



TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

AUTOMATION AND COMPUTER SCIENCE FACULTY

eng. Cristian Gabriel DRAGOMIR-LOGA

PHD THESIS
(abstract)

**RESOURCES DISTRIBUTION IN DATABASE SYSTEMS.
DEVELOPMENT OF IMPLEMENTATION METHODS.**

ADVISOR:

Prof.PHD Eng. Iosif IGNAT

2007

Table of contents

| | |
|--|----|
| 1. Resources distribution problem in DB (DataBase) Systems | 1 |
| 1.1. Resources distribution | 1 |
| 1.2. DB Systems Applications | 3 |
| 1.2.1. Company Applications | 3 |
| 1.2.2. High Volume Data Stream Applications | 3 |
| 1.2.3. Review of DB Systems Applications architectures | 4 |
| 1.3. Distributed management of data | 5 |
| 1.4. Research trends | 9 |
| 1.5. Thesis objective and contributions | 12 |
| 1.6. Thesis structure | 14 |
| 2. Related work | 16 |
| 2.1. Methods of distributed DB schema determination | 16 |
| 2.2. Using views in DB systems | 20 |
| 2.3. Data replication in DB systems | 27 |
| 2.4. Identification problem in DB systems | 32 |
| 3. Methods of distributed DB schema determination | 34 |
| 3.1. The case of in memory distributed DB system | 34 |
| 3.1.1. Methodologies of local schema determination of a distributed DB system | 35 |
| 3.1.2. Case study | 37 |
| 3.1.3. <i>Frag_Aloc</i> Method | 40 |
| 3.2. The case of recursion presence in global DB schema | 44 |
| 3.2.1. Integrity constraints | 44 |
| 3.2.2. <i>Det_Im_Ini</i> Method | 47 |
| 3.3. Conclusions | 58 |
| 4. Distribution achievement method by active views | 61 |
| 4.1. Active views | 61 |
| 4.2. Decomposition of operations on active views in operations on fragments | 62 |
| 4.3. Scenarios | 74 |
| 4.4. Experimental results | 82 |
| 4.4.1. Solution for configuration restrictions | 82 |
| 4.4.2. One level hierarchy | 85 |
| 4.4.3. Multilevel hierarchy | 89 |
| 4.5. Conclusions | 94 |

| | |
|--|------------|
| 5. Distribution achievement methods by replication..... | 96 |
| 5.1. ROWA replication logical schemas..... | 96 |
| 5.2. Experimental results..... | 101 |
| 5.3. Comparison of ROWA replication logical schemas..... | 110 |
| 5.4. Conclusions..... | 116 |
| 6. Contributions to identification in distributed DB systems..... | 118 |
| 6.1. Identification notion..... | 118 |
| 6.1.1. Pure distributed system..... | 119 |
| 6.1.2. Combined system..... | 121 |
| 6.2. According_Range algorithm..... | 123 |
| 6.3. Identification problem in case of synonyms..... | 127 |
| 6.3.1. Ability to handle synonyms..... | 127 |
| 6.3.2. Synonyms Convertor..... | 132 |
| 6.4. Experimental results: Distributed Evaluation Environment in E-learning | 139 |
| 6.5. Conclusions..... | 150 |
| 7. Final conclusions and future work..... | 153 |
| References..... | 163 |
| Appendices to distribution achievement method by active views..... | 173 |
| Appendix 1..... | 173 |
| Appendix 2..... | 178 |
| Appendix 3..... | 183 |
| Appendix 4..... | 184 |
| Appendix 5..... | 190 |
| Appendix 6..... | 191 |
| Appendix 7..... | 191 |
| Appendix 8..... | 192 |
| Appendix 9..... | 193 |
| Appendices with published papers..... | 194 |

Abstract

1. Resources distribution problem in DB (DataBase) Systems

DB Systems appeared at the early '60. Relational model is the logical data model that has been imposed by it self although E. F. Codd published his papers over thirty years ago (1970). The research in DB Systems has begun to treat distribution somewhere on '80.

A *resource* or *system resource* in a computer system is "any physical or virtual component with limited availability".

For this thesis we consider a generic company with hierarchical organization and assume geographical distribution (headquarters, region, area-in-region, locality, area-in-locality ...). Suppose that the physical connection between nodes allows high speed communication, than the system is taxonomically "tightly coupled". Quite so, there may be many applications with heterogeneous DB schema. The nodes may work independently; off and on the connection is made by interfaces. In the later case the system is taxonomically "weakly coupled".

Gartner Group [Sco1995] has introduced five styles for client-server processing, depending on the way the components of an application are divided: distributed presentation, remote presentation, distributed function, remote data management and *distributed data management*.

Related to distributed DB schema design it was observed that the subject of *in memory distributed DB Systems* was not studied enough.

Another direction of interest is the influence of *recursion presence at data model level* on distributed DB schema design.

We consider that is worth of interest for *performance assurance methods* of *distributed DB* applications.

The majority in domain papers discuss replication as a solution to fault tolerance. We consider that *replication* is an attractive subject as a method for distribution achievement in a DB System.

We consider that is worth to investigate the problems related to *object instances identification* and *data model* in *data integration* from diverse sources.

The general purpose of this thesis was:

- ◇ *to develop methods for distribution of resources in DB Systems based on hierarchical organization of a generic company;*
- ◇ *to study several variants the distribution could be achieved in DB Systems;*
- ◇ *to offer solutions to the problems revealed by means of resources distribution could be achieved in DB Systems.*

The term "implementation" in title of this paper refers to development of practical solutions for problems emerging from resources distribution in DB Systems.

