EXPERIMENTAL EVIDENCE ON DATA WAREHOUSE FRAGMENTATION AND ALLOCATION IN A DISTRIBUTED CONTEXT

Tekaya Karima, Abdellaziz Abdelatif
Data-processing Department, Faculty of Sciences of Tunis, Rommana, El Manar 2, Tunisia

Habib Ounalli
Data-processing Department, Faculty of sciences of Tunis, Rommana, El Manar 2, Tunisia

Keywords: Distributed data warehouse, Fragment Evaluator, Fragmentation, Allocation, Primary and derived horizontal fragmentation.

Abstract: Since a relational data warehouse has the same physical structure as a classical database, it can enjoy all the benefits realized during the past in distributed databases such us data availability, simplicity, rapid local data access and transparent access to remote sites. So, it would be interesting to test its decentralization by adapting the most adequate fragmentation and allocation technique. In this paper, we present a data warehouse fragmentation and allocation approach in a distributed context. We conduct first, computation studies using a mathematical cost model. Then, we test our approach on a real data warehouse by using the APB1 benchmark data set on Oracle11G.

1 INTRODUCTION

Data Warehouse (DW) decentralisation starts by data fragmentation and then fragment allocation. Obviously, fragmentation is considered today as an important option in physical DW design. Its process begins with the collection of the predefined queries throughout the geographically dispersed company sites. Then, it extracts from these queries all the join, projection and selection operations. These operations are used for logical tables partitioning. The deployment of the fragmentation in a DW must be quit founded with the specificities of the multidimensional modelling. In fact, a relational DW is modelled by a star schema which consists of a fact table and several dimension tables. The former is a huge table constituted by two attribute types: key attribute (conjunction of foreign keys) and measure attributes. The fact table is interconnected to dimension table's by links of cardinalities one to many. The latter, around the fact table, are generally small-sized, denormalized and contain two types of attributes: descriptive and hierarchical ones. Despite the fact that a relational DW has the same physical structure as a relational database, it is distinguished by its multidimensional aspects. In fact, each measure in the fact table is determined according to a number of dimension keys. Fragmentation must respect the granularity level of the fact table. In addition, analyses are done according to one or many dimension(s). Each dimension has some hierarchical attributes that support ascending (or descending) analyses. Such a topology (star model) is conceived to answer effectively the requirements of the analysis queries. A well adapted fragmentation technique, therefore guarantees a best deployment of the fact table granularity level and dimension hierarchical attributes. Otherwise, in a distributed context, the fragmentation technique aims to decentralize the DW. Fragmentation is based on decision makers requests, their geographical location and the number of partitions needed to efficiently meet their needs. Fragmentation is no longer limited to optimize data storage, but it extends to the reorganization of the whole logical model of the DW into several local models according to local needs of the company sites. As in to the centralized environment, local and remote

Karima T., Abdelatif A. and Ounalli H..
EXPERIMENTAL EVIDENCE ON DATA WAREHOUSE FRAGMENTATION AND ALLOCATION IN A DISTRIBUTED CONTEXT.
DOI: 10.5220/0003104801020110
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2010), pages 102-110
ISBN: 978-989-8425-30-0

access to data in the decision support system must be transparent to users, i.e., managed by the current distributed data base management system. Consequently, the DW can benefits from all the advantages of distributed databases such as data availability, simplicity, rapid local data access and transparent access to remote sites.

The paper is organized as follows: in section 2, we present related works. In section 3, we illustrate a DW fragmentation and allocation strategy. In section 4, we use a mathematical cost model to evaluate the fragment cost. In section 5, we apply some experiments on a real DW to evaluate our approach. Section 6 contains concluding remarks.

2 STATE OF THE ART

DW fragmentation has been recently a subject of study in the literature. As it has a similar relational physical structure like relational databases, researches start by adapting classical fragmentation techniques (vertical, horizontal and hybrid fragmentation). These experiences demonstrate that DW fragmentation task becomes very difficult facing star model multidimensional aspect. In fact, proposals can be classified according to the DW architectures. In the following sections, we classify works according to three different architectures: centralized, parallel, and Distributed DW (DDW).

2.1 Centralized DW Fragmentation

Many research studies address the issue of fragmenting relational DWs to efficiently process analytical queries into centralized context.

(Datta et al., 1999) and (Golfarelli et al.,1999) use vertical fragmentation of the fact table; While (Datta et al., 1999) use vertical fragmentation technique to build the Curio index and to improve ad-hoc query performance, (Golfarelli et al.,1999) use this technique on warehouse views optimization. (Bellatreche & Boukhalfa, 2005) use the horizontal fragmentation technique. According to Bellatreche, the best way to fragment a star schema is by using the primary and derived fragmentation technique. The fact table fragmentation is based on the partitioning schemas of dimension tables. (Wu & Buchmaan, 1997) recommend combining horizontal fragmentation technique and vertical fragmentation technique for query optimization.

