase an incomplete global query (IGQ) is resulted and is denoted as:

$$IGQ = (A^u, [D^u], [<SC>][<M>])$$

Where,

A^u is the set of data items selected by the user;

$[D^u]$ is the set of data objects specified by the user [optional];

$[<SC>]$ is the set of selection criteria given by the user [optional];

$[<M>]$ is the set of system metadata given by the user [optional].

Join condition determination and global query completion. This phase is to determine the missing information (if any), which are essential for global query processing, of global queries issued by users. With the IGQ issued by the user, the minimal set of data objects, O_{min} , that are necessary to answer the query will be determined by the Minimum-Set-of-ERs algorithm. When multiple data objects required, join operations are usually needed. Then, a shortest path is determined that joins all data object in the minimal set identified. Horizontal partitioned data objects do not affect the representations of global objects and relationship among them and hence, this phase is not affected as well. As a result of this phase, a complete global query (GQ) is formulated as the following:

$GQ ::= (A, D, <C>|<M>);$ or

Where,

A is the union of A^u and A^s , which are a set of data items that are implied by IGQ required for global query processing;

D is the union of D^u and D^s , which is the set of data objects determined by the global query system and it is required for the completion the GQ;

<C> is a set of expressions of selection criteria (given by the user) and join conditions (given by the user and/or determined by the system) or both; and

<M> is a set of metadata required to complete the global query.

Global query decomposition. The global query decomposition procedure is to break down the global query, GQ, into a set of local files and data items to be drawn from each of these files. As a result, local queries can be formulated afterwards. To improve the overall performance of global query processing in Meatdatabase, we decompose a global query into a minimal set of local files that is required to complete the global query.

The original process [CH96] did not take horizontal partitioning into account and that must be revised to process global queries with horizontally partitioned data objects. It identifies all possible local files for required data items. Then, the minimal file access list will be obtained followed by finding equivalent data items in each of the identified local files.

With the newly added metadata, *partitioned*, global data objects identified, D, can be categorized into 2 groups: horizontally partitioned and normal data objects. In turn, local files corresponding to those horizontally partitioned data objects can be found. By interfering the selection criteria, where required/excluded partitions are stated by users, in the GQ and partitioned conditions of local files, local files that contain no required partition or specifically excluded can be screened out. If the user did not state the required partition in the GQ, all found local files would be accessed.

Combinations of local files with minimum overlapping that resemble the required partitions will then be found. For all found combinations, none of the local files in it can be replaced by another member in that combination. In addition, one of those found combinations must be accessed so that data in local files of horizontally partitioned data objects can be retrieved correctly. However, the combination with the least number of local files is the minimal file access list only if the combination contains all required data items in the GQ. Otherwise, the source local files of data items that are not contained in each of the combinations will be found so that the minimal file access list can be guaranteed.

Then, for each of local file in the minimal file access list, required equivalent local data items will be identified. By using new metadata, converse/reverse order and domain mapping, data items with different level of abstraction and schematic variances can be accessed. Finally, reversion rules required for result integration will be appended, if necessary.

The global query decomposition can also be summarized into the following steps:

1. Identify local files, F'_{HP} , for horizontal partitioned data objects in D .
2. Given F'_{HP} , find combinations of $f'_i \in F'_{HP}$ that contains all required partitions required by the GQ and result C'_{HP} .
3. Determine the minimal file access list, F_{HP} , given C_{HP} and A .
 - 3.1. Given $c_{HP} \in C_{HP}$,
 - 3.2. Determine data items, $A_N \in A$ that has no equivalence in any of local files in c_{HP} .
 - 3.3. Determine the minimal set of files, F_N , containing A_N .
 - 3.4. Keep c_{HP} and F_N if $\|c_{HP}\| + \|F_N\| <$ previously known best solution.
 - 3.5. Return c_{HP} , as F_{HP} , and F_N .
4. Identify data item(s) to be accessed from each of local file in F_{HP} and F_N .
 - 4.1. For each local file, determine equivalent data items in the file.
 - 4.2. Find data items with different level of abstraction
 - 4.3. Find data items with schematic variances
5. Append rules for result integration.

1. Identify local files for horizontal partitioned data objects in D .

```

for each  $d_i \in D$ 
  if partitioned( $d_i$ ) then
     $F'_{HP} = F'_{HP} \cup \text{GET\_candidate\_partitions}(d_i, \langle SC \rangle)$ 
  end if
end for

Function GET_candidate_partitions( $d_i, \langle SC \rangle$ ) /*returns a set of local files*/
/* $d_i$  is a global data object*/
/* $\langle SC \rangle$  is the selection criteria in DNF =  $\{\langle sc_1 \rangle, \langle sc_2 \rangle, \dots, \langle sc_i \rangle, \dots\}$ */
 $P = \text{GET\_partitions}(d_i) /* \{p_1, p_2, \dots, p_i\} */$ 
 $PA = \text{GET\_PA}(d_i) /* \{pa_1, pa_2, \dots, pa_i\} */$ 
for each  $p_i \in P$ 
   $\langle PC \rangle = \text{GET\_PC}(p_i) /* \{\langle pc_1 \rangle, \langle pc_2 \rangle, \dots, \langle pc_i \rangle, \dots\}$  in DNF*/
  for each  $\langle pc_j \rangle \in \langle PC \rangle$ 
    Return_this_partition = FALSE
    for each  $\langle sc_k \rangle \in \langle SC \rangle$ 
      for each  $pa_i | (pa_i \in \text{GET\_dataitems}\langle sc_k \rangle$ 
        and  $pa_i \in \text{GET\_dataitems}\langle pc_j \rangle)$ 
        /*GET_dataitems returns a set of data items used in

```

```

                                a set of expression*/
                                if overlap(range_of(<sck>, paj), range_of(<pcj>, paj)) =
                                TRUE then
                                /*Check if range of paj in <sck> overlap with that of
                                <pcj>*/
                                Return_this_partition = TRUE
                                end if
                            end for
                        end for
                    if Return_this_partition = TRUE then
                        Result_set = Result ∪ <pcj>
                    end if
                end for
            end for
        return Result_set
    end GET_candidate_partitions

```

2. Given F'_{HP} , find combinations of $f'_i \in F'_{HP}$ that contains all required partitions required by the GQ and result C_{HP} .

```

for each  $f \in F_{HP}$ 
     $C_{HP} = C_{HP} \cup \text{Get\_combos}(f, F_{HP} - \{f\}, \emptyset)$ 
end for

Get_combos(head, rest, COMBO)
    if head =  $\emptyset$  then
        return  $\emptyset$ 
    else
        if rest =  $\emptyset$  then
            return COMBO ∪ {{head}}
        else
            for each resti ∈ rest
                Get_combos(resti, rest - {resti}, COMBO)
                for each {combo} ∈ COMBO
                    if GET_partitions({head}) ⊇
                       GET_partitions({combo})
                       /*GET_partitions returns PC of a set of local
                       files*/
                       return COMBO ∪ {{head}} - {combo}
                    else if GET_partitions({head}) ⊆
                       GET_partitions({combo})
                       return COMBO ∪ {combo}
                    else
                        return COMBO ∪ {combo ∪ {head}}
                    end if
                end for
            end for
        end if
    end for
end if

```



```

    end if
end Get_combos

```

3. Determine the minimal file access list from C_{HP} .

```

for each  $c_{HP} \in C_{HP}$ 
   $A_N = A - \forall a(\text{Equivalent}(a) \in f_{HP}) \mid (a \in A \wedge f_{HP} \in c_{HP})$ 
  if  $A_N \neq \emptyset$  then
    Min_files =  $\|c_{HP}\| + \|A_N\|$ 
    Current_set = Get_min_sets( $A_N$ )
    if  $\|Current\_set\| < \text{min\_files}$  then
       $F_{HP} = F_{HP} \cup c_{HP}$ 
       $F_N = F_N \cup Current\_set$ 
    end if
  end if
end for

Get_min_sets( $A$ )
  for (each  $a \in A$ )
     $F_a = \text{get\_files}(a)$ 
    if  $\|F_a\| = 1$  then
       $A = A - \{a\}$ 
       $F_{min} = F_{min} \cup F_a$ 
       $A = A - \forall a' \exists f(a' \in f \mid (a' \in A) \wedge (a' \neq a) \wedge (f \in F_a))$ 
    end if
  end for
  if  $A = \emptyset$  then
    return  $F_{min}$ 
  else
    return Get_min_sets( $F_{min}, A, \emptyset$ )
  end if
end Get_min_sets

Get_min_sets_h( $F_{min}, A, \text{results}$ ) /*results=0 initially*/
  Current_best =  $\|A\| + \|F_{min}\|$ 
  Select a from A;
   $F_a = \text{get\_files}(a)$ 
  for ((each  $f \in F_a$ ) and ( $F_a \neq 0$ ))
    if  $A - \{a\} \neq \emptyset$  then
      if  $\|F_{min} \cup \{f\}\| < \text{current\_best}$  then
        results =  $\{F_{min} \cup \{f\}\}$ 
        Current_best =  $\|F_{min} \cup \{f\}\|$ 
      else if  $\|F_{min} \cup \{f\}\| = \text{Current\_best}$  then
        results = results  $\cup \{F_{min} \cup \{f\}\}$ 
      else
        Get_min_sets_h( $F_{min} \cup \{f\}, A - \{a\}, \text{results}$ )
      end if
    end if
  end for
end if