2. Related work

From the first chapter there were identified the following directions of investigation:

- ◇ conceive distributed DB schema;
- ◇ views usage in DB (the aim of performance in distributed DB Systems);
- ◇ data replication in DB Systems;
- ◇ identification problem in distributed DB Systems.

Chapter two presented the present stage of research following the path of mentioned directions.

3. Methods of distributed DB schema determination

In the third chapter our discussion for distributed DB schema determination treats two cases: at the network nodes work In Memory Data Systems (IMDS); the global DB schema contains recursion.

In first case our contribution is the **algorithm *Frag_Aloc*** which adapts the mixed fragmentation algorithm MAKE-PARTITION [Nav1995] to fragment and allocate the fragments at the nodes of distributed IMDS.

Notation: $\mathcal{L} = \{L_1, L_2, \dots, L_m\}$ the set of m locations; $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_q\}$ the set of q applications; $\mathcal{T} = \{T_1, T_2, \dots, T_t\}$ the set of t transactions; $\mathcal{F} = \{F_1, F_2, \dots, F_k\}$ the set of k fragments.

Definition 3.1: The Image IM_i at the location L_i is the amount of grid cells at the location L_i and is determinate by (3.1.3.1) formula, where MTL is the Mapping Transactions to Locations and MTG is the Mapping Transactions to Grid.

$$IM_i = MTL \triangleright_{loc=L_i} \triangleleft MTG \quad (3.1.3.1)$$

The total cost of allocation at location L_i is C_{T_i} that depends on: unitary cost of accessing the Image IM_i (memory operation), unitary cost of Log operation (disk and memory), and unitary cost of Checkpoint operation (disk). T_i is the set of update and query transactions at location L_i and TA_i is the set of update transactions at location L_i .

$$C_{T_i} = \frac{\dim(IM_i)}{unitm_i} + \frac{\dim(Log_i)}{unitd_i} + \max\left(\frac{\dim(Log_i)}{unitm_j}\right) + \frac{\dim(Chk_i)}{unitd_i} \quad (3.1.3.2)$$

$$\dim(IM_i) = \sum_{t \in T_i} \left(\sum_{f \in IM_i} (Cardinality_f * Touple_Length_f) \right) * freq_t \quad (3.1.3.3)$$

$$\dim(Log_i) = \sum_{t \in TA_i} \left(\sum_{f \in IM_i} (Cardinality_f * Touple_Length_f) \right) * freq_t \quad (3.1.3.4)$$

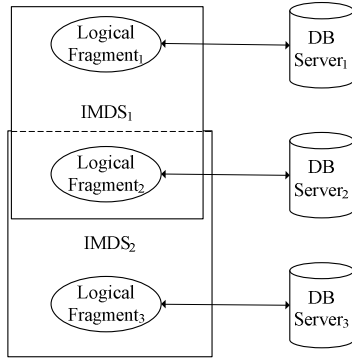


Figure 3.1.2.2 Example, of client-server restoring schema

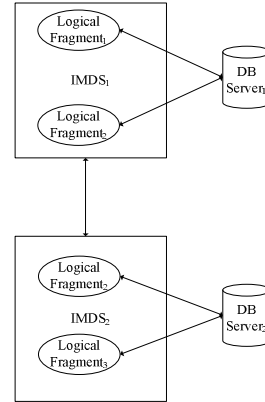


Figure 3.1.2.3 Example of sharing-disk restoring schema

The third term in formula 3.1.3.2 is null for client-server restoring schema.

The IN parameters for *Frag_Aloc* algorithm are: global DB schema, the set of locations (a tree data structure), the Mapping Transactions to Predicates (MTP), the Mapping Transactions to Attributes (MTA), and MTL. Output of algorithm will be the Mapping Fragments to Locations (MFL) and type of restoring schema (tsr).

The difference from Navathe et. al. algorithm MAKE-PARTITION [Nav1995] consists in absence of Grid_Optimization() because it does not fit and it was replaced with the function Migration() that uses tree representation of locations to reduce the cost of allocation.

Complexity of *Frag_Aloc* algorithm is given by allocation part of algorithm which is $O(k^m)$.

Two suppositions were made for **distributed DB schema determination in case of recursion presence in global DB schema**:

Supposition 3.1: global DB schema contains recursive relationships.

Supposition 3.2: data distribution will be achieved by replication methods presented in chapter five.

Guardian condition for fragments consistency we assume is location identifier (simple predicate $p: LOCATION.id = \{list\ of\ values\}$) or by inheritance from current node ancestors.

From location identifier point of view there are two variants of distribution implementation: with respectively without location identifier duplication. Alternative without location identifier duplication uses heuristics like: "if UNIT.id is id_unit_value then location is $id_location_value$ ". Alternative with location identifier duplication has to keep on the restriction derived from functional dependency:

$$Key_X \rightarrow location_key \quad (3.2.1.1)$$

If data structure is of type intersection then $location_key$ has to be taken out from relation. Suppose a key Key_Y , an additional relation ($location_key, Key_Y$) must be defined. For each relation R_{ir} from IM_i ($i = \{1,2,\dots,m\}$ denote location and r denote relation) distinct and common parts have to be determined.