2.2 Parallel DW Fragmentation

Some research studies concentrate on the issue of fragmenting relational DWs into a parallel context: (Costa & Madeira, 2004) and (Furtado, 2004) focus on the horizontal partitioning of the DW among a cluster of computers. Their technique partitions the fact table over all nodes and replicates small dimension tables in each node. Authors exploit existing database management system partitioning techniques. While Costa and Madeira use a row-by-row round-robin partitioning technique, Furtado uses hash partitioning technique based on the analysis of the workload. (Ciferri & Souza, 2002) introduce The WebD2W System. The latter propose a DW horizontal fragmentation algorithm. This algorithm uses only one dimension table in the fragmentation process but no considerations were given to dimension attributes relationship hierarchies. (Ciferri, 2007) extends the work done by (Ciferri & Souza, 2002) and proposes the MHF-DHA algorithm. In this work, the fragmentation process uses multiple dimension tables as a basis to fragment the fact table and addresses the treatment of dimension attributes relationship hierarchies.

2.3 Distributed DW Fragmentation

Very few works concentrate on the issue of fragmenting relational DWs to the decentralization purpose. (Noaman & Barker, 1997) proposed a specific architecture for a DDW. This one is based on the ANSI/X3/SPARC architecture that has three levels of schemas: internal, conceptual and external. To distribute the DW, authors exploit a top-down strategy that uses horizontal fragmentation (Noaman et al., 1999). They proposed a horizontal fragmentation algorithm for the fact table. This algorithm is an adaptation of the work done by (Ozsu & Valduriez, 1991). (Wehrle et al., 2005) propose a data grid infrastructure to implement the DW decentralization. Their data model is based on "chunks" as atomic entities of a DW that can be uniquely identified. Then, they build contiguous blocks of these chunks to obtain suitable fragments of the DW. The fragments stored on each grid node must be indexed in a uniform way to effectively interact with the existing grid services.

2.4 Discussion

The fundamental aspects of the DW fragmentation have been studied by many works. But the most important of them have limited its use in a

centralized environment. What distinguishes our work from those works is that they aim to improve the data storage organization and the parallel queries execution by partitioning a table into multiple sub-tables. Besides, data allocation process is not considered as a problem since the DW is centralized. In our context we intend to decentralize the data storage of the whole DW according to the user's needs. Starting from a general model, we generate several sub-models. Each sub-model is specified as a future data mart. Works focused on the decentralized context are in general theoretical and lack of experimental evidence. Other studies have investigated technical aspects of the DW decentralization on a data grid. They proved experimentally that we can use easily advanced infrastructure; such as data grid to support DDW.

Therefore, it will be interesting to *evaluate and experiment* a DW fragmentation and allocation strategy for a decentralization purpose.

3 DW FRAGMENTATION AND ALLOCATION STRATEGY

The DW decentralization has two main phases: data fragmentation and fragment allocation. Data fragmentation consists in generating several sub-models from the general star model. To fragment the star model tables we use predicates as criterion. We apply the predicate construction technique (Ozsu & Valduriez, 1991) to construct the minterm predicate list. Then, we adapt the horizontal primary and derived fragmentation technique to fragment dimension and fact tables. Finally, we allocate each fragment to the corresponding site. We use three fragment allocation techniques: simple allocation, allocation with replication and allocation with some studied replications.

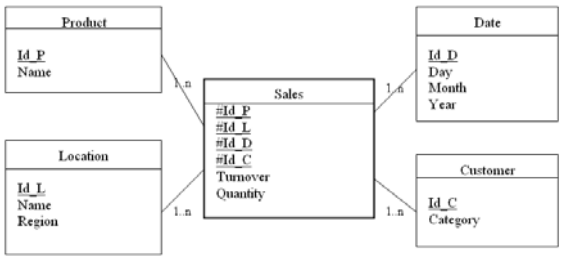

Figure 1: Sale activity star schema.

We choose, as an example, the sales activity schema (Figure 1). It consists of one fact table (Sales) and a set of dimension tables (Product, Location, Date and Customer) connected by foreign keys for all possible links. The fact table contains business facts or measures and foreign keys which refer to the primary keys in the dimension tables.