```

```

end for
end Get_min_sets_h

```

4. Identify data item(s) to be accessed from each of local file in F_{HP} and F_N .

4.1 Find equivalent data item, E , of $a \in A$ in each of f in F_{HP} or F_N .

4.2 Attach local data item(s) with different level of abstractions to f , given E and a .

```

if Rord(E, a)  $\neq$  0 and DM(E,a) = 0 then
    A = A - {a}
else if Cord(E, a)  $\neq$  0 and DM(E,a) = 0 then
    {a'} = {a}  $\cup$  LA( E, f, Crule)
    /*LA returns a set of global data items that are mapped to a local
    one*/
    A = A - {a'}
else
    If DM(E,a) = 0 then
        A = A- {a}
    end if
end if
Af = Af  $\cup$  E

```

4.3 Attach local data item(s) with schematic variances to f , given E and a .

4.3.a For specialization, drop local items that are not selected according to the $\langle SC \rangle$ in the GQ .

```

if DM(E, a) = 1 then /*specialization*/
    b = binding_item(E) | (DM = 3)
    /*binding_item returns a item given a set of equivalent data
    item*/
    if b  $\in$  GET_dataitems( $\langle SC \rangle$ ) then
        E = E -  $\forall e(\neg$  satisfy (e,  $\langle SC \rangle$  | b)
    end if
end if

```

4.3.b For generalization, attach file-specific selection criteria to limit the resulted domain of E in local query for f .

```

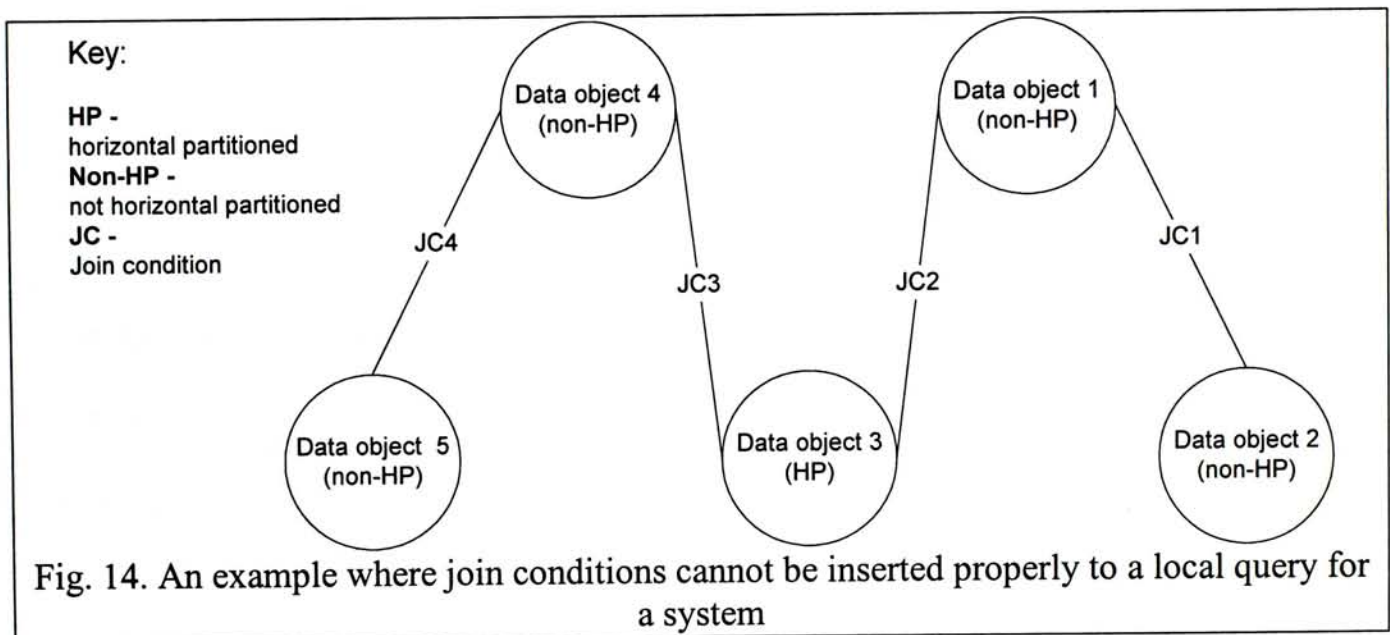
if DM(E, a) = 2 then /*generalization*/
    G = Equivalent(E)
    b = binding_item(E) | ((DM = 3)  $\wedge$  f)
    for each ((g  $\in$  G) and (g  $\in$  A)) then
         $\langle SC \rangle_f$  = "  $\vee$  " & b & " = " & Covert(g |GET_rule(b)
         $\wedge$  g  $\in$  G)
    end for
end if

```

```
end for  
end if
```

5. Append required rules for result integration.

Local queries formulation. After the minimal set of local files to be accessed is identified, system-specific $\langle\langle SC \rangle\rangle$ and $\langle JC \rangle$ are required for formulating local queries. Each of these local queries targets only one local system. The original algorithm first inserts system-specific $\langle SC \rangle$ and $\langle JC \rangle$ to groups of local files that are in the same source local systems. Such that, join operations can be performed at local level and hence the advantage of distributed processing is leveraged.



Consider the situation as shown in Fig. 14., there are five data objects involved in a GQ. Among them, four are not horizontal partitioned. There are four different join conditions needed among these local data objects. Assuming that there are 2 local systems needed among these local data objects. Assuming that there are 2 local systems containing all the files we want: one with local files of data object 1, 2 and a partition of 3 while the other system having the rest of required. Hence, performing join operation at local level will return incorrect results because there are only parts of required data in data object 3 is in each of systems. As a result, JC2 and JC3 cannot be inserted although local

files of data object 3 are in the same systems as those of non-horizontal partitioned data objects.

The algorithm for this phase is modified to process GQ with both horizontal partitioned and non-horizontal-partitioned data objects involved. It processes local files for horizontal partitioned data objects and those for others separately. Steps are:

1. Group members of F_N by targeted local system.
2. Apply system-specific selection criteria and join conditions to each group.
3. Apply system-specific selection criteria to members in F_{HP} .
4. Formulate local queries for each group and each member of F_{HP} .

Local query generation. Given that F_N , with system-specific $\langle SC \rangle$ and $\langle JC \rangle$, and F_{HP} , with system-specific $\langle SC \rangle$ are resulted in the previous phase, local queries are generated according to local system specifications. For SQL-bases system, local queries for non-horizontal-partitioned data objects will be formulated as the following:

```
SELECT a1i, a2i, a3i, ....  
FROM f1i, f2i, f3i, f4i, ....  
WHERE  $\langle SC \rangle^i$  AND  $\langle JC \rangle^i$ 
```

For horizontal partitioned data objects:

```
SELECT a1j, a2j, a3j, ....  
FROM f1j  
WHERE  $\langle SC \rangle^j$ 
```

Result integration. During the result integration phase, results returned from local queries are assembled by applying global join conditions. Local query results from all partitions of a horizontal partitioned data object, $R_{HP} = \{r_1, r_2, \dots, r_n\}$, are integrated first on a pair-wise basis until a single view is resulted. The operation can be done by the following SQL statement:

$$R_{HP} = R_{HP} \cup (r_i \text{ LEFT JOIN } r_j \text{ UNION } r_i \text{ RIGHT JOIN } r_j) - \{r_i\} - \{r_j\}$$

where r_i and r_j are local query results and $i \neq j$. Then join conditions determined in previous phase is applied to all local query results and results the answer of the global query.

With the revised procedures, the Metadatabase is extended with the capability to processing global queries against horizontal partitioned data objects, i.e. 2.II and 2.V in Fig 13. As mentioned, MDBMS ensure inter-subsystem consistency, i.e. identical values of equivalent record identifiers are mapped to the same real-world object and the values of equivalent attributes across local systems are consistent. This gives MDBMS a major advantage over other integration methodologies on handling horizontal partitioning. It is because reversing the attribute values of a set of records to determine consistency is greatly depend on system taxonomy or probabilistic inferences [BHP94, PRSL93, LP93, RR95, AKWS95]. Yet, ensuring consistency among local systems is out the scope of this paper. In the following method presented, we assumed consistency is enforced.

5.5.Housing the new metadata

With new metadata for handling horizontal partitioning problem identified in previous two sections, we need to modify the current GIRD definition to put the new metadata in action. The objective is to minimize the impact of these modifications to the existing framework, methods and most importantly, integrity of the Metadatabase.