Definition 3.2: Common Part ($\mathcal{P}C_{ir}$) of relation R_{ir} is: $\cup(R_{ir} \cap R_{jr})$ where j domain is from 1 to $d-1$ (d is hierarchical tree degree). We suppose that R_{ir} and R_{jr} have all the same deep.

Definition 3.3: Distinct Part ($\mathcal{P}D_{ir}$) of relation R_{ir} consists of difference $R_{ir} - \mathcal{P}C_{ir}$.

Ideal case is when data structure is of type partition $\mathcal{P}C_{ir} = \emptyset$ for every i_node and every r_entity from global schema. Distinct Part of R_{ir} is candidate for one publication at ancestor level, and Common Part generates more than one publication at ancestor level (top-down). The exponent of a prime factor gives the number of subscriptions. Let FP_{ic} denote a prime factor, component of R_{ir} and n be the total number of prime factors, R_{ir} components.

$$R_{ir} = \bigcup \left(PD_{ir}, \bigcup_{c=1..n} (FP_{ic}) \right) \quad (3.2.2.1)$$

A tree extraction from the forest consecutive for $i = 1..p$ (ancestors of current node $IDNode$ on same path) and for $j = 1..q$ (from current node descending to leafs) is made like below:

$$Tree_Unit_{IDNode} = \bigcup_{\substack{i=1..p \\ j=1..q}} (tmpascu_i, tmpdescu_j) \quad (3.2.2.2)$$

$$loc_Unit_dom = Forest_Unit = \bigcup_{IDNode=1..m} (Tree_Unit_{IDNode}) \quad (3.2.2.3)$$

The Imposed Condition in distribution alternative without location identifier duplication is to define loc_Unit_dom , the domain of unit instances allocated to each location (node).

The paper continued with a case study concerning *timetable generation* [Le12002(2)]. The resources in this problem are: *Curricula, Study Groups, Professors and Classrooms*.

An additional *Imposed Condition* in distribution alternative without location identifier duplication is determinate by fragmentation decision only on actual values. The conclusion is that entities of type "completed entity" have to be entirely replicated.

In distribution alternative with location identifier duplication, resource level generates functional dependencies of type (3.2.1.1) where Key_X is in particular: person identifier, classroom identifier, curricula identifier.

Our contributions are two algorithms that usage heuristics for the starting point, to limit the search space and to eliminate circular references. Det_Im_Ini is an algorithm that determines initial image of distributed by replication DB. $Ajust_Dom()$ is a function used in case of intersection between different branches of the tree \mathcal{E}_{ref} . Next to Det_Im_Ini the algorithm Def_Pub will obtain the publications and subscriptions at each level in hierarchy.

The circular references for entity E_{ref} determine another *imposed condition*: suppose "Link Entities" can connect different prime factors, integrity restrictions of type foreign key between a

"Link Entity" in publication A and entity E_{ref} in publication B , will be avoid by means of "CHECK" restrictions.

The number of subscriptions to publications $domain_Loc_Unit$ is prime factor degree. The number of subscriptions to publications obtained for red entities is one. The number of subscriptions to publications obtained for blue entities is determined by intersection with location image.

4. Distribution achievement method by active views

The way a practical method of distribution could be conceived was analyzed in chapter four.

Assumption: At each node L_i we consider DB has tables which correspond to fragments allocated to that node (conform MFL) and views which remake global schema. In case of horizontal fragmentation view definition consists of union of fragments. In case of vertical fragmentation view definition consists of join of fragments.

Definition 4.1: An Active View $VA=\{V, A\}$ is a view V with the following properties:

- remakes a global relation from two or more fragments;
- has the set A of actions attached to it.

The activate events are updating operations {insert, update, delete}. VA definition consists of standard blocks SELECT-FROM-WHERE.

Chapter four continued with an analysis regarding performance evaluation in distributed DB System, the way operations on active views are decomposed in operations on fragments.

In the graphical representation of queries, leafs of operators trees are global relations and each of intermediate node is unary or binary relational algebra operation. On hierarchical organized company model case study we presented equivalence transformations with respect to six equivalence criteria.

The optimizer is the means performance of DB System is attended. Input is an inquiry demand and optimizer generates an optimal execution plan from an alternative execution set.

Definition 4.2: A *Scenario* noted \mathcal{S} is a function with two sets of parameters: input parameters I and output parameters \mathcal{E} .

A testing problem is a sum $\sum_{i=1}^n (\mathcal{S}_i)$ of relevant \mathcal{S}_i scenarios of object operation. The number n of scenarios must be determined that testing problem be complete. The structure of input parameters is composed from hardware architectures and software parameters. \mathcal{E}_i parameter represents the result of object operation for \mathcal{S}_i applied on I_i .

The scenarios in our experiments used the following input parameters I :

1. Hardware Parameters:

We tried to use a minimal configuration formed of three computers (alpha, beta, gamma) connected in LAN.

2. Software Parameters:

- Software parameters of DBMS (MSSQL Server 2000). Software parameters of Operation System (Windows) were fixed in all experiments.
- Logical Operations.

Output parameters \mathcal{E} were execution plans and performance quantifications.

In the first experiment we worked with a one-level hierarchy, a simpler and smaller DB. The second experiment used a multi-level hierarchy suchlike DB schema from "TPC-H benchmark" and near real dimension. Both experiments had a series of scenarios where we gave attention to the effect of indexes and statistics on execution plans.

