People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Amar Telidji University - Laghouat
Faculty of science
Department of Mathematics and Computer science

Contribution to improving the performance of XML complex data warehouses

Thesis submitted to obtain Computer Magister Grade
Specialty ENGINEERING COMPUTER SYSTEMS (ESI)

By
Sellami BENAISSI

In front of the jury composed of:

Pr. Mohamed Bachir YAGOUBI	Professor at Amar Telidji University - Laghout	President
Pr. Abdelouahab MOUSSAOUI	Professor at Ferhat Abbas University - Setif	Supervisor
Dr. Youcef OUINTEN	A.Professor at Amar Telidji University - Laghout	Examiner
Dr. Abdelkrim TAHARI	A.Professor at Amar Telidji University - Laghout	Examiner

2015 - 2016

Dedications

To my Dear mother

To my Dear father, Allah's mercy

To my lovely wife

To my children: Meriam, Kutaiba and sirine

To my brothers: Mohammed and the rest

To all the beloveds

I dedicate this work

Acknowledgements

I thank Allah My god first, O Lord, praise to you and thank you.

I also extend my sincere thanks and praise to virtuous **Professor Mr. Abdelouahab MOUSSAOUI**, for the proper sponsorship to me during the study, with an apology to him for all shortening in its right.

And thanks go to the virtuous professors, members of jury, for honoring me to discuss the thesis.

And don't forget to thank everyone who helped me with the guidance and advice and support, and in particular Mr. Nadir Zinelaabidine, and Mr. Yagoubi Rached especially on moral support which greatly benefiting me and my friend the English teacher: Kamal belkacemi, also do not forget my friends and colleagues: Hawas, Mohammed Mekki, Hocine, Nacer and the rest.

And a link thanks to the loyal and sincere, which had helped me with the effort and time and all that have, my dear wife and Dear beloved: Habiba.

Thank you all

Thanks

ملخص

لقد تم اقتراح العديد من النماذج وتقنيات التحسين في الأدبيات. وقد قدمنا في بحثنا هذا التقنيات الرئيسية لتحسين أداء مستودعات البيانات، مثل الفهرسة و المرئيات المجسدة و التجزئة بأنواعها، فالعمل المقدم في هذه المذكرة يدخل في إطار التصميم الفيزيائي لمستودعات البيانات المعقدة XML، وقد اقترحنا مقاربة لتحسين أداء الإستعلام تستند على التجزئة بالبعثرة. و هي المقاربة التي أسميناها XHaFrag إختصارا لـ" XML Hash Fragmentation"، تتركز على تجزئة مستند الوقائع فقط باعتباره المستند الأكبر و الأكثر تحديثا من باقي المستندات (الأبعاد)، تتميز ببساطة و وضوح المبدأ، وسهولة التطبيق و التركيب، وتنويع الدالة، و عدم التقيد بدالة بعثرة واحدة، كما تتميز بإمكانية العمل مع مختلف الحقول أو خصائص مستند الوقائع، و قمنا بتجربتها على منصة تجارب XWeB،و هو الأمر الذي أثبت لنا نجاعة و فائدة هذه المقاربة، وأنها فكرة قابلة للتطوير و التطبيق.

كلمات رئيسية، XML، نظم دعم القرار، مستودع البيانات، تحسين الأداء، الفهرسة، التجزئة، دالة البعثرة.

Abstract

Many models and optimization techniques have been proposed in the literature. We have presented in our research the main techniques to improve the performance of data warehouses, such as indexing, materialized views, and fragmentation with its types. The work presented in this thesis is part of the physical design of the XML complex data warehouse. We have proposed a query optimization approach based on the concepts of Hash Fragmentation, which we called XHaFrag short for "XML Hash Fragmentation", it is based on the fragmentation of the facts document only, because it is the largest document, and the most updated among the rest of the documents (dimensions). This approach is characterized by the simplicity and clarity of its principle, ease of application and implementation. It is also characterized by diversity of function, and non-compliance with one hash function, and the possibility of working with various fields or properties of facts document. This approach was subject to experiment on the benchmark XWeB, which proved its efficiency and its usefulness and the experiment also proved that it is an idea that can be developed and applied

Keywords: XML, decision support systems, data warehouse, improve performance, indexing, fragmentation, hash function.