New meta-attributes:

Meta-attribute name	Attribute Type	Meta-entity	Domain
Partitioned	Boolean	ENT-REL	T/F
PA	Numeric	NameAs	Null/Itemcode
PA-expr	String	NameAs	Characters

Putting **Partitioned** into the meta-entity **ENT-REL** gives users extra information on whether a data object is partitioned. **Partitioned** is set to TRUE if a global data object is horizontal partitioned or vice versa. The **NameAs** meta-entity keeps the mapping of the global data objects to the local name. **PA** is added to **NameAs**, as a part of the composite

key, so as to group clusters of horizontal partitioned data objects. **PA** is synonym of **Itemcode** and is valued Null if a data object is not horizontal partitioned. Closely related to **PA**, a new meta-attribute **PA-expr** is added to **NameAs** as well. It is a set of well-form-formula (WFF) with format of

$$[\mathbf{PA}] \langle \mathbf{operator} \rangle \langle \mathbf{value} \rangle$$

where,

[PA] is a dummy section that structures a WFF and it will be replaced by proper local item name during the local query generation phase;

$\langle \mathbf{operator} \rangle \in \{<, =, >, <=, >=, \neq\}$;

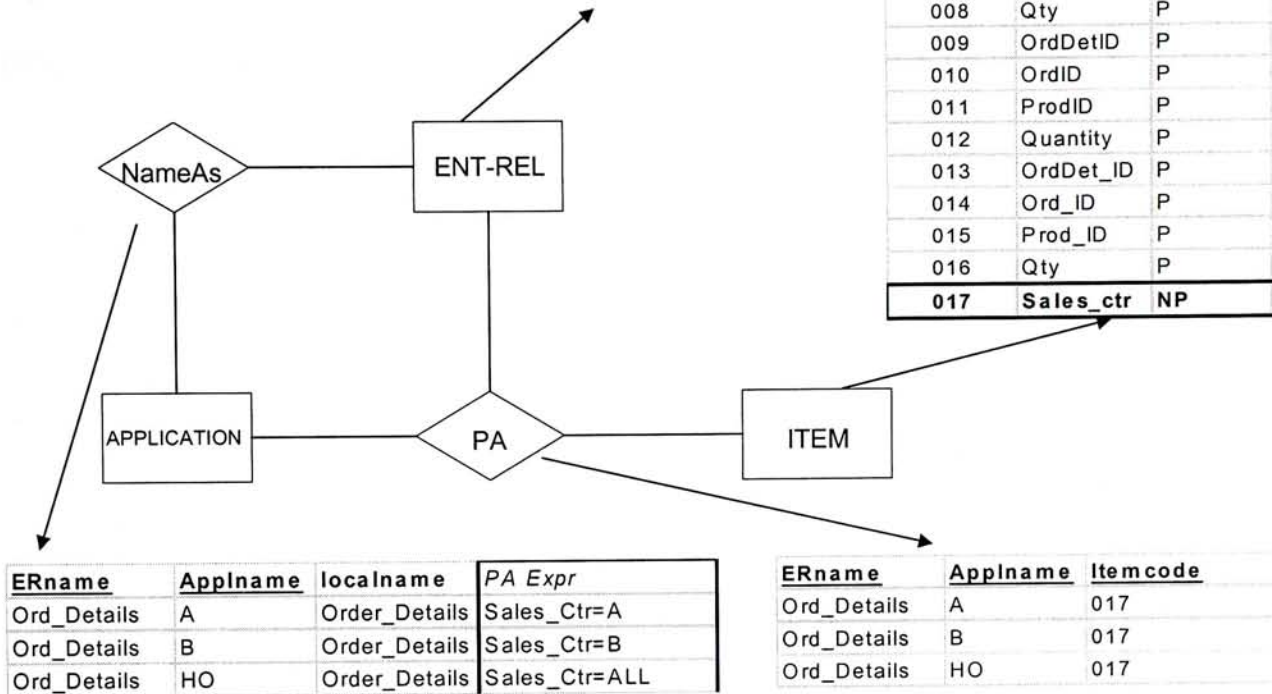
$\langle \mathbf{value} \rangle$ is the value assigned by the modeler that states the domain(s) of a local data object. *ALL* is an artificial value assigned to those data object that is valid for any **PA-expr**. If the domain of a local data object covers more than one segments of the **PA** that partitioned the cluster logically, DNF of the WFFs will be stored in **PA-expr**.

Users can issue queries against the Metadatabase instead of sub-systems. These queries will return information on the enterprise information systems that are modeled in the Metadatabase instead of actual data stored in local systems. Therefore, with these additional meta-attributes, users can identify horizontal partitioned data objects in the enterprise. According the new structure, the Metadatabase will be populated for the Order_Details as the following (only related part of GIRD is shown):

Revised representation

Itemcode	Itemname	Itemtype	App
001	OD_ID	P	A
002	OrderID	P	A
003	ItemID	P	A
004	Qty	P	A
005	ODID	P	B
006	OID	P	B
007	ProductID	P	B
008	Qty	P	B
009	OrdDetID	P	HO
010	OrdID	P	HO
011	ProdID	P	HO
012	Quantity	P	HO
013	OrdDet_ID	P	GIRD
014	Ord_ID	P	GIRD
015	Prod_ID	P	GIRD
016	Qty	P	GIRD
017	Sales_ctr	NP	GIRD

ERname	ertype	akey	Partitioned
Ord_Details	OE	ODID	Y



ERname	Appname	localname	PA Expr
Ord_Details	A	Order_Details	Sales_Ctr=A
Ord_Details	B	Order_Details	Sales_Ctr=B
Ord_Details	HO	Order_Details	Sales_Ctr=ALL

ERname	Appname	Itemcode
Ord_Details	A	017
Ord_Details	B	017
Ord_Details	HO	017

Fig. 15 Example of populating new metadata (partial)

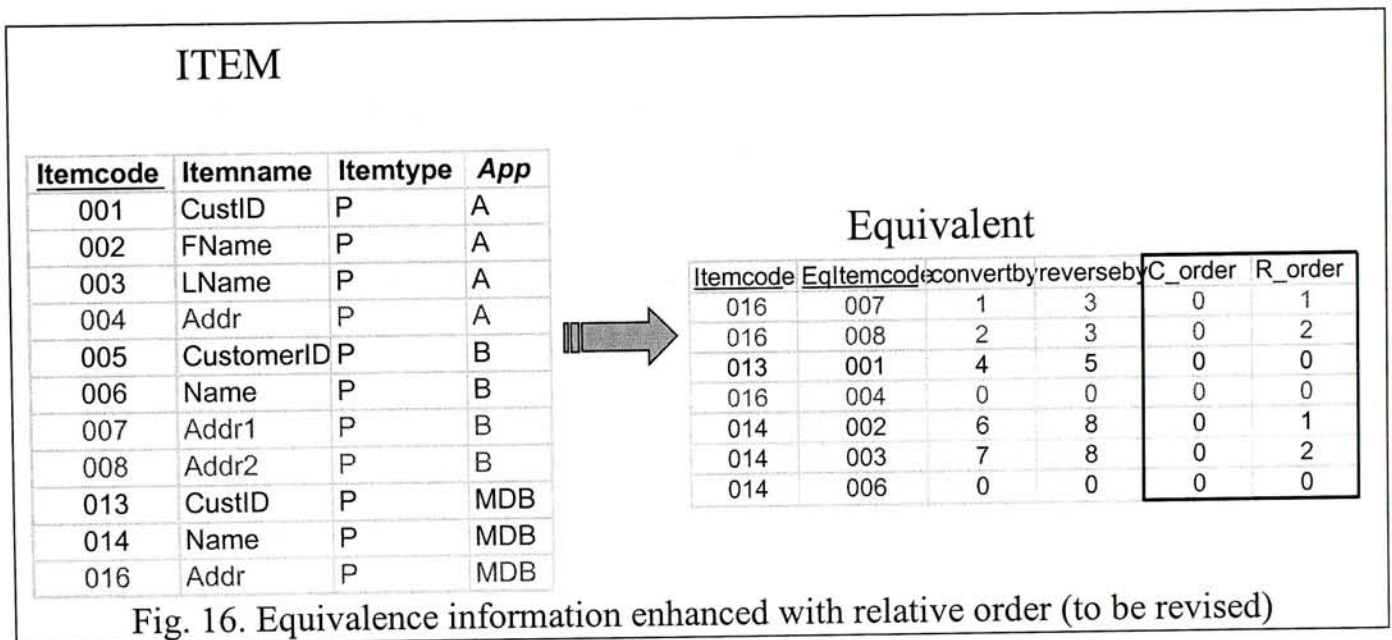
In Fig. 15., the global data object Ord_Details is signified to be horizontal partitioned by the new metadata **Partitioned**. The partition attribute, Sales_ctr, that partitions Ord_details is added to ITEM with itemcode equals 017. Two new metadata are added to NameAs, where local names of a global data object in different applications are stored. They are to model how Ord_Details are partitioned into local systems, i.e. the designated subset(s) of data carried by each of the local partitions.

To incorporate the new metadata identified for tackling the variations of horizontal partitioning problem, we first tabulate them as the following:

New meta-attributes:

Meta-attribute name	Attribute Type	Meta-entity	Domain
C order	Numeric	Equivalent	Integer
R order	Numeric	Equivalent	Integer
DomainMap	Numeric	Equivalent	Integer

C/R_order keeps the relative sequences (for its value > 0) of a set of item when they are passed into a conversion/reversion rule. A zero is assigned if the rule takes only one parameter. Therefore, by **C/R_order**, not only identifies a set of items that maps to each other with a pre-defined rule, it also helps passing the right parameter to the right slot when triggering rules. An example of populating the new **C/R_order** into the Metadatabase is shown in Fig. 17. *FName* and *LName* from application A is concatenated (by a rule) as the global data item *Name*.



In Fig. 16, Rule#3 takes two data items in system A, item 002 and item 003, as parameters and converts the local data items to the global data item *Addr* (016) in MDB.

DomainMap is a numeric meta-attribute that signifies the type of domain mapping among equivalent data objects discussed in the pervious section. When a mapping from a global data to local data items are 1-to-m (or vice versa), it can be either a problem of different in level of abstractions or a schematic variance problem. The determinant of the type of the variant is the value of **DomainMap**: 1 - the domain of the global data item is a specialization of that of the local data item; 2 - the domain of the global data item is a

generalization of that of the local data item; 3 – the equivalent data item is in the domain of the data item; 0 – domains of the global and local items are disjoint.