Active Views have been implemented with "partitioned views" and triggers. DB fragments have been implemented with "CHECK" constraints.

The trigger ins_act_view illustrates the action for INSERT on generic active view act_view :

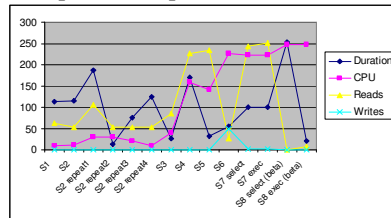
References between a local fragment component of *act_view* and a completed table *tab_compl* were solved with *upd_act_view_{tab_compl}* action of generic active view *act_view*.

The conclusions of chapter four are summarized below:

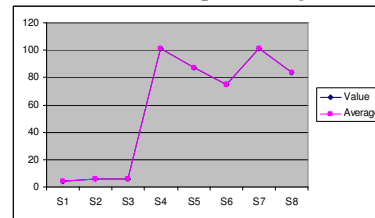
- We showed that distribution could be achieved by active views and VA gives the solution to restrictions generated by hardware and software configuration.
- Statistics helped DBMS optimizer in re-grouping some elementary operations (first selection then join and last union). The second experiment on near real DB confirmed it.
- A limitation of configuration showed that VA without indexes generates a lower estimation from optimizer (ERC/RC parameter differed from 1).
- Data volume in first experiment indicated that statistics and indexes did not give a suggestive increase of performance (performance is better on high data volume like in second experiment). The performance varied with computer loading at query moment.
- It was revealed a VA limitation in the second experiment, scenarios S4 .. S8. The sub-schema had to be reorganized in sense that VA definition was restricted to base tables. Experimental configuration did not allow imbricate view definition.

Next figures give the **results of scenarios S1 .. S8**:

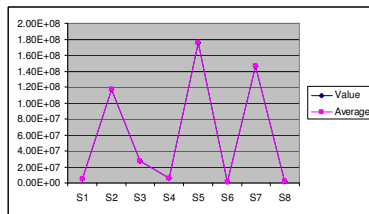
Comparative experimental results:



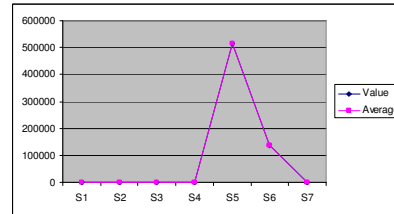
"Cumulative client processing time":



"Cumulative wait time on server replies":



"Rows affected by SELECT":



5. Distribution achievement methods by replication

In chapter five we presented replication topologies, personal contributions to ROWA replication method.

A DB could be seen like a collection of data elements \mathcal{X} . A *replicated* DB is a set of DBs with the property that each DB keeps a copy X_i of the element X at different location. An operation $O_i(X)$ of a transaction T_i could be a read or write access to logical element X , in DB. Logical operation $O_i(X)$ must be translated in physical operations on all X_i copies.

Global DB schema considered for discussion contained principal entity types: Complete Entity (A and D), Incomplete Entity (C) and Relationship Entity (D). We supposed the following primary keys: A.a, B.b and C.c (D inherits primary key components from B and C). Let m be the number of locations organized in a hierarchical structure that has d degree and h depth (in analyzed example $h=2, d=3$).

We conducted our discussion on **two cases**:

1. Each table contains location key and fragmentation is made upon value of location key (partition).
2. Completed tables A and B are not fragmented. C and D tables are fragmented upon C.c *between* v_i and v_j . In this case it is possible that two leafs have same values in table A or in table B but not in table C or in table D (intersection).

Logical data elements \mathcal{X} (resources), in all proposed replication variants are determinate by hierarchical locations structure. A logical data element X is compound from one or many tables (together with other DB objects). The discussion analyzed in each case the way X and X_i are assigned to nodes. Logical data element X is associated to the concept of *publication* from the replication model "publisher-subscriber", and each X_i copy of X is associated to the concept of *subscription*. Instance of X_i at time 0 is initial snapshot.

In **first ROWA replication variant** the starting point for publication definition is the root. We define publications successively from the root to depth on top of leafs. The number of publications depends on degree d of the tree.

In first case of analysis, each publication at node L_{ij} will be composed by four articles (number of tables) and there exists a filter on value of location key. The specifications regarding initial snapshot are implicit (recreation of table). At the depth i , each j node $L_{i,j}$, has defined $d_{i,j}$ publications for level below ($d_{i,j}$ is degree of $L_{i,j}$ node and $i = 1 \dots h-1$; $j = 1 \dots d$) and has defined one subscription to parent suiting publication. Total number of publications at depth i is at most d^{i+1} and total number of subscriptions is d^i . Maximum number of subscriptions per tree is $(m-1)$ and maximum number of publications per tree in terms of d, h is:

$$\frac{d^{h+1} - 1}{d - 1} - 1 \quad (5.1.1)$$

In second case there are two generic publications. First of the two includes tables A and B without filter. The second generic publication includes tables C and D and contains a filter on C.c *between* v_i and v_j . Both generic publications have the number of articles equal with the number of tables (two). Maximum numbers of publications and subscriptions versus first case are double.

The **second ROWA replication variant** distinguishes from first ROWA replication variant by supposition that root node is the only one that has defined publications: $(m-1)$ in first case of analysis and m in second case of analysis. We must remark for the second case of analysis that the number of publications has reduced because the publication that contains table A and table B is reused. At intermediate levels and leaf levels there are defined only subscriptions: $(m-1)$ respectively $2(m-1)$.