Table of contents

LIST OF FIGURES

List of Tables

INTRODUCTION

With the emergence of modern communication technologies, especially the World Wide Web (Internet), and the widening impact of the companies and the enterprises - particularly international companies – the data of these companies is no longer locked in closed offices of the same building, but has become circulating across the entire world; a world that has become like a small village - if not a house -, its population interact and deal with each other. This fact has produced a terrible increase in the volume of data and mutual information which must be saved. This situation created two main challenges: 1- to secure and preserve this data, 2- to inquire about it and make use of it to make the right decision at the right moment and extract knowledge. These things combined with others were a strong reason for the emergence of an important science technology called the "Decision Support System" which includes several phases. It starts with collecting the dispersed and proliferated data in one big location represents a huge database called "Data Warehouse" which is a set of objective data (subject-oriented), integrated, time-variant (historical), and non-volatile (permanent) used to support management decisions (Inmon, 1992) . The most important framework for data exchange is through networking and across the World Wide Web, and the techniques used on them are extremely important. The most important of these techniques that is concerned with the exchange and save pages and data is XML technology (eXtensible Markup Language). Using this technique was because it provides an easy and fruitful mechanism to save the data regardless to its diversity and differences, The mutual data is not just a simple text but may be varied and complex files, such as images, audio, video and binary files which are called complex data. This, led to the emergence of complex data warehouses, or XML data warehouse.

But with the increasing volume of data warehouses, the strength of the competition between the companies using them, and the importance of the decisions extracted from these warehouses and their influence on the permanence and stability of these companies, it is necessary to reach this decision with adequate speed and accuracy. Dredging through all the data saved in a huge data warehouse requires a lot of time and effort, thus the solution is to look for innovative ways to facilitate the process of enquiry (especially that the queries used on the data warehouse are complex ones. This is because the purpose of them is to reach the right decision). These ways are found to speed up the inquiry process ,i.e. improve the performance of data warehouses.

Data warehouses are originally databases. Technologies applied and adopted to improve the performance databases can also be used to data warehouse as the relational data warehouses is the origin of complex data warehouses XML. The innovative technologies to improve the performance of the first may be fit for the other, Therefore, improving the performance of the storage technologies has taken a significant part in this thesis. We have traced previous studies and research that focused on these techniques.

Improving the performance includes two directions are : First: the field of capacity and storage efficiency. Second : - and this is our focus – the field of speed and accuracy inquiries about data. In the first direction we may find techniques such as: Compression of databases and data warehouses. In the second direction which got too much interest among researchers - as the large storage size is no longer a problem now because of the tremendous progress in storage technologies - we find a variety of techniques of improving the performance of warehouses, such as: Indexing, materialized view, Fragmentation, ... and others. Some of these techniques requires keeping a copy of data and data redundancy, such as indexing. We call this class: "Redundant Techniques", and the other is called: "No Redundant Techniques".

We have divided this thesis into an introduction and four chapters revolving around the title "Contribution to Improving the Performance of XML Complex Data Warehouses ", so we tried to take the subject in all its aspects; research and excavation. We also deepened in some aspects, because of their importance and the direct relationship to the research topic.

- In the first chapter, we dealt with data warehouses, because data warehouses is an integral part of a very important process is the decision-making process, it was necessary to talk about it with some prolixity, this chapter has been divided into two sections, *The first section* dealt with decision support systems, we dealt with the concept of decision support and levels of decision and types of then decision-making process with its various stages, beginning from the source data to the exploitation phase in order to obtain the knowledge and access to appropriate decisions, through the process of ETL, which means the extraction and transform and load data, and is a very important process, without it you cannot create a data warehouse, which is addressed in the *second section*: Where I touched on the data warehouses, I spoke about the beginning of warehouses and its history, and its definition (especially definition: WH Inmon and definition Ralph Kimball) and their characteristics which spoken by Inmon in its definition are: subject-oriented, integrated, time-variant and non-volatile, Then we talked also to the goals of data warehousing and types of data within which detailed data and aggregate and MetaData, in this section also I spoke about OLAP, its rules, and multi-dimensional model. And I devoted to data warehouses this Section, although it an integral part of the decision support system because of its importance and its direct relation to the subject of thesis without the rest of the stages of the system.