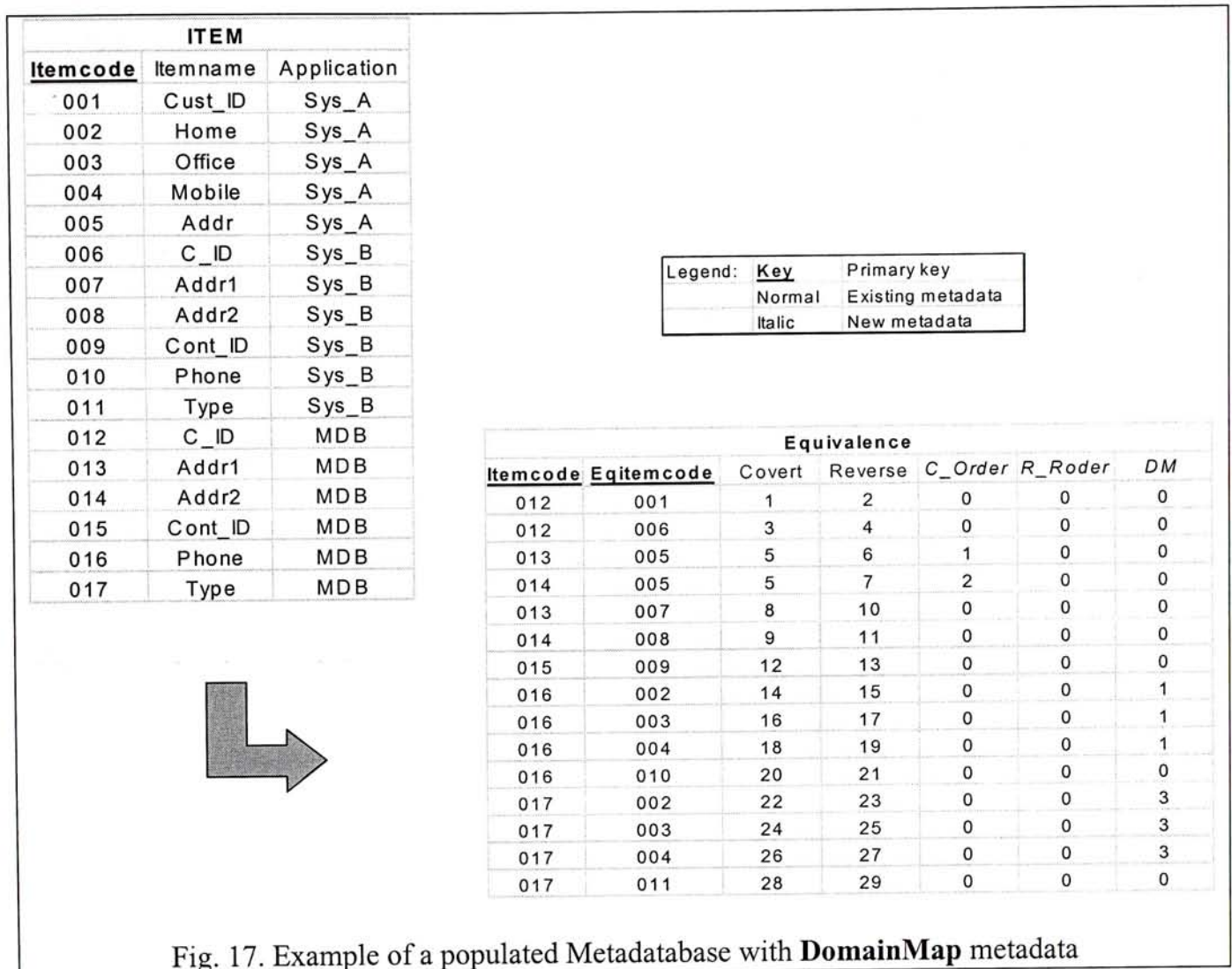


Fig. 17. Example of a populated Metadatabase with **DomainMap** metadata

As shown in Fig. 17, the domains of local data items, **home** (002), **office** (003) and **mobile** (004), are subsets to the global item **phone** (016). At the same time, items 002, 003 and 004 are in the domain of item **contype** (017) showing that it is a schematic variance problem. Separate local queries will be issue targeting local items 002, 003 and 004 to retrieve the corresponding data for global data item 016.

5.6.Example

To summarize that modifications and enhancements made to the current Metadatabase, the revised GIRD definitions are in appendix A. An example is provided with detail walk-through of the methods to conclude this chapter.

Example:

```
SELECT firstname, lastname, street, district, phone
FROM Customers, Contacts
WHERE Customers.CustID=Contact.ConID AND
ConType = "Office";
```

The purpose of this query is to find all customers' office contact numbers. By going through the global query formulation and join condition determination, the minimal set of global data object is determined to be {Customers, Contacts} and attributes (data items) $A = \{firstname, lastname, street, district, phone\}$. Both elements in data objects required in this GQ are horizontal partitioned by checking the meta-attribute **Partitioned** in ENT-REL. Hence,

1. $F'_{HP} = (A.Customer, B.Customer, B.Contact)$
2. $C'_{HP} = (A.Customer, B.Customer, B.Contact)$
3. As there is one combination can be found such that $C_{HP} = C'_{HP}$
4. As all required data items are in C_{HP} such that C_{HP} is the minimal file access list

To determine which data items are drawn from the local data objects, the following table can be formed by consulting the Equivalent of the Metadatabase.

Global	Site A – Customers
Data item	Data item
Firstname	Firstname
Lastname	Lastname
(street, district)	Addr
phone	Home, office, mobile

For Site A, as "contype = "office"" is given in the GQ, home and mobile is dropped from the data item list and this selection criteria will not be passed onto the local query for Site A.

Global	Site B – Customers
Data item	Data item
(firstname, lastname)	Name
Street	Street
District	District

For Site B, difference in level of abstractions between the global and local customer name is identified.

During the local query decomposition phase, two local queries is formed. For Site A, (SiteA.customers, {firstname, lastname, addr, office}, NULL). There are two elements in L_{Op} targeting Site B, with the join condition in GQ, we can group these two elements as (SiteB.Customers, {name, street, district, phone}, <“Customer.ID = Contacts.CustID” AND “Contype = “Office””>). Such that, 2 sub-queries launched against the two local systems:

Site A:
SELECT firstname, lastname, addr, office
FROM Customers;

Site B:
SELECT name, street, district, phone
FROM Customers, Contacts
WHERE Customer.ID = Contacts.CustID AND
Contype = “Office”;

There are two reversion required during the result integration phase: (1) *addr* from Site A is reversed into *street* and *district*; (2) *name* from Site B reversed to *firstname* and *lastname*. Also, both *office* and *phone* from both sites united in the domain of the global data item *phone*. As a result, the global query result is returned.

CHAPTER 6

Analysis

Multidatabase languages extend the traditional data manipulation languages with capability handling multiple data sources. This approach does not have an integrated schema and it does not mediate changes at the local system level. Users are exposed to all technical details of local systems, i.e. local systems are not transparent to users, and they need to resolve conflicts by themselves. Thus, users must specify all technical details of local systems to formulate a global query. Metadata and knowledge, which can automate conflict resolutions without compromising local autonomy, are not included in this approach. When comparing incompatible data formats, e.g. joining a numeric data item to an alphanumeric one, user-defined conversion functions are required making the query even more complicated, if not incomprehensible. For retrieving information from horizontal partitioned data objects, manually operations from identifying the existence of horizontal partitioned data objects to creating resolutions are required in the multidatabase approach. By combining metadata and knowledge, the Metadatabase approach makes global query formulations more automated and horizontal partitioned data objects are transparent from users. Given that, the proposed methods for resolving the horizontal partitioning problem and its variants inherit the strengths of the Metadatabase approach. Therefore, the Metadatabase approach save users from dealing with complicated technical details while they are unavoidable when multidatabase approach is employed

Federated Database Management Systems (FDBMS). A FDBMS takes a snap-shot of schemata of local systems are mapped using a common data model (CDM) before integration begins. As a result, only those data items, which are explicitly expressed in

local systems, can be modeled in the integrated schema. Thus, the partition concepts that might be implicitly implied by local systems cannot be modeled in systems using the FDBMS approach. Therefore, all local partitions must be accessed to determine if any required data exists in a particular partition and that is an unnecessary burden for the method presented in this study. In addition, knowledge of local systems that is incorporated in a FDBMS is limited when compared to that in the Metadatabase approach and knowledge is not incorporated in the integrated schema. Missing knowledge regarding which and how data object are partitioned, these methodologies can only either impose assumption on modeling data objects at local systems or hard-coded the knowledge into the system. That, in turn, reduces local autonomy or adaptability of the system. A federation dictionary is built to help resolving conflicts among systems in the FDBMS approach and is implemented as the data dictionary/directory (DD/D) in MERMAID and the integration schema (IS) in Multibase. Conflict resolution methods are hard-coded and stored in the DD/D and IS in MERMAID and Multibase respectively. Hence, these methods must be changed accordingly when there is any change in local systems. On the other hand, methods for conflict resolutions are parameterized, which take metadata as parameters and are independent from local systems. Hence, these methods are more reusable and, most importantly, more manageable in the Metadatabase system.