3.1 Predicate Construction Phase

In this phase, we start by extracting predicates from the most used queries trough the company sites. Then, we use the com_min algorithm (Ozsu & Valduriez, 1991) to generate minterm predicate list. According to the Com_min algorithm (Ozsu & Valduriez, 1991), a relation, or a fragment, must be partitioned "into at least two parts which are accessed by at least one application differently". We have chosen the Com_min algorithm because it guarantees data completeness, table reconstruction and fragment disjointness.

We take as an example the nine following queries running on the DW tables. We can extract from each query the used predicate list:

Q1. This application analyses sales turnover into site 1 according to the location ordered by products. The predicates are {Id_P=1,2 & Id_L=1}

Q2. This application analyses sales quantities into site 1 according to the location ordered by customer categories. The predicates are {Id_C=1,2,3 & Id_L=1}

Q3. This application analyses sales turnover quantities into the sites: 1, 2 and 3 according to the location ordered by customer categories and products. The predicates are {Id_P=1,2,3,4 & Id_C=1,2,3,4,5,6,7,8,9 & Id_L=1,2,3}

Q4. This application analyses sales turnover into site 2 according to the location ordered by products. The predicates are { Id_P=2,3 & Id_L=2}

Q5. This application analyses sales quantities into site 2 according to the location ordered by customer categories. The predicates are { Id_C=4,5,6 & Id_L=2}

Q6. This application analyses sales turnover quantities into site 2 according to the location ordered by customer categories and by products. The predicates are { Id_P=2,3 , Id_C=4,5,6 & Id_L=2}

Q7. This application analyses sales turnover into site 3 according to the location ordered by products. The predicates are { Id_P=3,4 & Id_L=3}

Q8. This application analyses sales quantities into site 3 according to the location ordered by customer categories. The predicates are { Id_C=7,8,9 & Id_L=3}

Q9. This application analyses sales turnover quantities into sites 3 according to the location

ordered by customer categories and by products. The predicates are { Id_P=3,4 , Id_C=7,8,9 & Id_L=3}

We suppose that:

- In the site 1: just the product Id_P=1 and Id_P=2 are on sale, there is one location Id_L=1 and the corresponding customers are Id_C=1, Id_C=2 & Id_C=3.
- In the site 2: just the product Id_P=2 & Id_P=3, are on sale, there are two locations Id_L=2 & Id_L=3 and the corresponding customers are Id_C=4, Id_C=5 & Id_C=6.
- In the site 3: just the product Id_P=1, Id_P=2, Id_P=3 & Id_P=4 are on sale, there are two locations Id_L=4 & Id_L=5 and the corresponding customers are Id_C=7, Id_C=8 & Id_C=9.

We recapitulate the complete and minimal dimension predicate list by site in the following lists:

Site1:	Site 2:	Site3:
Id_P=1,2	Id_P=2,3	Id_P=1,2,3,4
Id_L=1	Id_L=2, 3	Id_L=4,5
Id_C=1,2,3	Id_C=4,5,6	Id_C=7,8,9

After Appling the com_min algorithm to the preliminary predicate list, the corresponding complete and minimal predicate list is:

MP={P1:Id_P=1, P2: Id_P =2, P3:Id_P=3, P4: Id_P=4, P5: Id_L=1, P6: Id_L=2, P7: Id_L=3, P8: Id_L=4, P9: Id_L=5, P10: Id_C in [1,2,3], P11: Id_C in [4,5,6], P12:Id_C in [7,8,9], P13: Id_R=1, P14: Id_R=2, P15: Id_R=3}.

The next phase of the predicate construction technique is the conjunction of simple predicates which is called a minterm predicate. In fact, each simple predicate can be produced in its natural form or by negation. The number of minterm predicate is $z=2^m$; with m the number of simple predicate. Minterm predicates are exponential on the number of simple predicates; we can eliminate meaningless minterm predicates by identifying those which contradict to a set of implications.

We suppose as an example that we have the following implication list:

i1: P10 → ¬P11 ∧ ¬P12
i2: P11 → ¬P12 ∧ ¬P10
i3: P12 → ¬P10 ∧ ¬P11
i4: P13 → ¬P14 ∧ ¬P15
i5: P14 → ¬P13 ∧ ¬P15
i6: P15 → ¬P13 ∧ ¬P14
i7: P5 →¬P6 ∧ ¬P7 ∧ ¬P8 ∧ ¬P9
i8: P6 → P7 ∧ ¬P5 ∧ ¬P8 ∧ ¬P9
i9: P7 → P6 ∧ ¬P5 ∧ ¬P8 ∧ ¬P9
i10: P8→ P9 ∧ ¬P5 ∧ ¬P6 ∧ ¬P7
i11: P9 → P8 ∧ ¬P5 ∧ ¬P6 ∧ ¬P7

The implication list integrates fact implications, for instance: site 1 contains one location: id_L1=1, two product id_P=1, id_P=2 and id_C in [1, 2, and 3]. The implication list depends on the semantic of the application.