- The second chapter is addressed to explain XML technology and complex data warehouses XML, it has been divided to three sections. In *the first section*: beginning, we talked about complex data, its concept and its characteristics, because our theme is about complex data warehouse, and because XML documents is one of the semi-structured complex data, I addressed for XML in terms of its history, its rules, techniques and associated standards (DTD, SAX, RELAX NG, DOM, XSD, ... and others), The XML technology is a vast space in which we find: The Main language XML and its rules, then the XML Parser to interpret the data in the documents XML, As there are two types of them are: Tree-based Parser and Event-based Parser, then VALIDATION come to make sure that the XML document in the structure complies with the conditions set out in accordance with these Schemas such as DTD or XSD (XML Schema), To display XML pages in format, size, and style that we want, we find languages of styles that are interested pages appearance and format of the document in terms of Fonts, colors, places of images, backgrounds and others. Among the most important criteria, we find CSS: Cascading Style Sheets and XSL: Extensible Stylesheet Language, then we find the standards and technologies that can navigate and query data in documents such as XQuery, XPath, XUpdate .... and others,

- *The second section*: We have moved to talk about XML databases, and database management systems DBMS that are saving and processing data within files and XML documents, there are databases relational XML where data is stored on the XML document into a relational Databases (XML-Enabled Databases), also, there are native XML database, two types of Native XML databases according to the structure are: the text-based and model-based. *The third section*: After briefing the XML language and technology, Then reflect on the XML database, we got to one of the most important points in Our thesis, a Complex Data Warehouses XML, Because the data warehouse is in fact a huge database, applies to what we discussed earlier about XML databases in the previous section, We have addressed this section in the concept of XML data warehouse, and previous studies in the field of modeling and management of XML data warehouses, we have addressed this section in the concept of XML data warehouse, and previous studies in the field of modeling and management of XML data warehouses.

- In the third chapter, and after fulfilling an important part of the thesis title, a "complex data warehouses XML", the first part of it remained "improve performance", which is addressed in this chapter, It was just about techniques and ways to improve performance in databases and data warehouses XML, there are two types of these techniques are: Redundant Structures, No Redundant Structures, In the first, we find Vertical Fragmentation, Mixed Fragmentation, Materialized Views, Index, and Management Of Buffer. In The No Redundant techniques, we find Horizontal Fragmentation, Parallel Processing, and Scheduling Queries, Redundancy means repeat data, which requires an additional size.

- The last chapter is the butter And purpose of the Search, as in which we suggested a new approach to improve the performance of XML complex data warehouses, we called this approach XHaFrag, short for: XML Hash Fragmentation, and as is clear from the naming, the proposal is based on the principle: horizontal Fragmentation by Hash Function, This means that the division of the data warehouse is using the primary horizontal Fragmentation according to the hash function, In the proposal, only the facts document is fragmented without the rest of the warehouse documents, because this document is the largest and the most changes, as we have the experience of performance on XWeB platform, and we explained the results, which show the role of this method of improving the performance of queries, the approach is characterized by simplicity and ease of installation, and is an initial idea can be built upon and developed.

CHAPTER 1

Decision Support Systems And Data Warehouses

Summary:

In this chapter, we present various concepts related to decision support systems and data warehouse, in terms of modeling, design and strategy, and exploitation, and other.

1) Chapter 1: Decision Support Systems and Data Warehouses

1.1 SECTION I: DECISION SUPPORT SYSTEMS

The age of computer networks and the Internet is the appropriate description for the time being, thanks to the existence and influence of these technologies in all aspects of life, in economic, scientific and social fields and even political, in fact, the Internet has become the area for human interaction and relationship, shopping and the exchange of experiences and gains .... and others. Networks and the Internet are also considered as an important source of growing data day after day.

Due to the Web proliferation and other factors, including globalization and the development of communication means (Figure 1), companies should be able to compete strongly in the global markets, they should have to consider in their strategic planning, production, and strategies marketing, financing and investment decisions taken by them, To perform these tasks, the company must be open to new technologies, among the most important of these techniques, we find those related to databases and the exchange and analysis of information, such as XML techniques ، And Business Intelligence (BI) which is defined as a set of techniques for processing, evaluation, presentation and preparation of the data in order to be meaningful and easier to understand and help to make the right decision (Kalakota and Robinson, 2000)(Negash, 2004) (Yeoh and Koronios, 2010) .It also gives a clear view to managers in the performance of the company to improve its ability to respond faster than its competitors to new opportunities or market risk, BI is based on an information system called decision support DSS system as opposed to system operational information, as companies have benefited more from these techniques gave more gains (depending on the nature and activity the company).