Integrated schema approach. Conflicts at the data object and data item levels are eliminated in systems using the integrated schema approach. As a result, information can be shared among systems. Yet, any change made to a local system will propagate to all other systems, as it has to be reflected in the integrated schema. In other words, it is difficult to make changes in local systems and hence only low adaptability and autonomy can be achieved in this approach. Information in horizontal partitioned data objects can

be retrieved by consolidating data from different data sources. In order to differentiate data from various data sources, additional information must be attached to each of data in local systems. For example, a "location code" must be attached to all sales records in the Sales Center A such that the data source of those records can be identified after they are consolidated at the global level with sales records from other sales centers. Nonetheless, the location codes are implied by the system installed in different sales centers and they are functional at the local system level. Besides, it dose not allow overlapping among partitioned. Therefore, resolution for information retrieval from horizontal partitioned data objects provided by the integrated schema approach would be restrictive and inefficient. On the other hand, the methods presented in this study enjoy the capability of modeling knowledge provided by the Metadatabase approach. We designated each subset of data (a partition) in a set of horizontal partitioned data objects with partition attributes and partition conditions. They are implicitly implied by the set of horizontal partitioned data objects and might not be physically exists in any of local systems. In the Metadatabase approach, the knowledge is modeled by means of metadata without affecting structures of local systems. It certainly yields a better adaptability and local autonomy. In additional, partition conditions can model complex relationships among partitions such as, overlapping, include, disjoint, etc.. As a result, horizontal partitioned data objects are better modeled not by imposing changes to local system or forgoing valuable information but by knowledge of local systems.

CHAPTER 7

Conclusion and Future Works

Horizontal partitioned data are often found in a heterogeneous distributed environment. Without knowing the existence of such data objects, information retrieval using global queries might return with incomplete or incorrect results. The situation is even more complicated when structure differences, as a result of local autonomy, are taken into consideration. However, the problem of information retrieval with horizontal partitioned data objects is not sufficient addressed in previous studies.

In this study, we have identified the generic semantic and schematic knowledge for representing horizontal partitioned data. This knowledge is then modeled in terms of metadata to describe how they are horizontal partitioned over local systems. Previous researches had not effectively model horizontal partition data because semantic knowledge about local systems was not formally incorporated. In addition, knowledge base was not included in previous methodologies such that there was no effective and systematic way to store, retrieve and manage semantic knowledge of local systems at the global level.

The Metadatabase approach is adopted as the implementation foundation of this study because a knowledge base is formally incorporated in the methodology. The approach also has achieved high level of local system autonomy, transparency, interoperability and adaptability. We extended the existing Metadatabase system with the capability of retrieving horizontal partitioned data objects from local system. We first modeled and represented the knowledge of horizontal partitioned data object using metadata which are subsequently incorporated into the Metadatabase. Then, we enhanced the global

processing of the existing Metadatabase with new methods that will utilize newly identified metadata for retrieval.

As a result, the limitation in retrieving information from local system with horizontal partitioned data objects using the Metadatabase system is lifted while the four favorable characteristics (i.e. autonomy, transparency, interoperability and adaptability) are preserved. Horizontal partitioned data objects are transparent to users who can formulate global queries without realizing data objects are horizontal partitioned. The changes in any local system only require updating the respective metadata stored in the Metadatabase. Therefore, the Metadatabase system maintains a high degree of adaptability. Furthermore, there is no restriction imposed to local systems and they are free to evolve according to local needs as long as these changes are reflected by updating the Metadatabase. Information sharing is not hindered even there are differences in software and/or hardware, system designs, etc. Thus, high level of local autonomy and interoperability are sustained.

Further research efforts should focus on two areas: (1) Extend the modeling tools to help system analysts and modelers to model and represent horizontal partitioned data objects in local systems. Therefore, the insertion of partition attributes, partition conditions of each partition and other related metadata can be more automated; and (2) Optimize global query processing with more sophisticated optimization methods, for example, simultaneously takes into account for object size, local computation time and network efficiency.

References

- [AKWS95] S. Agarwal, A. Keller, G. Wiederhold and K. Saraswat, "Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Database," IEEE 1063-6382/95, 1995.
- [Bou91] M. Bouziane, "Metadata modeling and management," PH.D dissertation, Computer Science Dept., Rensselaer Polytechnic Inst., Troy, NY, 1991.
- [BCDE93] O. Bukhres, J. Chen, W. Du and A.K. Elmagarmid, "InterBase: An Execution Environment for Heterogeneous Software Systems," IEEE 0018-9162/93/0800-0057, 1993.
- [BDKV92] P. Buneman, S. Davidson, A. Kosky and M. VanInwegen, "A Basis for Interactive Schema Merging," IEEE 0073-1129-1/92, 1992.
- [BHP94] M. Bright, A. Hurson and S. Pakzad, "Automated Resolution of Semantic Heterogeneity in Multidatabase," ACM Transaction on Database Systems, Vol. 19, No. 2, June 1994.
- [BLN86] C. Batini, M. Lenzerini and S.B. Navathe, "A comparative analysis of methodologies for database schema integration", ACM Computing Surveys 18(4), December 1986.
- [CBTY89] A. Chen, D. Brill, M. Templeton and C. Yu, "Distributed Query Processing in a Multiple Database System," IEEE 0733-8716/89/0400-0390, 1989.
- [CH96] W. Cheung and C. Hsu. "The Model-Assisted Global Query System for Multiple Database in Distributed Enterprise," ACM Transaction on Information Systems, 1996.
- [Chu90] C. Chung, "DATAPLEX: An Access to Heterogeneous Distributed Databases," Communication of the ACM, Vol. 33, No. 1, Jan 1990.
- [CR93] S.M. Chung and C.N. Ravikiran, "A Heterogeneous Distributed Information System," IEEE 0-8186-4212-2/93, IEEE, 1993.
- [DH84] U. Dayal and H. Hwang, "View definition and generalization for database integration in MULTIBASE: A system for heterogeneous distributed database," IEEE Trans. Software Engineering, SE-10, 6, 1984.
- [DR93] D. Zhou and K. Ramamohanarao. "Representation and Translation of Queries in Heterogeneous Databases with Schematic Discrepancies," Interoperable Database Systems (DS-5) (A-25) pp117-189, IFIP 1993.

- [ERS98] Elagarmid A, Rusinkiewicz M. and Sheth A. (editors), 1998. "Management of Heterogeneous and Autonomous Database Systems, Chapter 1", Morgan Kaufmann Publishers, Inc.
- [GSC95] M Garcia-Solaco, F. Saltor and M. Castellanos, "A Structure Based Schema Integration Methodology," IEEE 1063-6382/95, IEEE, 1995.
- [GSC96] M Garcia-Solaco, F. Saltor and M. Castellanos, "Extensional Issues in Schema Integration," Database Reengineering and Interoperability, p261-273, Plenum Press, New York 1996.
- [HBP94] A.R. Hurson, M.W. Bright, and H. Pakzad, "Multidatabase Systems: An Advanced Solution for Global Information Sharing," Los Alamitos, CA, IEEE Computer Society Press, 1994.
- [HBRY91] Cheng Hsu, M'hamed Bouziane, Lauriw Rattner and Lester Yee, 1991. "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach", IEEE Transactions on Software Engineering Vol. 17, No. 6, June 1991.
- [HRY+92] Cheng Hsu, Gilbert Babin, et, al, "What is Rensselaer's Metadatabase System?," 0-8186-2615-1/92, 1992 IEEE.
- [HTTB93] C. Hsu, Y. Tao, M. Bouziane and G. Babin, "Paradigm Translation in Integrating Manufacturing Information Using Meta-model: The TSER Approach," Information System Engineering, September 1993.
- [Hua94] J. Huang, "Multibase: A Heterogeneous Multidatabse Management System," IEEE 0730-3157/94, 1994.
- [Lit93] W. Litwin, "O*SQL: A language for Object Oriented Multidatabases Interoperability," Interoperable Database System (DS-5) (A-25), IFIP, 1993.
- [Lit94] W. Litwin, "Multidatabase System," Englewood Cliff, NJ, Prentice Hall, 1994.
- [LMR90] W. Litwin, L. Mark and N. Roussopoulos, "Interoperability of multiple autonomous databases," ACM Computing Surveys, 22(3), September 1990.
- [LP93] E. Lim and S. Prabhakar, "Entity Identification in Database Integration," IEEE 1063-6382/93, 1993.
- [LSS94] E. Lim, J. Srivastava, S. Shekhar, "Resolving Attribute Incompatibility in Database Integration an evidential Reasoning Approach," IEEE 1063-6382/94, 1994.
- [Mot87] A. Motro, "Superviews: Virtual Integration of Multiple Databases," IEEE Transactions on Software Engineering, vol. SE-13, July 1987.