If we have the following fact implication list:

- i12: P13→P1 ∧ P2 ∧ P5 ∧ P10
- i13: P14 → P2 ∧ P3 ∧ P6 ∧ P7 ∧ P11
- i14: P15 → P1 ∧ P2 ∧ P3 ∧ P4 ∧ P8 ∧ P9

We will obtain the corresponding fact minterm predicates list:

- m1: Id_P=1 ∧ Id_P=2 ∧ Id_L=1 ∧ Id_C IN [1,2,3] ∧ Id_R=1
- m2: Id_P=2 ∧ Id_P=3 ∧ Id_L=2 ∧ Id_L=3 ∧ Id_C IN [4,5,6] ∧ Id_R=2
- m3: Id_P=1 ∧ Id_P=2 ∧ Id_P=3 ∧ Id_P= 4 ∧ Id_L= 4 ∧ Id_L=5 ∧ Id_C IN [7,8,9] ∧ Id_R=3

3.2 DW Fragmentation Phase

The fragmentation phase consists, first, in fragmenting the dimension table according to the minterm predicate list (Primary horizontal fragments). Then, we use a semi-join (α) operation between dimension primary horizontal fragments and the fact table to generate horizontal derived fragments. Table 1 contains the primary horizontal fragments list of the previous example.

Table 1: Primary Horizontal fragments.

R1=σ $_{pa\ R=1}$(Region)	L1=σ $_{pa\ L=1}$(Location)	C1=σ $_{pa\ C=1,2,3}$(Country)	P1=σ $_{pa\ P=1}$(Product)
R2=σ $_{pa\ R=2}$(Region)	L2=σ $_{pa\ L=2,3}$(Location)	C2=σ $_{pa\ C=4,5,6}$(Country)	P2=σ $_{pa\ P=2}$(Product)
R3= σ $_{pa\ R=3}$(Region)	L3=σ $_{pa\ L=4,5}$(Location)	C3=σ $_{pa\ C=7,8,9}$(Country)	P3=σ $_{pa\ P=3}$(Product)

After applying derived horizontal fragmentation on table Sales, we obtain the following derived horizontal fragment list: F^1 , F^2 and F^3

- F^1 = α(Sales, R1) ∩ α(Sales,P1) α (Sales,P2) ∩ α(Sales,L1) ∩ α(Sales,V1).
- F^2 = α(Sales,R2) ∩ α(Sales,P2) ∩ α(Sales,P3) ∩ α(Sales,L2) ∩ α(Sales,V2).
- F^3 = α(Sales,R3) ∩ α(Sales,P1) ∩ α(Sales,P2) ∩ α(Sales, P3) ∩ α(Sales,P4) ∩ α(Sales,L3)∩ α(Sales,V3).

3.3 Allocation Phase

There are three possible allocation strategies: simple

fragment allocation, allocation with fragment replication and allocation with some studied fragment replication.

We take as an example the fragments uses frequency matrix. The columns present horizontal fragments, rows represent the three sites of the company, and each cell contains the frequency of a fragment uses.

	R1	R2	R3	L1	L2	L3	C1	C2	C3	P1	P2	P3	P4	F1	F2	F3
S1	30	30	30	30	30	30	30	12	6	30	30	30	30	30	15	15
S2	0	30	0	0	30	0	8	30	12	0	25	30	0	0	30	0
S3	0	0	30	0	0	30	13	10	30	22	3	6	30	0	0	30

3.3.1 Simple Allocation Strategy

Each fragment is allocated into the site where it is the most frequently used. This option is generally chosen when the accesses cost between the organization sites does not have negative impact on the DW performance. The corresponding allocation matrix is given bellow.

	R1	R2	R3	L1	L2	L3	C1	C2	C3	P1	P2	P3	P4	F1	F2	F3
S1	A	U	U	A	U	U	A	U	U	A	A	U	U	A	U	U
S2	O	A	O	O	A	O	U	U	U	O	U	A	O	O	A	O
S3	O	O	A	O	O	A	U	U	A	U	U	U	A	O	O	A

The columns present primary horizontal fragments of dimension tables. The columns R1, R2 and R3 present primary horizontal fragments of the dimension table Region. L1, L2 and L3 present primary horizontal fragments of the dimension table Location. P1, P2, P3 and P4 present primary horizontal fragments of the table Product. F1, F2, F3 present derived horizontal fragments of the fact table Sales. The matrix rows represent the three sites of the company. Each cell contains the state of each fragment on the corresponding site. A indicates that a fragment is allocated on a site. O indicates that a fragment is out or non-existing on a site. U indicates that a fragment is used or accessed by a site but it's not replicated in it.