For the **third ROWA replication variant** we considered: 1 - the fact that updates are merely made at leafs level and 2 - publication has priority vis-à-vis subscription. In first case of analysis, at the basis of tree are defined d^h publications (at most), each of it with a subscription which corresponds to node parent. At intermediate level, each node keeps one publication to a DB compounded from d components (subscriptions to his children. Total number per tree is $(m-1)$ publications and $(m-1)$ subscriptions. In second case of analysis, publications for tables C and D pursue the same rule. Existence of more than one publication with A and B articles, at last level, is excluded. As a consequence the combination between third replication variant and second analysis case is invalid.

Another possible method, named **fourth ROWA replication variant** is inverted schema. In first case of analysis each leaf contains one publication and $h-1$ subscriptions (path length to the root). Maximum total numbers per tree are d^h publications and $d^h * (h-1)$ subscriptions.

In the second case of analysis, for generic publication that includes articles with tables C and D, at most d^h publications and $d^h * (h-1)$ subscriptions will be defined. The combination fourth ROWA replication variant and second case of analysis however is not valid for the same reason like presented at third ROWA replication variant.

In favor of second case of analysis, where tables A and B on different branches could have common tuples, third and fourth replication variants will be replaced with combination top-down for A and B respectively bottom-up for C and D. A and B will have one publication at root with $(m-1)$ subscriptions, and for C, D will be defined d^h publications at leafs with $d^h * (h-1)$ subscriptions. Combined replication variant is named **fifth ROWA replication variant**.

Configuration for practical validation of proposed model consisted of three computers. One of them functioned both online and offline (unattached to the network). The methods 'eager'

respectively 'lazy' were tested with Microsoft SQL Server 2000. In thesis we presented the steps required for defining publications and subscriptions. The metrics used to validate proposed replication methods were as follows: duration of publication initialization; duration of synchronization; DB dimension; correctness; conflicts; scalability.

Conclusions of chapter five are summarized below:

- ◇ The first two variants: "top-down" with the number of publications at each level of hierarchical structure equal with tree degree; and "star" with publications only at root in number of m-1 are natural in the sense of data partitioning.
- ◇ Variants "bottom-up" and "inversed" impose some restrictions on how publications/subscriptions could be defined in first case of analysis (partition of data): (a) initial snapshot application has to delete existing filtered data and (b) referential integrity must not be propagated.
- ◇ Variants "top-down" are correct. Variants "bottom-up" and "inversed" in second case of analysis (intersection of data) are invalid. As a solution is variant five, the combination of "top-down" variant for *Completed Entities* in global schema with "bottom-up" variant for the other entities in global schema.
- ◇ Analysis of initialization of publications duration observed in first two variants duration is the same and it is lower than duration in inverted variant. The minimal duration is attended by "bottom-up" method.
- ◇ Synchronization duration is in proportion with length from leaf to root (leaf-root-intermediate_node). Priority level can generate extra communication when data intersection generates conflicts.
- ◇ Replicated DB dimension ratio growth reported to initial DB dimension was 25 percent per publication.
- ◇ Primary key configuration table for entities of type *Completed Entities* must be replicated in manner "top-down".
- ◇ Conflicts could be detected earlier if synchronization is made in fixed order. In variant five publication corresponding to *Completed Entities* is first to be synchronized.
- ◇ System is scalable but with some complexity at reorganization. Variant "top-down" is more expensive because of re-initialization of subscriptions, or necessity to delete and generate again the top publication together with connected publications and subscriptions.

6. Contributions to identification in distributed DB systems

In chapter six there were discussed problems related to identification in distributed DBs. In the first section of chapter six we analyzed identification notion structured on data distribution characteristic: pure distributed system (achievement of distribution by active views without replication), combined system (achievement of distribution by active views and replication, figure 6.1.2.1).

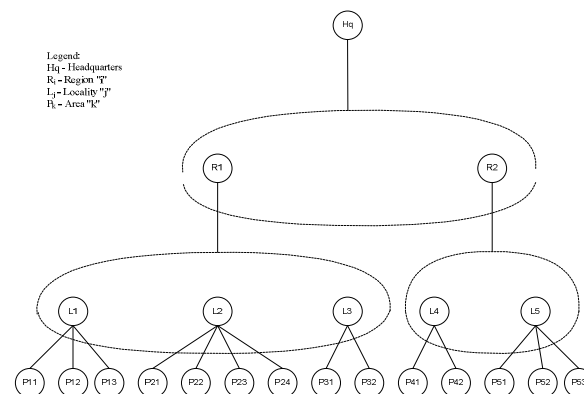


Figure 6.1.2.1 Combined distributed system: active views and replication

Dotted line in figure 6.1.2.1 denotes active view aggregation at a different level. Each constituent fragment of an active view corresponds to a data fragment entirely replicated from below level.

Chapter six continued with presentation of own contribution **algorithm According_Range** useful to correlate an internal identification values set IR and an external set ER of identifiers [Dra2003(1)].

Definition 6.1: External set ER of identifiers is a set of values $\{er_1, er_2, \dots, er_n\}$ with the property that each element of the set uniquely identifies a named object of class C of n real world objects (external objects).