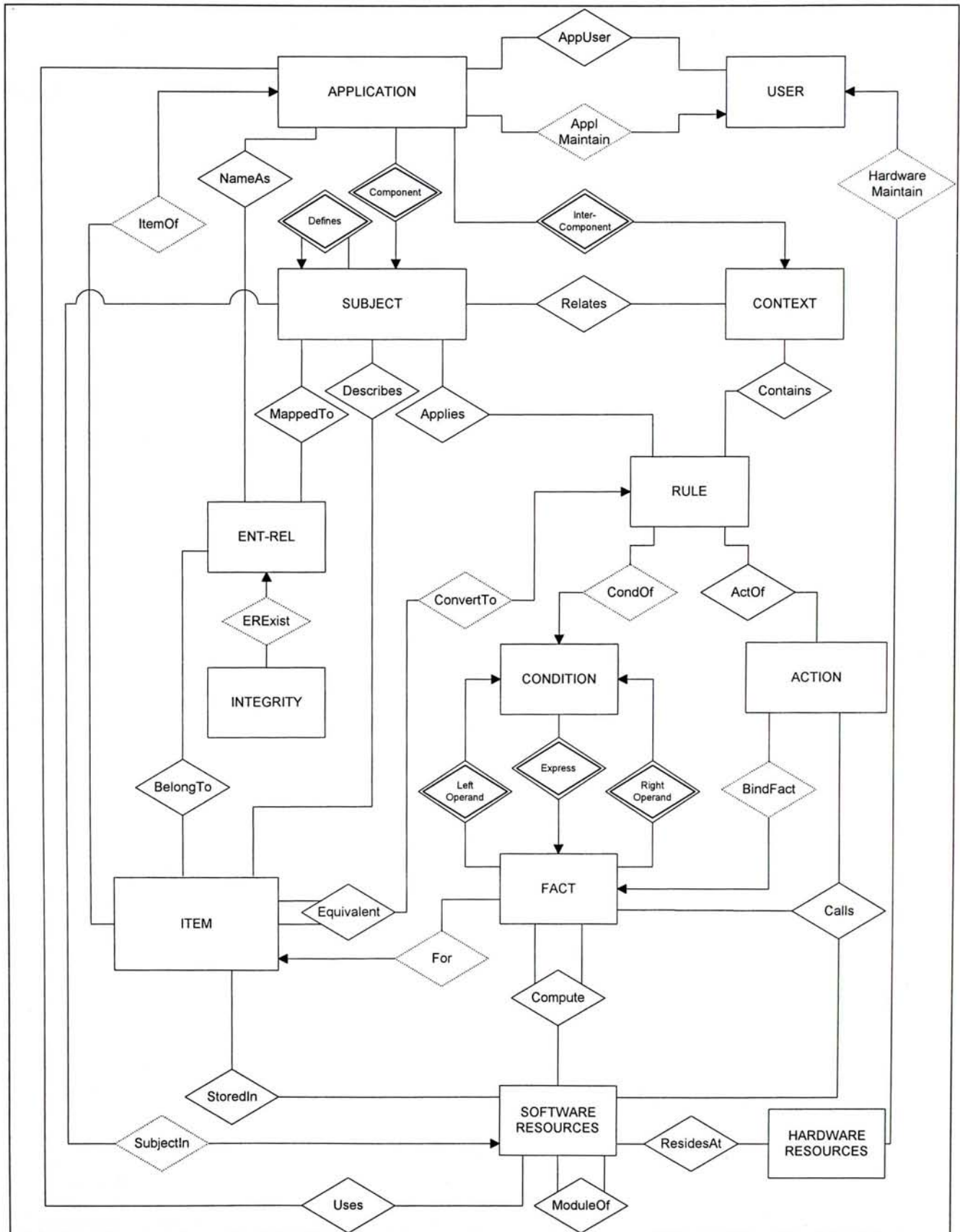
- [PRR91] W. Perrizo, J. Rajkumar and P. Ram, "HYDRO: A Heterogeneous Distributed Database System," ACM 0-89791-425-2/91/0005/0032, 1991.
- [PRSL93] S. Prabhakar, J. Richardson, J. Srivastava and E. Lim, "Instance-level integration in federated autonomous databases," Proceedings of the 26th Annual Hawaii International Conference on System Sciences, Vol. 3, p62-69, 1993.
- [RPR88] M.P. Reddy, B.E. Prasad and P.G. Reddy, "Query Processing in Heterogeneous Distributed Database Management System," ?, 1988
- [RR95] V. Ramesh and A. Ram, "A Methodology for Interschema Relationship Identification in Heterogeneous Database," IEEE 1060-345/95, 1995.
- [SBD+81] J. Smith, P. Bernstein, U. Dayal, N Goodman, T. Lander, K. Lin and E. Wong, "Multibase – integration heterogeneous distributed database systems," Proc. of AFIPS, 1981.
- [SCG93] F. Saltor, M.G. Castekkanos and M. Garcia-Solaco, "Overcoming Schematic Discrepancies in Interoperable Databases," Interoperable Database Systems (DS-5) (A-25) pp191-205, IFIP 1993.
- [Shi81] D.W. Shipman, "The Functional Data Model and the Data Language DAPLEX," ACM Transactions on Database Systems, Vol. 6, No. 1, March 1981.
- [SK93] A. Sheth and V. Kashyap, "So Far (Schematically) yet So Near (Semantically)," Interoperable Database Systems (DS-5) (A-25) pp283-312, IFIP 1993.
- [SL90] A. Sheth and J. Larson, 1990. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", ACM Computing Surveys, Vol. 22, No. 3, September 1990.
- [Sou93] C. Soutou, "Towards a Methodology for Developing a Federated Database System," IEEE 0-8186-4212-2/93, 1993.
- [SP94] S. Spaccapietra and C. Parent, 1994 "View integration: A step forward in solving structure conflicts", IEEE Transactions on Knowledge and Data Engineering 6(2), April 1994.
- [SY96] G. Suzuki and M. Yamamuro, "Schema Integration Methodology Including Structural Conflict Resolution and Checking Conceptual Similarity," Database Reengineering and Interoperability, p247-260, Plenum Press, New York 1996.
- [TBD87] M. Templeton, D. Brill, S. Dao, E. Lund, P. Ward, A. Chen and R. MacGregor, "Mermaid-A Front-end to Distributed Heterogeneous Database," IEEE proceedings, vol. 75, No. 5, May 1987.

- [YOL97] L.L. Yan, M. Tamer Ozsü and L. Liu, "Accessing Heterogeneous Data Through Homogenization and Integration," IEEE 0-8186-7946-8/97, 1997.
- [ZSC95] J. Zhao, A. Segev and A. Chatterjee, "A Universal Relation Approach to Federated Database Management," IEEE 1063-6382/95, 1995.

Appendices

A. GIRD Definitions

A1. GIRD Model⁴



A2. GIRD/SER Contents

⁴ Materials of this appendix are modified based on [CH96] and [Bou91]

KEY: Boxed attributes are new meta-attributes.

Application-view SUBJECT

Attributes:

Applname, descript, Userid, username, class, position, phone, office, address, password, accesscode, addedby, deateadded, modifby, lastmod, nummods

Functional dependencies:

Applname	→	descript, addedby, dateadded, modifby, lastmod, nummods
Userid	→	username, class, position, phone, office, address, addedby, dateadded, modifby, lastmod, nummods
(Applname, Userid)	→	password, accesscode

Synonyms:

Userid = addedby

Functional-view SUBJECT

Attributes:

Applname, Sname, SSname, descript, xcoord, ycoord, Fileid, Cname, Itemcode, itemname, itemtype, format, length, domain, unit, defvalue, Rname, acttype, Factid, factname, facttype, factvalue, valuetype, valueof, Procid, Functid, direction, relpos, relorder, Eqitemcode, convertby, reverseby, corder, rorder, domainmap, Parid, parorder, Resid, resname, addedby, dateadded, modifby, lastmod, nummods

Functional dependencies:

Sname	→	descript, xcoord, ycoord, SSname, Applname, Fileid, addedby, dateadded, modifby, lastmod, nummods
Cname	→	descript, xcoord, ycoord, Applname, addedby, dateadded, modifby, lastmod, nummods
Itemcode	→	itemname, itemtype, descript, format, length, unit, domain, defvalue, Applname, addedby, dateadded, modifby, lastmod, nummods
Rname	→	rtype, descript, Condid, numconds, addedby, dataadded, modifby, lastmod, nummods
Condid	→	leftfact, operator, rightfact, addedby, dateadded, modify, lastmod, nummods
Actid	→	acttype, Factid, addedby, dateadde, modifby, lastmod, nummods
Factid	→	factname, descript, facttype, factvalue, valuetype, valueof

Resid	→	resname
(Cname,Sname)	→	direction
(Itemcode, Sname)	→	relpos
(Sname, Rname)	→	reorder
(Cname, Rname)	→	reorder
(Actid, Rname)	→	reorder
(Itemcode, Eqitemcode)	→	convertby, reverseby, <u>corder</u> , <u>rorder</u> , <u>domainmap</u>
(Factif, Funcid, Parid)	→	parorder
(Actidm Procid, Parid)	→	parorder

Synonyms:

Resid = Fileid = Functid = Procid
 Itemcode = Eqitemcode = valueof (if facttype = 1)
 Factid = leftfact = rightfact = Parid = valueof (if facttype =5)
 Condid = valueof (if facttype = 4 or 5)
 Sname = SSname

Structural-view SUBJECT

Attributes:

Applname, ERname, ertype, descript, akey, partitioned, Inname, inttype, master, slave, Itemcode, itemname, PA, format, length, domain, unit, defvalue, Sname, relpos, inpkey, posinpkey, localname, PC, addedby, dateadded, modifby, lastmod, nummods

Functional dependencies:

ERname	→	ertype, descript, akey, <u>partitioned</u> , addedby, dateadded, modifby, lastmod, nummods
Inname	→	inttype, descript, master, slave, addedby, dateadded, modifby, lastmod, nummods
Sname	→	Applname
(Sname, ERname)	→	addedby, dataadded, modifby, lastmod, nummods
(Itemcode, ERname)	→	relpos, inpkey, posinpkey
(ERname, Applname)	→	localname, <u>PC</u>
(ERname, Applname, Itemcode)	→	<u>PA</u>

Synonyms:

ERname = master = slave

Resource-view SUBJECT

Attributes:

Applname, Fileid, Resid, resname, extension, restype, descript, sizevalue, sizeunit, coding, developedby, Serialno, hname, htype, location, nodename, nodeaddr, purchby, datepurch, manufacturer, Itemcode, itemname, dataorg, Subresid, relationship, path, invokecom, relpos, maintainedby, addedby, dateadded, modifyby, lastmod, nummods

Functional dependencies:

Resid	→	resname, extension, restype, descript, sizevalue, sizeunit, coding, developedby, addedby, dateadded, modifyby, lastmod, nummods
Serialno	→	hname, htype, descript, location, nodename, userid, nodeaddr, manufacturer, purchby, datepurch
Itemcode	→	Itemname
(Applname, Resid)	→	Dataorg
(Subresid, Resid)	→	Relationship
(Resid, Serialno)	→	path, invokecom
(Itemcode, Fileid)	→	Relpos

Synonyms:

Resid = Subresid = Fileid

A3. GIRD/OER Constructs

Meta-entities (meta-OE)

Key: CONSTRUCT NAME (Primary key, attribute[1],...,attribute[n])

- Application** (Applname, descript, Userid, addedby, dateadded, modifyby, lastmod, nummods)
- User** (Userid, username, class, position, phone, office, address, addedby, dateadded, modifyby, lastmod, nummods)
- Subject** (Sname, descript, xcoord, ycoord, Sname, Applname, Fileid, addedby, dateadded, modifyby, lastmod, nummods)
- Context** (Cname, descript, xcoord, ycoord, Applname, addedby, dateadded, modify, lastmod, nummods)
- Ent-rel** (ERname, ertype, descript, akey, partitioned, addedby, dateadded, modifyby, lastmod, nummods)
- Integrity** (Intname, inttype, descript, master, slave, addedby, dateadded, modifyby, lastmod, nummods)
- Item** (Itemcode, itemname, itemtype, descript, format, length, domain, unit, defvalue, Applname, addedby, dateadded, modifyby, lastmod, nummods)

Rule	(<u>Rname</u> , rtype, descript, Condid, addedby, dateadded, modifby, lastmod, nummods)
Condition	(<u>Condid</u> , leftfact, operator, rightfact, addedby, dateadded, modifby, lastmod, nummods)
Action	(<u>Actid</u> , acttype, factid, addedby, dateadded, modifby, lastmod, nummods)
Fact	(<u>Factid</u> , factname, descript, datatype, factvalue, valuetype, valueof)
Software-resource	(<u>Resid</u> , resname, extension, restype, descript, sizevalue, sizeunit, coding, developedby, addedby, dateadded, modifby, lastmod, nummods)
Hardware-resource	(<u>Serialno</u> , hname, htype, descript, Userid, location, nodename, nodeaddr, manufacturer, purchby, datepurch, addedby, dateadded, modifby, lastmod, nummods)

Meat-plural Relationships (PRs)

- 1) **Actof** (Actid, Rname, relorder)
- 2) **Applies** (Sname, Rname, relorder)
- 3) **Appluser** (Applname, Userid, password, accesscode, addedby, dateadded, modifby, lastmod, nummods)
- 4) **Belongto** (Itemcode, ERname, relpos, inpkey, posinpkey)
- 5) **Calls** (Actid, Procid, Parid, parorder)
- 6) **Computes** (Factid, Functid, Parid, parorder)
- 7) **Contains** (Cname, Rname, relorder)
- 8) **Describes** (Itemcode, Sname, relpos)
- 9) **Equivalent** (Itemcode, EqItemcode, convertby, reverseby, corder, rorder, domainmap, addedby, dateadded)
- 10) **Mappedto** (Sname, ERname, addedby, dateadded, modifby, lastmod, nummods)
- 11) **Moduleof** (Subresid, Resid, relationship)
- 12) **NameAs** (ERname, Applname, PC, localname)
- 13) **PA** (ERname, Applname, Itemcode)
- 14) **Relates** (Cname, Sname, direction)
- 15) **Resident** (Resid, Serialno, path, invokecom)
- 16) **Storedin** (Itemcode, Fileid, relpos)
- 17) **Uses** (Applname, Resid, dataorg)

Meta-mandatory Relationships (MRs)

Component (Application, Subject)

Role Application = owner

Role Subject = owned

Inter-components (Application, Context)

Role Application = owner

Role Context = owned
Defines (Subject1, Subject2)
 Role Subject1 = owner (i.e. the superclass)
 Role Subject2 = owned (i.e. the subclass)
Express (Condition, Fact)
 Role Condition = owner
 Role Fact = owned
Loperand (Fact, Condition)
 Role Fact = owner
 Role Condition = owned
Roperand (Fact, Condition)
 Role Fact = owner
 Role Condition = owned

Meta-Functional Relationship (FRs)

Administrator (Application, User)
 Role Application = determinant
 Role User = determined
Bind-Fact (Action, Fact)
 Role Action = determinant
 Role Fact = determined
Condof (Rule, Condition)
 Role Rule = determinant
 Role Condition = determined
Convert (Equivalent, Rule)
 Role Equivalent = determinant
 Role Rule = determined
ERExist (Integrity, Ent-rel)
 Role Integrity = determinant
 Role Ent-rel = determined
For (Fact, Item)
 Role Fact = determinant
 Role Item = determined
Maintain (Hardware-Resource, User)
 Role Hardware-Resource = determinant
 Role User = determined
Itemin (Item, Application)
 Role Item = determinant
 Role Application = determined
Subjectin (Subject, Software-resource)
 Role Item = determinant
 Role Application = determined

A4. Definition of Meta-attributes

Key: attribute-name – non-key field in meta-relations
--

Attribute – key field in meta-relations

Attribute – both key and non-key field in meta-relations

Rows with double outer-box are new items.

Meta-attribute name	Description	Synonym(s)
Accesscode	An attribute of the meta-PR-AppUser that identifies a user's authorized data access level; e.g. Read(R)/Write(W)/Execute(E)/Delete(D).	
Actid	Unique identifier (primary key) for meta-entity- Action	
Acttype	Class of actions of the production rule: 0 – assignment-statement action 1 – declarative-statement action 2 – procedure-call action	
Addedby	Name/initials of a modeler or information administrator who entered the meta-entity or relationship into the GIRD. Provides for an audit trail.	
Address	Home address of a user in meta-entity- User	
Akey	Alternative primary keys for an Ent-rel base relation	
<u>Appname</u>	Unique name (primary key) for an application	
Class	Classification scheme for end-users; can serve to control privileges and data access.	
Cname	Unique name (primary key) for the meta-entity-Context	
Coding	The type of physical representation of a software resource; e.g. Pascal or LISP for program code; and ASCII or VASM for data files.	
<u>Condid</u>	a) Unique name (primary key) for meta-entity-Condition b) an attribute for meta-entity-Rule	
Convertby	Used in meta-PR-Equivalent to represent the rule converting the format of the first item to the format of its equivalent.	Rname
Corder	Used in meta-PR-Equivalent to represent relative position of Eqitemcode when passed into a conversion rule for converting to the first item. For value: >0 – relative sequence 0(zero) – the Eqitemcode is the only parameter in the conversion rule NULL – the domain of the Eqitemcode is part of that of Itemcode	
Dataorg	Indicates how the data is organized in an application in meta-PR-Uses .	
Dateadded	Date the instance of meta-entity or meta-relationship was added to GIRD.	

Meta-attribute name	Description	Synonym(s)
Datepurch	Purchase/acquisition date for hardware resources	
Defvalue	Default value, if any, for a meta-entity-Item	
Descript	Description of all defined meta-entities and meta-relationships	
Developedby	The name of the firm or person who developed a software resource.	
Direction	a) Indicates how the link (data flows) between a Context and Subject is directed graphically: 1 – toward Subject 2 – toward Context 3 – bi-directional nil – none b) An attribute of meta-PR-Relates	
Domain	The set of values that can be assigned to a data item in meta-entity-Item .	
EqItemcode	The equivalent data item in meta-PR-Equivalent	Itemcode
ERname	Unique name (primary key) for meta-entity-Ent-rel .	
Ertype	The type of Ent-rel : OE – operational entity PR – plural relationship	
Extension	The file-name extension (if any) for a software resource.	
<u>Factid</u>	a) Unique system-generated identifier (primary key) for meta-entity-Fact b) An attribute of meta-entity-Action	
Factname	An attribute represents the fact name.	
Facttype	Attribute of meta-entity-Fact that indicates the type of the fact represented and how its value is to be assigned: 0 – constant value 1 – value retrieved from local data item 2 – identifier whose value is supplied by user at run-time 3 – value computed by a function call 4 – value of an expression 5 – identifier bound by an action of a rule 6 – declarative fact	
Factvalue	The calculated or referenced value, or a constant, that binds a fact during the rule inferenceing process.	
<u>Fileid</u>	An attribute of meta-entity-Subject	Resid
Format	The data item representation type in meta-entity-Item	
Funcid	a) Identifies the function to be called for binding a fact b) Key fields in meta-PR-Computes	Resid