3.3.2 Allocation with Replication Strategy

In this strategy, each fragment is allocated where it is used. We opt for the local data accesses rather than the data remote accesses. The corresponding allocation matrix is given bellow.

	R1	R2	R3	L1	L2	L3	C1	C2	C3	P1	P2	P3	P4	F1	F2	F3
S1	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
S2	O	A	O	O	A	O	A	A	A	O	A	A	O	O	A	O
S3	O	O	A	O	O	A	A	A	A	A	A	A	A	O	O	A

3.3.3 Allocation with some Studied Replication Strategy

In this strategy, each fragment is allocated where it is the most frequently used and replicated into sites in which the data remote accesses cost is elevated. The corresponding allocation matrix is given bellow.

	R1	R2	R3	L1	L2	L3	C1	C2	C3	P1	P2	P3	P4	F1	F2	F3
S1	A	A	A	A	A	A	A	U	U	A	A	A	A	A	U	U
S2	O	A	O	O	A	O	U	A	U	O	U	A	O	O	A	O
S3	O	O	A	O	O	A	U	U	A	A	U	U	A	O	O	A

To evaluate generated fragments, we conduct numeric and experimental studies. Numeric studies are based on the adaptation of the known objective function developed in (Chakravarthy, 1992). Experimental studies are based on distributing a real DW by using the data set of the APB1 Benchmark (OLAP Council, 1998).

4 DW FRAGMENT EVALUATOR

Our first strategy to evaluate DW fragments is to use a cost model. Each fragment is evaluated by its local processing cost and remote one on each site. It is supposed that there are no data redundancies between fragments. Fragments are allocated to sites where they are the most used. DW Fragment Evaluator (DWFE) computes the processing cost of each fragment. The Fragment Evaluator (FE) is given by:

$$FE = E_M^2 + E_R^2 \quad (1)$$

Where E_M^2 is the local horizontal fragment access cost and E_R^2 is the remote one.

4.1 Local Fragment Access Cost

The local horizontal fragment access cost E_M^2 is the first component of the DWFE. *As an adaptation of the work presented by Chakravarthy in* (Chakravarthy, 1992), E_M^2 is given by:

$$E_M^2 = \sum_{i=1}^{M} \sum_{t=1}^{T} [\, q_{tc} * |S_{itc}| (1 - |S_{itc}| / n_i)] \quad (2)$$

Where M is the total number of fragments; T is the total number of transactions in the site c; $|S_{itc}|$ is the number of rows in F_i accessed by a transaction t at a site c; q_{tc} is a frequency of an application t at a site c and n_i is the total number of rows contained in a fragment F_i. This component uses the square-error criterion as given by Jain in (Jain, 1988) for data clustering. The objective here is to obtain a fragment which minimizes the square error for a fixed number of fragments. This criterion assigns a penalty factor whenever local rows are accessed in a particular fragment. (Chakravarthy, 1992)

4.2 Remote Fragment Access Cost

The remote horizontal fragment access cost E_R^2 is

the second component of the DWFE. For each transaction running on a fragment, we compute the ratio between the number of remote rows to be accessed by a transaction *t* and the total number of rows in each remote fragment. The remote horizontal fragment access cost which is an adaptation of the work presented by Chakravarthy in (Chakravarthy, 1992) is given in the following equation:

$$E_R^2 = \sum_{t=1}^{T} \sum_{K \neq i} \left[q_{tc}^2 * |R_{itk}| * |R_{itk}| \frac{R_{itk}}{n_{itk}^r} \right] \quad (3)$$

Where: $|R_{itk}|$ is the number of relevant remote row access in fragment *k* accessed remotely with respect to a fragment *i* by a transaction *t* at a site *c*; n_{itk}^r is the total number of rows in fragment *k* accessed remotely with respect to a fragment *i* by a transaction *t*.

We use the DWFE to measure the performance of a fragmentation schema. A lower DWFE value means fewer penalties and thus; better performance. We take as an example the following row numbers (Table 2). We use this example to computes the DWFE for each fragment.

Table 2: Row sets on each site.