Definition 6.2: Internal set IR of identification values is composed of key instances $\{ir_1, ir_2, \dots, ir_n\}$ of a DB representation of a class C of n real world objects (internal objects).

Definition 6.3: We nominate *correspondence* from set ER to set IR , ER and IR not null sets, a binary relation $C \subset ER \times IR$ that obeys the condition $card(C) = card(ER) = card(IR) = n$ (correspondence has the same cardinality like external objects set and internal objects set).

Let: nop be the number of objects, ip input position, op output position. It was assumed that external object set is divided in subsets and there is an order relation of subsets: *vertical*, respectively *horizontal*. Let nop be the cardinality of a subset. The elements of IR could be sorted, optional, on *criteria* at the moment correspondence is generated. Vertical order describes the following way in which binary relation C is built up: first external object from first subset match the first sorted internal object; next pair objects consists of the first external object from second subset and the second internal object; ... ; at step $(nop+1)$ the second external object from first subset match $(nop+1)^{th}$ internal object; etc. Horizontal order gives a natural relation between ER and IR .

Utility of *According_Range* algorithm was proved with the problem of documents printing in a distributed DB system with configuration like figure 6.1.2.1. The algorithm could be implemented as stored procedure. The weakness of *According_Range* algorithm consists in the fact that DB status is considered entirely constant in sense of sorting criteria. That is why it works better with small spans, in optimistic manner. Large spans imply blocking at higher granularity what is expensive in terms of shared access (both algorithm phases supposed *serialization* at transaction level). Identifiers values are added at m locations and by synchronization are viewed at a special site "*documents printing site*" (dps).

Next debated subject was **ability to treat synonyms**. Idea comes away from necessity to limit possible values of primary key constituents and to restrict access only to values from *acceptable domain*.

Let AC be the "*acceptable domain*" for a key constituent attribute and ac be a value in domain. For each ac there are zero or many values $syac$ from $SYAC$ (synonyms acceptable) domain. AC and $SYAC$ are treated like completed entities, accordingly will not be fragmented.

Any SQL specification $attr = ac$ must be translated in:
 $attr \text{ IN } lsyac$ (6.3.1)

$lsyac$ is: $\{ac\} \cup \{select\ syac\ from\ ASYAC\ where\ original\ value\ is\ ac\}$ (6.3.2)

$ASYAC$ is a dual ($oac, syac$) that transposes above relation. At external DB level there will be a view with n values for each internal value, where n minimum is 1.

Data integration is attained by means of resources sharing or by means of data transmission (loosely coupled systems). **Description of the identification problem in presence of synonyms** was based on an example with two generic companies called "Businesses" (B_1 and B_2). Each of the company has own applications and DB. There is no supposition about DB schema. The two companies share common customers. In example one company is a bank (B_2) and the other is a services company (B_1). A customer of B_1 issues a demand to B_2 where has an account so that B_2 to debit funds from his account, conditioned by announcement about date and amount (DD).

B_1 and B_2 could have a hierarchical organization structure and DBs could be distributed at many locations. The rule whereupon keys values are assigned in DB is subjectively. Suppose a person request for a service (B_1). The person must be recorded in table "Customer" where object

will receive an "idcust" value. The problem is in discerning if customer is really a new one or an existing one.

Stages in figure 6.3.1.1 treat with collection of customers. Communication between B_1 and B_2 is made on a protocol basis. It is unusual that DB systems of B_1 and B_2 be directly connected like in a homogeneous system (certainly the reason is data security). Updating processes "Update DB (B_1)" and "Update DB (B_2)" have as input a common agreed data structure and the outputs are updated DBs.

Interchanged data (DI) set consists of: source schema S , target schema T , a set \sum_{st} of source-to-target dependencies, and a set \sum_t of target dependencies [Fag2003]. Given a finite source instance, I , the data interchanged problem is to find a finite target instance, J , thereby $\langle I, J \rangle$ satisfy \sum_{st} and J satisfies \sum_t . Source-to-target dependencies set has the form:

$$\forall x(\phi_S(x) \rightarrow \exists y\psi_T(x,y)) \quad (6.3.3)$$

Where: $\phi_S(x)$ is a conjunction of atomic formulas on S ; $\psi_T(x,y)$ is a conjunction of atomic formulas on T ; free variable x is a vector of variables: x_1, x_2, \dots, x_k ; k is set of relations in S ; free variable y is a vector of variables: y_1, y_2, \dots, y_l ; l is set of relations in T .

Let x_1, x_2 be variables in x . A target dependency in \sum_t is *tg*d (6.3.4), or *eg*d (6.3.5).

$$\forall x(\phi_T(x) \rightarrow \exists y\psi_T(x,y)) \quad (6.3.4)$$

$$\forall x(\phi_T(x) \rightarrow (x_1 = x_2)) \quad (6.3.5)$$

Let O be a real world object, R_1 and R_2 be DB representations of object O in DB_1 (with schema S) and DB_2 (with schema T). **Identification problem** comes up in context of correspondence C when an instance of object O in DI has to match an instance of object O in DS (Data Source). That means to search for each O starting from R_1 the set \sum_{st} thereby result R_2 be a representation of object O .

In general case it is not the rule that object O has the same representation in two or more DBs. There comes the intervention of synonyms and *acceptable domain*. We consider for DB_1 and DB_2 an *acceptable domain* is defined for each key constituent attribute (primary and candidate).