Meta-attribute name	Description	Synonym(s)
Hname	Model number or name of a hardware resource.	
Htype	An attribute of meta-entity-Hardware-resource representing the type of hardware; e.g. line-printer, mainframe, etc.	
Inpkey	a) A flag (boolean value) indicating whether or not a data item is part of the primary key of an Ent-rel b) An attribute of meta-PR-Belongto	
Intname	Unique name (primary key) for an integrity constraint	
Inttype	The type of integrity constraint: FR – functional relationship MR – mandatory relationship	
Invokecom	a) The command to invoke a software resource on a hardware resource b) An attribute of meta-PR-Residesat	
Domainmap	An attribute indicates the type of domain mapping between the Itemcode and the EqItemcode meta-PR-Equivalent . 0 – normal mapping 1 – the domain of the EqItemcode is subset of that of the Itemcode 2 – the domain of the EqItemcode is superset of that of the Itemcode 3 – EqItemcode belongs to the domain of the Itemcode	
Itemcode	Unique system-generated identifier (primary key) for a data element in meta-entity-Item	
Itemname	The name of a data item in meta-entity-Item	
Itemtype	An attribute of meta-entity-Item to indicate whether the data item is “persistent” (exists in at least one local DB) or is generated at runtime.	
Lastmod	Date of last modification of GIRD meta-entities/relationship	
Leftfact	The left operand of an expression	Factid
Length	The length of a data item. (May refer to length in character or bytes depending upon implementation)	
Localname	An attribute of meta-PR-NameAs indicates the local name of a global data object in its corresponding local system.	
Location	Physical location for meta-entity-Hardware-resource	
Manufacturer	The manufacturer of a hardware resource	
Master	An attribute of meta-entity Integrity representing the role of an Ent-rel : a) the determinant of an FR b) the owner of a MR	ERname
Modifby	Identifier (name or initials) of an individual who	

Meta-attribute name	Description	Synonym(s)
	last modified an instance of a meta-relation	
Nodeaddr	Network address for a hardware resource	
Nodename	Network "node" name for a hardware resource	
Nummods	Number of modifications to a meta-entity. It exists in all meta-entity and most of meta-PRs.	
Office	Office location or address of meta-entity-User	
Operator	The logical operator in antecedent of a production rule. This includes the set of arithmetic and set operators.	
PA	A partition attribute of a cluster of horizontal partitioned local data object	Itemcode
PC	A set of WWF specifies the domain of a local data object with respect to the PA in a horizontal partitioned cluster. "ALL" is designated for satisfying any PA value automatically.	
Parid	It represents a parameter of a function or procedure.	Factid
Parorder	The relative position of the parameter in a function/procedure parameter list	
Partitioned	A flag (boolean value) indicates if a global data object in meta-entity-Ent-rel that is horizontal partitioned.	
Password	The password to an application in meta-PR-Appuser	
Path	Path to top level directory in which a software resource resides on a hardware resource	
Phone	Business telephone number of a user	
Posinpkey	The relative position of a data item field in the primary key of Ent-rel	
Position	Organizational position of the user; e.g. president, DBA, etc.	
Procid	It identifies the procedure to be called by a rule action.	Resid
Purchby	Identifier of individual responsible for the purchase of the hardware resource	
Relationship	The relationship among software resources; in meta-PR-Moduleof	
Relorder	Relative order (sequence) of a Rule within a Subject or Context – or of an Action in a Rule	
Relpos	Relative position of a data item in meta-entity-Ent-rel	
Resid	A unique identifier (primary key) for meta-entity-Software-resource	
Resname	Title/name of a software resource	
Reverseby	Used in meta-PR-Equivalent to represent the rule converting the format of Eqitemcode to the format of its equivalent (i.e. opposite of convertby).	Rname
Rightfact	The right operand of an expression	Factid

Meta-attribute name	Description	Synonym(s)
Rname	Unique (primary key) for meta-entity-Rule	
Rorder	Used in meta-PR-Equivalent to represent relative position of Eqitemcode when passed into a reversion rule for reversing to the first item. (i.e. opposite of corder). For values: >0 – relative sequence 0(zero) – the Eqitemcode is the only parameter in the conversion rule NULL – the domain of the Eqitemcode is part of that of Itemcode	
Rtype	The type of rule; e.g. Modeling, Production, Conversion, etc.	
Serialno	The unique identifier for meta-entity-Hardware-resource	
Sizeunit	The unit of measure for describing storage of a software resource; e.g. KBytes, blocks, cylinder, pages, etc.	
Sizevalue	Quantity of units of storage for a specified software resource (expressed in sizeunit)	
Slave	An attribute of meta-entity Integrity representing the role of an Ent-rel : a) the determined of an FR b) the owned of a MR	ERname
Sname	Unique name (primary key) of meta-entity-Subject	
Subresid	The key field in meta-PR-Moduleof representing a (sub-)software resource.	Resid
SSname	The upper-level (if any) subject name for meta-entity-Subject	Sname
Unit	An attribute of meta-entity-Item indicating a data item unit of measure (if applicable).	
<u>Userid</u>	Unique identifier (primary key) for meta-entity-User	
Username	Full name of a user in meta-entity-User	
Valueof	An attribute of Fact, which is: a) Itemcode – facttype = 1 (A data item) b) Condid – facttype = 4 or 6 (A result of an expression or declarative fact) c) Factid – facttype = 5 (An assignment to other fact)	
Valuetype	Data type (character, integer, etc.) of the value represented by a fact.	
Xcoord	X-coordinate of the graphical representation of a Subject or Context	
Ycoord	Y-coordinate of the graphical representation of a Subject or Context	

B. Problems Representations in Relation Algebra

B1. Horizontal problem

$$\prod_{k, \alpha_1, \dots, \alpha_n} R = \prod_{k^1, \alpha_1, \dots, \alpha_j} R^1 \cup \prod_{k^2, \alpha_k, \dots, \alpha_l} R^2 \cup \prod_{k^3, \alpha_x, \dots, \alpha_y} R^3 \cup \dots$$

where,

$$\begin{aligned} & \{eqv(k^1), eqv(\alpha_i), \dots, eqv(\alpha_j)\} \cup \{eqv(k^2), eqv(\alpha_k), \dots, eqv(\alpha_l)\} \cup \\ & \{eqv(k^3), eqv(\alpha_x), \dots, eqv(\alpha_y)\} \cup \dots = \{k, \alpha_1, \dots, \alpha_n\} \\ \Rightarrow & \{eqv(k^*), eqv(\alpha_{**}), \dots, eqv(\alpha_{***})\} \subseteq \{k, \alpha_1, \dots, \alpha_n\} \end{aligned}$$

k^* is the primary key or a part of primary key of R^* .

Assuming that,

$$\prod_{\{k, \alpha, \dots, \alpha\} - \{k^*, \beta_{**}, \dots, \beta_{***}\}} R^* = \{0\}$$

B2. Level of abstraction

$$\prod_{k, \alpha_1, \dots, \alpha_n} R = \prod_{k^1, f(\alpha_i, \dots, \alpha_j), g(\alpha_k, \dots, \alpha_l)} R^1 \cup \prod_{k^2, h(\alpha_x, \dots, \alpha_y)} R^2 \cup \dots$$

where,

f, g, h are equivalence functions, so that,

$$\begin{aligned} & \{eqv(k^1), eqv(\alpha_i, \dots, \alpha_j), eqv(\alpha_k, \dots, \alpha_l)\} \cup \\ & \{eqv(k^2), eqv(\alpha_x, \dots, \alpha_y)\} \cup \dots = \{k, \alpha_1, \dots, \alpha_n\} \\ \Rightarrow & \{eqv(k^*), eqv(\alpha_{**}, \dots, \alpha_{***})\} \subseteq \{k, \alpha_1, \dots, \alpha_n\} \end{aligned}$$

k^* is the primary key or a part of primary key of R^* .

Assuming that,

$$\prod_{\{k, \alpha, \dots, \alpha\} - \{k^*, \beta_{**}, \dots, \beta_{***}\}} R^* = \{0\}$$

B3. Schematic Variance

$$\prod_{k,\alpha} (R \triangleright \triangleleft_{k=k'} S) = \prod_{k'',\beta_1} T \cup \prod_{k'',\beta_2} T \cup \prod_{k'',\beta_3} T \cup \dots$$

where,

β_1, \dots, β_n shares the same domain, such that

$$\beta_1 \cup \dots \cup \beta_n = \beta$$

$$\Rightarrow \prod_{k,\alpha} (R \triangleright \triangleleft_{k=k'} S) = \prod_{k'',\beta} T$$

and

$$\prod_{\text{eqv}(k''), \text{eqv}(\beta)} T = \prod_{k,\alpha} (R \triangleright \triangleleft_{k=k'} S)$$

C. Details of local systems

Sales Center A	
Operation Entities:	
Customers	CustID , firstname, lastname, addr, mobile, pager, office
Orders	OrderID , CustID, date, amt
Order Detials	OD_ID , OrderID, ProdID, qty
Products	ProdID , name, unitprice

Sales Center B	
Operation Entities:	
Customers	CustomerID , name, street district
Contacts	ContID , CustomerID, phone, contype
Orders	OID , CustID, date, amt, remark
Order Detials	ODID , OID, ItemID, qty
Products	ItemID , name, unitprice

Head Office	
Operation Entities:	
PO	POID , CustomerID, date, amt
PO Details	POD_ID , CustID, PartID, qty
Parts	PartID , descript, unitprice
Invoice	Inv_ID , POID, date, remark
Materials	Mat_ID , descript, qty
Suppliers	Sup_ID , name, addr
Plural Relationships:	
BOM	Mat_ID , PartID , stdyield
Supply	Sup_ID , Mat_ID , unitcost

Global Model	
Operation Entities:	
Customers	CustomerID , firstname, lastname, street, district, sales_ctr
Contacts	ContID , CustomerID, phone, contype, sales_ctr
PO	POID , CustomerID, date, amt, sales_ctr
PO Details	POD_ID , CustID, ProdID, qty, sales_ctr
Product	ProdID , descript, unitcost, sales_ctr
Invoice	Inv_ID , POID, date, remark
Materials	Mat_ID , descript, qty
Suppliers	Sup_ID , name, addr
Plural Relationships:	
BOM	Mat_ID , PartID , stdyield
Supply	Sup_ID , Mat_ID , unitprice

CUHK Libraries



003803525