	Query	Region (3Rows)	LOCATION (5Rows)	COSTOMER (9Rows)	PRODUCT (4Rows)	SALES (100000Rows)
S1	Q1	R1(1Row)	L1(1Row)	C1(3Rows)	P1(1Row) P2(1Row)	F1(30000Rows)
	Q2	R1(1Row)	L1(1Rows)	C2(3Rows)	P1(1Row) P2(1Row)	F1(30000Rows)
	Q3	R1(1Row) R2(1Row) R3(1Row)	L1(1Row) L2(2Rows) L3(2Rows)	C1(3Rows) C2(3Rows) C3(3Rows)	P1(1Row) P2(1Row) P3(1Row) P4(1Row)	F1(30000Rows) F2(30000Rows) F3(40000Rows)
S2	Q4 Q5 Q6	R2(1Row) R2(1Row) R2(1Row)	L2(2Rows) L2(2Rows) L2(2Rows)	C2(3Rows) C2(3Rows) C2(3Rows)	P2(1Row) P3(1Row)	F2(30000Rows)
S3	Q7 Q8 Q9	R3(1Row) R2(1Row) R2(1Row)	L3(2Rows) L3(2Rows) L3(2Rows)	C3(3Rows) C3(3Rows) C3(3Rows)	P3(1Row) P4(1Row)	F3(40000Rows)

To allocate each fragment, we test three cases: (1) fragmentation with full copy resides on site 1; (2) fragmentation with one copy resides on every site and (3) fragmentation with some fragment replication. We present in table 3, table 4 and table 5 the local access cost and a remote access cost of each table.

Table 3: CASE 1: Fragmentation, full copy resides on site 1.

Tables	Local Access Cost	Remote Access Cost	FE Values
Customer	129,6	556,2	685,8
Product	154	87	241
Location	96	144	240
Sales	2565000	2250000	4815000

Table 4: CASE 2: Fragmentation copy resides on each site.

Tables	Local Access Cost	Remote Access Cost	FE Values
Customer	500,4	0	500,4
Product	415	0	415
Location	312	0	312
Sales	6615000	0	6615000

In case 1, the fact table is centralized in one site, the remote access cost is very high. In case 2, each fragment is replicated there where its frequency of use is positive. The remote access cost is always equal to 0, because the fragment is locally accessed by the decision support system. The Local access cost is very high for the sales table because this one is generally characterized by a huge number of rows. Consequently, replicating the fact table in each site is not a good option.

Table 5: CASE 3: Fragmentation with some replication.

Tables	Local Access Cost	Remote Access Cost	FE Values
Customer	345,6	0	345,6
Product	359	0	359
Location	312	0	312
Sales	1350000	1050000	2400000

In case 3, we remark that all local access costs are fewer than the case 2 because just some fragments are replicated.

We can deduce that if an organization is distributed geographically and if its decision support system is disseminated on several distant sites, it is more interesting to adapt DDW architecture since it gives better performance. In figure 2, we present the FE values for the three cases. It is clear that fragmentation with some fragment replication gives lower costs than fragmentation with full copy resides on site 1 and fragmentation with one copy resides on every site.

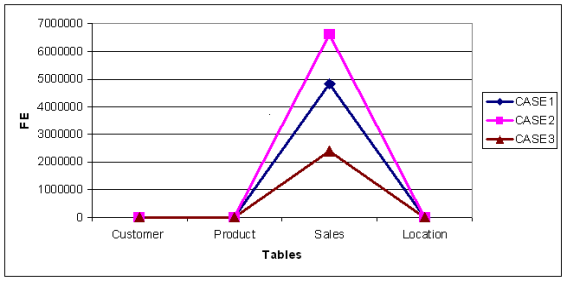

Figure 2: Fragment allocation strategies comparison.

5 EXPERIMENTAL EVALUATION

We have conducted an experimental study to evaluate our proposal. We have used the dataset from the APB1 benchmark (OLAP Council, 1998). The star schema of this benchmark has one fact table Actvars (33 323 400 tuples) and four dimension tables (Figure 3): Prodlevel (9 900 tuples), Custlevel (990 tuples), Timelevel (24tuples) and Chanlevel (10 tuples). We have considered a workload of 36 queries with 27 selection predicates. We have conducted experiments using Oracle 11g. The data set of APB1 benchmark is created and populated using generator programs offered by APB1. We use three machines having the same following characteristics. Intel Core2Duo, 2 Gb of memory.

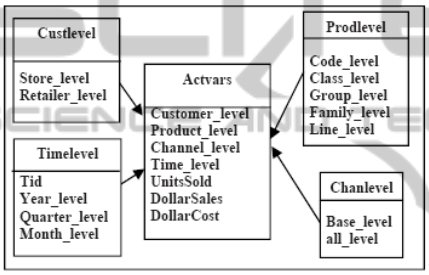

Figure 3: Star schema of the benchmark APB1 Sale activity star schema.