All companies strive to be the best through:

- Retain customers by providing products and services they are interested in.
- Possible risk management.
- Improve its activities.
- Intelligent and optimal exploitation of the data stored.
- Having more information than its competitors.
- Proactive

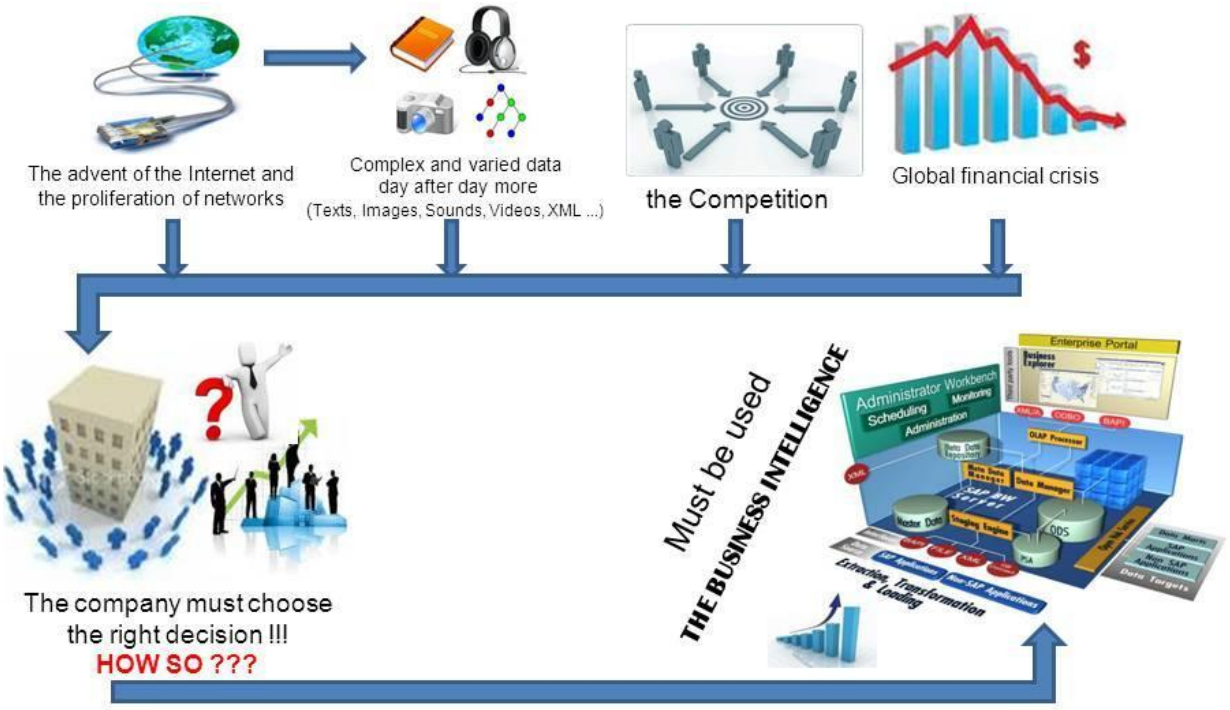

Figure 1 The motives of the use of Business Intelligence

With the complexity of information technologies, and unstable environment for companies with multiple mergers and acquisitions, and the great competition and the globalization of exchanges, the decision-makers need a clear and quick vision for their business at every moment, with easy-to-use tools, without causing harm to the existing production system. This what Decision Support Systems **DSS** provides sufficiently. These systems were the outcome of evolution in information technology during the seventies of the last century and evolved during the eighties. They were not a technical revolution, but a natural progression for computer using method, where the decision support system simply focuses on providing appropriate support to improve the quality of decisions, which depends on several factors, the most important, including:

- The suitability of the information available.
- The adequacy of the information available to the number of proposed alternatives.
- The appropriateness of the models used to analyze the problem and all this at a specific point in time which is the decision-making time.

The decision support systems are working to achieve these demands by integrating data and models and software in an efficient system for decision-making, The concept of decision support systems took ample space in many of the literature business management, because of its importance to contemporary systems as a result of the multiplicity and diversity of variables These systems help managers decision-makers on the integrated thinking for the time being and for the future, in an organized manner based on the data processing ,(Nahmias, 1997) pointed out that the decision support systems are designed software consists of a system language and knowledge base and tackle problems systems, are used to support decision-making process, and (Russell and Taylor, 2003) has been known to decision support systems as systems for data processing in which human intelligence interacts with information technologies to support decision-making process, as these systems are to adapt to the capabilities of the beneficiary as well as the ability to display information known to him by a methodology. Based on the above definitions, and other definitions (Ivancevich et al., 1994) (Kalta et al., 1998) (O'brien, 1998) (Keen and Morton, 1978) can be defined decision support system summarized as follows:

1.1.1 Definition 1: Decision Support System

Decision Support System is an interactive information system that works to collect, transfer, analysis and processing data in order to facilitate the decision-making process (especially unstructured and semi-structured decision) or in order to obtain knowledge

1.1.2 Levels of decision-making:

Decision-making process varies depending on the organizational level, which carries it out. Anthony has divided the decisions of the organization into three sections (Anthony, 1965):

1.1.2.1 Strategic Decision:

Strategic (or long-range) decision is used for managing long term business plans and goals. It is used to provide critical information in a timely fashion to support organizational planning and decision making efforts enabling the organization to improve its competitiveness and business performance. It is something which every business needs in order to compete in a dynamic, difficult and competitive business environment. Top Management use the key performance indexes (KPI's) to check if the parameters used to achieve long term business goals

have achieved the set target or no such as growing market share, reducing costs, and increasing revenues..

1.1.2.2 Tactical (Management) decision :

Tactical (medium-range) decision is the application of BI to analyze business trends frequently comparing a specific metric to the same metric from previous month or year. As business initiatives are launched (marketing campaigns, new products, for example) to help align actual business performance with planned performance, tactical BI analytics are employed by senior managers, business analysts, and line-of-business (LOB) managers to measure and optimize the performance of those initiatives. This tactical BI analyzes business operations over a period of days, weeks, or months.

1.1.2.3 Operational Decision :

Operational Decision delivers information to the point of business, the front-liners of business where information is used as part of operational business, Operational (short-term) activities provide the day-to-day flexibility needed to meet customer requirements on a daily basis within the guidelines established by the more aggregate plans discussed above. Short-range operating schedules take the orders directly from customers, or as generated by the inventory system and plan in detail how the products should be processed through a plant. In most cases detailed schedules are drawn up for one week, then one day and finally one shift in advance. The schedules involve the assignment of products to machines, the sequencing and routing of orders through the plant, the determination of replenishment quantities for each stock keeping unit and so on. .

According to what has been said above, characteristics of levels and decisions can be summarized in (Table1).

Later another level of decisions has been added, namely (Laudon and Laudon, 2000) :

1.1.2.4 Knowledge-level Decision:

Knowledge-level decision making deals with evaluating new ideas for products and services; ways to communicate new knowledge; and ways to distribute information throughout the organization. support knowledge and data workers in an organization. The purpose of knowledge-level systems is to help the business firm integrate new knowledge into the business and to help the organization control the flow of paperwork. Knowledge-level systems, especially in the form of workstations and office systems, are the fastest growing applications in business today.

Decisions properties	Strategic decisions	Administrative decisions	Operational decisions
Time horizon	Long term	Medium-term	Short-term
Frequency	Unique	Low Frequency	Many and too frequent
Degree of uncertainty	Very high	High	Low
Reflectivity	Almost zero	Low	High
Level decision	General Directorates	Functional and operational departments	Decentralized decisions

Table1 Levels of decision-making and their characteristics

1.1.3 Types of Decisions:

The sections and levels of decision depends on the type of the decision itself, and the type of decision varies according to the nature and degree of recurrence, as may be the decision relates to routine operations and clear recur periodically or otherwise, Based on this (Simon et al., 1987) divided decisions to :

1.1.3.1 Structured Decision:

Structured Decision has frequent and routine nature and is taken according to clear procedures, for this they do not need to be processed each time. These decision-making procedures are pre-programmed according to defined criteria, preferably by the operational and administrative levels to take such decisions and lack of concentration in the highest levels to ensure the speed of doing business and not disable it .

1.1.3.2 Semi structured decision:

When actions are predetermined but not enough to make the decision, the problem should be defined to design alternatives and choose the most appropriate alternative; in this case the problem is known and has a pre-approved procedures .

1.1.3.3 No structured Decision :

Are those decisions that require from decision-makers to do the trial and evaluation in order to identify the problem, and these decisions are unfamiliar and significant and non-routine, and there is no clear understanding or full approval about the decision that should be taken, such as the expansion of sales to foreign markets. They are non-routine decisions where the procedures are non-specific and taken in the conditions of uncertainty and in senior management levels.