AC is updated in manner of facts. If matching could not be establish than a rule based system will decide the instance added to $ASYAC$. A special case is when two maybe different objects from DB_1 and DB_2 have to be put into relation. Suppose $ASYAC$ is included in DB_1 and DB_2 . Next situations will be distinguished:

1. The two DB objects represent the same real world object. *oac* could be dissimilar (*oac*₁ and *oac*₂). Each of *oac* will consider the other a synonym.
2. Between the two DB objects exists "one-to-many" relationship (an example is a person which account must fund own invoices and invoices of his family members). This relationship must be part of DB.

DI will include $LSYAC$ for each key constituent attribute. Constituent attribute values (k number of attributes) for composed key candidates will form a subset of Cartesian product:

$$LSYAC_1 \times LSYAC_2 \times \dots \times LSYAC_k \quad (6.3.6)$$

We consider primary key $Pkey$ will be internal assigned.

$$Pkey \rightarrow \rightarrow Syac_i, i=1..k \quad (6.3.7)$$

The multi-valued dependency (6.3.7) is translated in PJ/NF in k relations $CKEY_i(Pkey, Syac_i)$.

In last years XML became the new standard for data interchanged on Internet. At the receiving moment of a XML document, a process is initiated which exploit hierarchical structure of DOM. The process will generate a derived document where a node "element type" (E) is tied to a DB table and a node "attribute type" (A) is tied to a column. A node "PCDATA type" ({S}) has the role in specifying useful information. In case of distributed DB the process will select data fragment that operation is addressed for application. Depends on fragmentation predicates if exists any rule that could simplify XML document, to reduce its dimension (the document representing global schema instance is 'split' in many *document-fragment*).

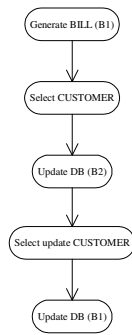


Figure 6.3.1.1 Activity diagram for discussed example

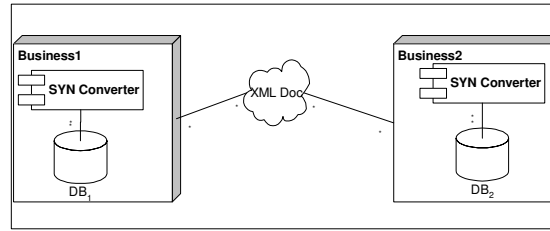


Figure 6.3.2.2 System Architecture

SYN Converter is a component which basis role is translation of (6.3.1). The synonyms for a key candidate constituent will be present in XML document like multi-value attribute or as distinct elements with concordant attributes. SYN Converter beyond other duties will reveal *pkey* from the join:

$$\bowtie (CKEY_1, CKEY_2, \dots, CKEY_k) \quad (6.3.9)$$

And on a proper direction will construct a "well-formed" XML document.

An instance "customer id candidates" is showed below in both cases:

1. multi-value attribute:

```
<Surname Sur1 Sur2 />
<Forename For1 For2 For3/>
```

2. distinct element for identifier candidate attribute:

```
<Surname>
  <Value1 Sur1/>
  <Value2 Sur2/>
</Surname>
<Forename>
  <Value1 For1/>
  <Value2 For2/>
  <Value3 For3/>
</Forename>
```

SYN Converter will assign distinct names to each attribute resulted from LSYAC values. In first above alternative the whole LSYAC set become a multi-value attribute. The algorithm that executes SYN Converter tasks when global view has to be materialized begins for each instance of a master relation with extraction of Sy_{ac_i} set. At each location there exists a local schema Σ_L that maps relational schema to materialized global view (XML document). Auxiliary information caught by special XML nodes of type PCDATA consists in assertions like functional dependency Address \rightarrow Customer and is useful to solve orphan instances in temporary universal table.

For example in figure 6.3.1.1 an optimization consists in regenerating XML document representing materialized global view at each new cycle and until than primary keys will accompany.

Practical testing of Synonyms Converter was presented with e-learning application, DTSEE, for student knowledge evaluation.

7. Final conclusions and future work

In chapter seven we summarized contributions in each chapter of the thesis and presented future work.

We want to underline that distributed DB systems are really helpful as it will be in the future.

Essential contributions of this thesis are as follows:

1. Developed a mixed method of global DB schema fragmentation and allocation of fragments to in memory DB system nodes.

2. Presented the imposed conditions to distributed DB schema that contains recursion relationships. Developed a method that determines initial status of distributed DB system in context of data recursion and distribution achieved by replication.
3. Proposed active views for distribution achievement at external DB level. Tested active views on two data sets corresponding to hierarchical organization model.
4. Presented five methods for distribution achievement by replication of type ROWA. Tested proposed methods on same data sets like in active views case.
5. Critical analyzed of de facto status in identification problem in commercial DB systems.
6. Conceived a method that creates a matching between two identifiers data sets in a distributed DB system.
7. Proposed the concept *synonyms convertor* and presented two applications of this concept.

Possible future research would be in context of applications conceived as a hardware progress fact. We imagined the following possibilities: adaptation of our fragmentation algorithms to P2P systems, with and without replication; analyzing the possibility of use ROWA replication methods to mobile DB systems; the study of synonyms convertor behavior in HVDS ("High Volume Data Stream"); data security and confidentiality; data integration of heterogeneous data sources in context of schema can change.