5.1 APB1 Dataset Benchmark Decentralization

We give bellow the complete and minimal predicate list to the dimension table (We replace the attributes Product_I and Customer_I by sequential numbers to simplify the tasks of decentralization):

P1: Channel_id = 'FH8X1SHVDLLC', P10: Product_I in [1..10000], P19: Customer_I in [1..1000]
P2: Channel_id= 'JPDO6VE1OWKB' P11: Product_I in [1001..2000] P20: Customer in [101..200]
P3: Channel_id= 'KQ0J3R74DPUL' P12: Product_I in [2001..3000] P21: Customer in [201..300]

P4: Channel_id= 'MDOKG8YES8UK' P13: Product_I in [3001..4000] P22: Customer in [301..400]
P5: Channel_id= 'PED69GC9HDXP' P14: Product_I in [4001..5000] P23: Customer in [401..500]
P6: Channel_id= 'PQXNSP5UQJNF' P15: Product_I in [5001..6000] P24: Customer in [501..600]

P7: Channel_id= 'QE3GST72V3OK' P16: Product_I in [6001..7000] P25: Customer in [601..700]
P8: Channel_id= 'S0GP76ZNP21M' P17: Product_I in [7001..8000] P26: Customer in [701..800]
P9: Channel_id= 'V45M6G2JVNEK' P18: Product_I in [8001..9000] P27: Customer in [801..900]

By applying the predicate construction technique, we obtain the following fact minterm predicate list:

PM1: P7^P8^P9^P16^17^18^25^26^27
PM2: P1^P2^P3^P10^P11^P12^P19^P20^P21
PM3: P4^P5^P6^P13^P14^P15^P22^P23^P24

We take as an example the following fragment uses frequencies matrix:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30
Site1	25	25	26	0	0	7	0	7	0	25	65	52	9	0	9	0	8	9	25	25	25	0	7	0	0	8	7	65	8	7
Site2	0	0	24	24	34	25	7	0	0	7	0	8	25	63	25	8	8	0	0	0	0	25	63	36	0	0	8	9	66	9
Site3	6	7	7	7	7	7	25	36	36	0	8	9	0	8	0	26	73	62	9	9	0	9	0	7	32	22	43	9	8	76

According to the fragment uses frequencies matrix, we generate the fragment allocation matrix using the simple allocation technique.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30
Site1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0
Site2	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0
Site3	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	1

Finally, we allocate the various fragments into three remote sites. The sites communicate through a virtual Private Network using a tunneling protocol. We use a workload of 36 queries that we execute to a centralized APB1 benchmark. We see in figure 4 that remote access increases the execution time of applications especially those whose users are geographically distant. Then, we execute the same workload into APB1 benchmark dataset distributed using our DW fragmentation and allocation approach.

5.2 Discussion

DDW architectures support multiple geographically-distributed business divisions. It allows each business division to have its own extraction, transformation, loading tools and multiple data marts. The central DW stores corporate-wide data and all components are synchronized across the DDW environment using a global metadata repository.

Queries workload can be classified into two categories: specific queries and general queries. Specific queries describe local needs (example: analyze sales amount by product of site 1). General queries describe needs of the whole company sites

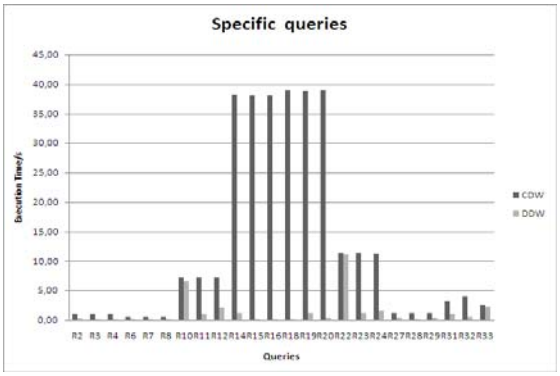

Figure 4: Specific queries execution time.

(example: compare sales amount by product of all the company sites). In figure 4, we present specific

queries execution time. Results show that DDW gives better performance for all the specific queries compared to the centralized context. Summed specific queries execution time is reduced by 82%. In figure 5, we present general queries execution time. Results show that DDW gives better performance for some general queries but not for others. For query 1 (sales analysis by product by year) and query 5 (sales analysis by quarter by year), the execution times are elevated compared to the centralized context. The specificity of those queries is that they are based on JOIN operations between all fragments localized at different distant geographical sites. In this case, the execution time is elevated compared to a CDW however the summed general queries execution time is reduced by 76%.