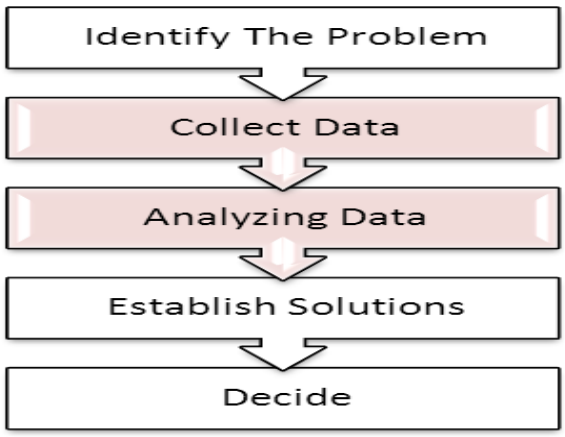

Figure 2 Stages of the decision support process

1.1.4 The decision making process

DSS involves five key stages, that reflect a series of successive procedures. However, the sequence of these procedures does not negate the existence of a state of feedback between those phases, Which may convey the decision support in a late phase due to difficulties or problems impeding progress in one of the subsequent phases, For example, it might identify the problem and study the surrounding environment, as well as formulating possible alternatives, However, unexpected reactions may arise when making the decision to activate, which is due to those who hold decision-making to the early stages of re-search for new formulas and alternatives.(McLeod and Schell, 2001) (Davis and Olson, 1984).

In short, the phases of decision support and successive procedures involved beneath it can be described by the following points:

1.1.4.1 Problem Identification Phase:

The first step towards making the best decision is to define the dimensions of the problem under study and the surrounding environment. This is done by collecting the evidence and data that constitutes informatics infrastructure of the case requiring support. There may be a problem, the decision-maker seeks to solve or there is an opportunity, the decision maker seeks to take advantage of it.

1.1.4.2 Data Collection Phase:

This data may be from internal or external sources, and this according to the desired purposes in order to reach the decision-making process.

1.1.4.3 The phase of formulating possible alternatives:

This phase is the first real stage for those in charge of decision support, during which the proposed initial form of all possible alternatives and choices is formulated. This phase is accompanied by the use of quantitative and descriptive analysis methods, for the implementation of simulation and expected scenarios in order to predict the results that may come out of the resorting to these alternatives. In addition to the formulation of the periodic follow-up phases of the implementation of programs and policies activate the systems, and the final assessment of yield generated practical results for those alternatives, this phase might be called the data analysis phase.

1.1.4.4 Selection Of Alternatives Stage:

This phase convey interest in supporting the decision to the higher levels of those who support it. It requires the need to resort to experts in the field of decision support, and those specialists issues depending on its forms, according to the issue under discussion. This stage requires careful and experienced in check to ensure the use of the alternative proposal of limited resources optimally, as well as ensure a logical statistics and indicators used in the construction and installation of the various alternatives. Concludes this stage to choose one of the options to act as a substitute team. Following the design implementation of this alternative on the ground plan.

1.1.4.5 The stage of decision-making:

This phase is the last stage of the decision-making process per cycle - As mentioned above, these stages can be repeated more than once - and by which design systems and periodic follow-up reports to ensure the harmonization of implementation of the plans and tasks previously depicted. This stage also concerned with the measurement of reactions, which may mean first reaction to those policies or programs, or to design a system to measure the reactions on a regular basis and in parallel implementation of the follow-up systems.

This is in terms of literature, but technically the decision-making process based on decision support system or intelligent business can be configured in three major phases, listed in the following figure:

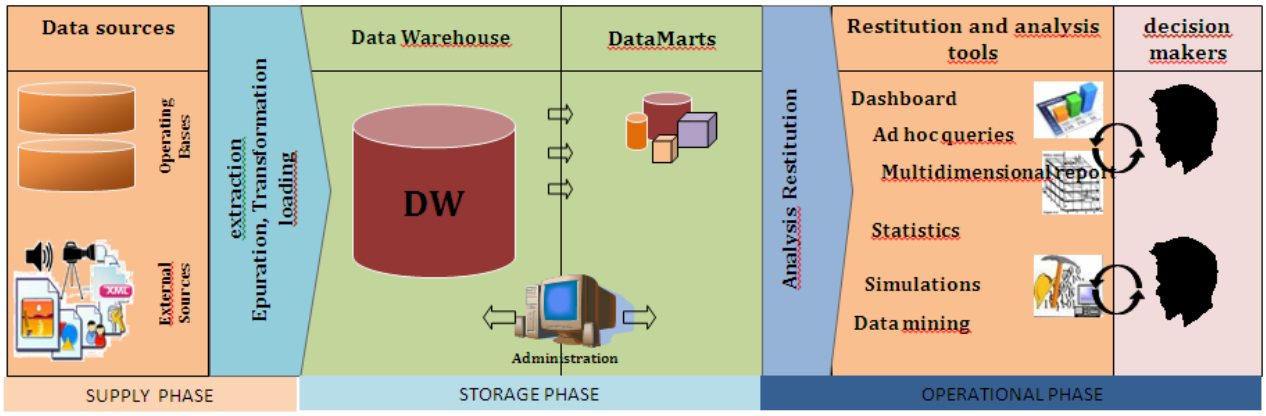

Figure 3 General Structure Of DSS

1.1.5 The General Structure Of The Decision Support System

We can divide the system into three main stages:

- Nutrition (Supply) stage
- Storage stage
- The exploitation phase

1.1.5.1 Supply phase

Decision support systems based primarily on data from the enterprise and external data, which may help in decision-making, and these data are a variety of structures and a variety of sources and heterogeneous ،it may be the result of operational databases (operational) or different files (texts, pictures, video, XML files(.....،. This diversity and heterogeneity makes it impossible to use the information in raw format, and this must filtering and transform and unify the structure of the data until they are ready for load and analysis, this is found in a series of complex processes gathered in one term is ETL (Extract. Transform, Load).

1.1.5.2 ETL process:

The objective of decision support system is to exploit enterprise data to arrive at the right decision, but given the diversity and the difference this data due to the difference source or the structure or formula ....., the benefit from the raw data to be difficult، which passes through several phases summarized in the term ETL (Extract. Transform, Load). (Which is a collection of various tools in the process of creating the raw data from various sources to become ready for use and exploitation, and this includes a set of procedures and elements, begin to:

1.1.5.2.1 Extract data

It is the process of reading the data from the source, the difficulty of this process lies in the multiplicity and diversity of the sources used, ،the difficulty increases with the complexity of the data and their recurrence rate ،During the extraction process being purified and filter the data and remove all unnecessary or redundant .After this step, the second step is:

1.1.5.2.2 Transform extracted data :

The extracted data has a different structure and nature from the structure of the data warehouse, which we want to be included in it, ،at this stage, the data is converted to match the data warehouse system criteria in terms of Coding, and abbreviations ،transfer element is responsible for the accuracy, validity and types of the data ،and is one of the most complex and the most important elements of ETL, and after extracting data from sources and convert it to be appropriate to the structure of the data warehouse comes the final stage in a process of ETL:

1.1.5.2.3 Load Data :

This phase is responsible for Load the extracted and converted data to the data warehouse ‹and during the load process must have a data warehouse offline‹ Therefore, it must be designed load tools to focus on efficiency and performance to reduce time out of service warehouse, and this is an important matter greatly, especially if the load being a day.

Note: ETL processes do not only exist at this stage (feeding stage), but may be found in other places during the decision-making process

1.1.5.3 Storage phase:

At this stage, are assembling and integrating Data in one place represents a database being exploited in the decision-making process, and there are multiple ways to store data in the *datawarehouse* or *dataMarts*, each with their pros and cons, and the system administrator is responsible for the choice among them, In the following section we will highlight more detail about data warehouses and storage methods.

No	ETL tools	Version	company
1.	Oracle Warehouse Builder (OWB)	11gR1	Oracle
2.	Data Services	XI 4.0	SAP Business Objects
3.	IBM Infosphere Information Server	9.1	IBM
4.	SAS Data Integration Studio	9.4M1	SAS Institute
5.	PowerCenter Informatica	9.5	Informatica
6.	Elixir Repertoire	7.2.2	Elixir
7.	Data Migrator	7.7	Information Builders
8.	SQL Server Integration Services	10	Microsoft
9.	Talend Studio for Data Integration	5.2	Talend
10.	DataFlow Manager	6.5	Pitney Bowes Business Insight
11.	Pervasive Data Integrator	10.0	Actian (Pervasive Software)
12.	Open Text Integration Center	7.1	Open Text
13.	Oracle Data Integrator (ODI)	11.1.1.5	Oracle
14.	Data Manager/Decision Stream	8.2	IBM (Cognos)
15.	Clover ETL	3.4.1	Javlin
16.	Centerprise	6.0	Astera
17.	DB2 Infosphere Warehouse Edition	9.1	IBM
18.	Pentaho Data Integration	4.1	Pentaho
19	Adeptia Integration Suite	5.1	Adeptia
20.	DMExpress	5.5	Syncsort
21.	Expressor Data Integration	3.7	QlikTech
22.	Relational Junction ETL Manager	5.3	Sesame Software

Table2 A list of the major products of ETL tools

Source : Site http://www.etltool.com

1.1.5.4 Exploitation phase:

At this stage, being exploit and use the stored data in order to obtain the knowledge and access to appropriate decisions, Data are analyzed using various statistical techniques, data mining, and research operations …etc.