Selected References

- (1) Bar1985 - Barbara D. and Garcia-Molina H., "Mutual exclusion in partitioned distributed systems", Dept. Computer Science, Princeton Univ., Princeton. NY, 1985, Tech. Rep. TR-346
- (2) Bel2000 – Bellatreche L., Karlapalem K., Simonet A., "Algorithms and Support for Horizontal Class Partitioning in Object-Oriented Databases", Distributed and Parallel Databases, Vol. 8, Issue 2, 2000, pages 155 - 179
- (3) Bun2001 - Buneman P., Davidson S., Fan W., Hara C. and Tan W., "Reasoning about keys for XML," DBLP, 2001
- (4) Cal2000 – Calvanese D., De Giacomo G., Lenzerini M., Vardi M. Y., "Answering Regular Path Queries Using Views", In Proceedings of 16th International Conference on Data Engineering, 2000, pages 389 - 398
- (5) Cer1984 - Ceri S. and Pelagatti G., "Distributed Databases: Principles and Systems", New York: McGraw-Hill, 1984
- (6) Dat1992 - Date C. J. and Fagin R., "Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases," ACM Transactions on Database Systems 17(3), 1992, pages 465-476
- (7) Deu2003 - Deutsch A. and Tannen V., "Reformulation of XML Queries and Constraints," International Conference on Database Theory (ICDT), 2003
- (8) Dra2003(1) - Dragomir G., "Identification in a Distributed Database System," in Proceedings of the 2003 SINTES11 Vol. 2, pages 323-327.
- (9) Dra2004 - Dragomir G. and Ignat I., "Distributed Database System in Interaction With XML, an Identification Problem," in Proceedings of the 2004 INES, pages 296-300
- (10) Dra2006(1) - Dragomir G., Pop A. and Sidea E., "Distributed Tool for Student Evaluation in E-learning ," 2006 IEEE International Conference AQTR, tome I, pages 364-369
- (11) Dra2006(2) - Dragomir G. and Ignat I., "Distributed Database Environment Testing," 2006 IEEE International Conference AQTR, tome II, pages 90-95
- (12) Dre1993 - Drenick P. E. and Smith E.J., "Stochastic Query Optimization in Distributed Databases", ACM Transactions on Database Systems, Vol. 18, No. 2, June 1993

- (13) Eze1998 - Ezeife, C., Barker, K. "Distributed Object Based Design: Vertical Fragmentation of Classes", *International Journal of Distributed and Parallel Databases*, Vol. 6, No. 4, 1998, pages 317-350
- (14) Fag2003 - Fagin R. and Kolaitis P. G., "Data Exchange: Getting to the Core", *ACM Symposium on Principles of Database Systems*, 2003, pages 90-101
- (15) Gan2004 - Ganesan P., Bawa M., Garcia-Molina H., "Online Balancing of Range-Partitioned Data with applications to Peer-to-Peer Systems", in *Proceedings of the 30th VLDB Conference*, Toronto, Canada, 2004, pages 444-455
- (16) Hay2005 - Hayashi H., Hara T., Nishio S., "A Replica Allocation Method Adapting to Topology Changes in Ad Hoc Networks", *DEXA*, 2005, pages 868 – 878
- (17) Hol2003 – Holliday J., Steinke R., Agrawal D., Abbadi A. E., "Epidemic Algorithms for Replicated Databases", *IEEE trans. Knowl. Data Eng.*, Vol 15, No 4, 2003, pages 1218 - 1238
- (18) Ioa1996 - Ioannidis Y., "Query Optimization", *ACM Computing Surveys*, symposium issue on the 50th Anniversary of ACM., Vol. 28, No. 1, 1996, pages 121-123.
- (19) Jag1994 – Jagadish H. V., Lieuwen D., Rastogi R. and Silberschatz A., "Dali: A High Performance Main Memory Storage Manager," in *Proceedings of the 20th VLDB Conference* Santiago, Chile, 1994
- (20) Lel2002(2) - Lelutiu A., Horea M. and Olah P., "About the Computer-Aided Time-Table Generation in an Education Information System", *SSGRR-2002*
- (21) Len2002 - Lenzerini M., "Data integration: A Theoretical Perspective," in *PODS*, 2002, pages 233-246
- (22) Lim1993 - Lim E. P., Srivastava J., Prabhakar S. and Richardson J., "Entity Identification in Database Integration", in *Proceedings Ninth International Conference on Data Engineering*, Vienna, Austria, April 19-23 1993, *IEEE Computer Society Press*, Washington DC, 1993, pages 294-301
- (23) McB2003 - McBrien P., Poulouvasilis A., "Data Integration by Bi-Directional Schema Transformation Rules", In *Proc. ICDE'03*, Bangalore, March 2003, pages 227-238
- (24) Nav1995 - Navathe S. B., Karlapalem K. and Ra M., "A Mixed Fragmentation Methodology For Initial Distributed Database Design", *Journal of Computer and Software Engineering*, Vol. 3, No. 4, 1995, pages 395-426
- (25) Ozs1991 - Ozsü M. T. and Valduriez P., "Principles of Distributed Database Systems", *Prentice Hall*, 1991
- (26) Wid1999 - Widom J., "Data Management for XML: Research Directions," *IEEE Data Engineering Bulletin*, Special Issue on XML, 22(3), Sep. 1999, pages 44 - 52.
- (27) Wie2005 – Wiesmann M., Schiper A., "Comparison of Database Replication Techniques Based on Total Order Broadcast", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 4, 2005, pages 551-566