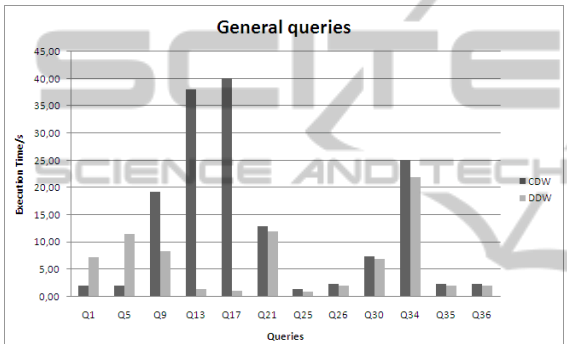

Figure 5: General queries execution time.

6 CONCLUSIONS

The design of DDW is an optimization problem requiring solutions to several interrelated problems including: data fragmentation, fragment allocation, and local optimization. Each problem can be solved with several different techniques, thereby making the distributed database design a very difficult task. Although there are many researches on the design of data fragmentation, most of them are focused on the centralized context and no considerations are given to the distributed allocation problem. Our work is considered one of the few dealing with data fragmentation and fragment allocation for the decentralization purpose. In this paper, we adapt a DW fragmentation approach using the predicate construction technique to generate the predicate list and the primary and derived horizontal technique to fragment dimension and fact tables. Then, we study three fragment allocation techniques into a distributed environment. First, we allocate fragment according to the simple allocation technique. Then, we replicate fragment there were they are used according to the fragment allocation with replication technique. Finally, we revise some fragment replication using the allocation with some fragment replication technique. After that, we conduct computing evaluation by using a DWFE and we compare the three fragment allocation cases. Finally, we implement our approach on a real DW. Results demonstrate that DW decentralisation gives better performance when data storage is distributed trough the company sites. But, the execution time for queries which are based on JOIN operations between all distributed fragments is higher than in a centralized context. As future work we intend to study OLAP distant queries optimization in a DW distributed context.

REFERENCES

Bellatreche, L., & Boukhalfa, K., 2005, An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse, *In DAWAK'07, 7th International Conference on Data Warehousing and Knowledge Discovery*. Volume 3589 of LNCS,

Chakravarthy S., et al., 1992. An objective function for vertically partitioning relations in distributed databases and its analysis, *Distributed and Parallel Databases journal*.

Ciferri, C. D. A. & Souza, F., 2002, Focusing on Data Distribution in the WebD2W System. *In DAWAK'02, 4th International Conference on Data Warehousing and Knowledge Discovery,* Vol. 2454 of LNCS.

Ciferri, C. D. A., et al., 2007. Horizontal fragmentation as a technique to improve the performance of drill-down and roll-up queries. *In SAC'07, 22nd ACM Symposium on Applied Computing.*

Costa, M. & Madeira, H., 2004. Handling Big Dimensions in Distributed Data Warehouses using the DWS Technique. *In DOLAP'04, 7th ACM Eleventh International Workshop on Data Warehousing and OLAP*.

Datta, A. & Ramamritham, K., & Thomas, H. M., 1999. Curio: A Novel Solution for Efficient Storage and Indexing in Data Warehouses. In VLDB'99, 25th International Conference on Very Large Data Bases.

Furtado, P., 2004. Workload-based Placement and Join Processing in Node-Partitioned Data Warehouses. *In DaWaK'04*, 6th *International Conference on Data Warehousing and Knowledge Discovery*., p. 38-47.

Golfarelli, M., Maio, D. & Rizzi, S., 1999. Vertical fragmentation of views in relational data warehouses. *SEDB'99. In Settimo Convegno Nazionale su Sistemi Evoluti Per Basi Di Dati.*

Jain, A. & Dubes, R., 1988. Agorithms for clustering Data. Prentice Hall Advanced Reference Series, Englewood Cliffs, NJ.

Noaman, A. Y. & Barker, K., 1997. Distributed data warehouse architectures. *Journal of Data Warehousing.* Vol. 2.

Noaman, A. Y., Barker, K., 1999. A Horizontal Fragmentation Algorithm for the fact relation in a Distributed Data Warehouse. *In ICKM'99, 8th International Conference on Information and Knowledge Management.*

OLAP Council, 1998. Apb-1 olap Benchmark, release 2. http://www.olapcouncil.org/research/bmarkly.htm.

Ozsu, M. T. & Valduriez, P., 1991. *Principles of Distributed Database Systems*. (pp. 657). Prentice-Hall, Inc. Upper Saddle River, NJ, USA

Wehrle, P., Miquel, M. & Tchounikine, A., 2005. A Model for Distributing and Querying a Data Warehouse on a Computing Grid. *In* ICPADS'05, *11th International Conference on Parallel and Distributed Systems*. IEEE Computer Society.

Wu, M.-C. & Buchmann, A. P., 1997. Research Issues in Data Warehousing. *In Datenbanksysteme in Buro, Technik und Wissenschaft*, pages 61–82.