1.1.5.4.1 Dataminig :

It is a process by which sort large amounts of data in order to extract relevant information. This term is used increasingly in the sciences to extract information from massive data sets resulting from modern experimental and observational methods, Is to extract knowledge patterns from large data sets through a combination of techniques from statistics and artificial intelligence with database management process, which is a step in the knowledge discovery in databases (KDD) where the use of analytical methods, such as: Neural Networks, Genetic Algorithms, Decision Trees, Hybrid Models to identify patterns and relationships in data sets, the process of knowledge discovery in databases by identifying patterns and trends in data collected using different methods such as: Classification, Sequential analysis, clustering and Association Rule،

1.1.5.4.2 Ad Hoc Reporting:

Through ad hoc reports, you can create and save report and modify it to get a particular display of the data .You can also choose the information you want displayed، organized as we want, and you can create tabs to view related data, these reports contribute to save time by creating reports provide the parties concerned with the data that you want to see it exactly ،it is also collects all information that you find relevant together, and can be easily share reports so that it can take faster decisions based on the data.

1.1.5.4.3 Operations Research :

(Source :Wikipedia) Operations research is a discipline that deals with the application of advanced analytical methods to help make better decisions. It is often considered to be a sub-field of mathematics. Employing techniques from other mathematical sciences, such as mathematical modeling, statistical analysis, and mathematical optimization, operations research arrives at optimal or near-optimal solutions to complex decision-making problems. Because of its emphasis on human-technology interaction and because of its focus on practical applications, operations research has overlap with other disciplines, notably industrial engineering and operations management, and draws on psychology and organization science. Operations research is often concerned with determining the

maximum (of profit, performance, or yield) or minimum (of loss, risk, or cost) of some real-world objective. Originating in military efforts before World War II, its techniques have grown to concern problems in a variety of industries.

1.1.6 Evolution of business intelligence market

Total turnover achieved in the business intelligence tools market, according to the study conducted by IDC's office in 2010 is about 9,016.5 million dollars to jump to 10,193.1 million in the year 2011 and then 10,966.2 million in 2012, which demonstrates the importance of this area and the attention span of the major corporate, five companies major dominates the commercial field of Business Intelligence market are SAP, IBM, SAS, Microsoft, Oracle, SAP company alone has around 20% of the business intelligence tools market in the world for several years (see Table3)،

Enterprises	Revenue (million$)			Share(%)			Growth(%)	
	2010	2011	2012	2010	2011	2012	2010-2011	2011-2012
SAP	1,866.4	2,074.5	2,165.4	20.7	20.4	19.7	11.2	4.4
IBM	1,302.6	1,484.1	1,534.3	14.4	14.6	14.0	13.9	3.4
SAS	975.2	1,062.8	1,133.8	10.8	10.4	10.3	9.0	6.7
Microsoft	801.0	899.8	1,044.1	8.9	8.8	9.5	12.3	16.0
Oracle	801.4	934.5	978.5	8.9	9.2	8.9	16.6	4.7
Other	3,269.9	3737.4	4110.1	36.3	36.6	37.6	14.3	10.0
Total	9,016.5	10,193.1	10,966.2	100.0	100.0	100.0	13.0	7.6

Table3 Worldwide Business Analytics Software Revenue by Leading Vendor,2010-2012

Source :IDC, June 2013

1.2 Section II : DATA WAREHOUSES

Data warehouses are modern ancient techniques (Haisten, 2003)، since the early days of relational databases, the idea arose to maintain a stock of historical data for reference when you need it ،The idea was primitive to create archives of keeping historical data in spite of the use of special techniques to retrieve this data from various storage media, Then writings appeared to establish techniques to take advantage of this stock in decision support, but it did not explicitly mention that these techniques are "data warehouses techniques،"Since 1990, the beginning of the writings of both EFCodd and WHInmon who founded and developed the modern concepts of data warehouses ،In the late eighties, early signs appeared on the data warehouse, it was by W.H.Inmon with his assistant Richard Hackathorn in a set of books ،Who coined the term Data Warehouse (Kelly, 2007) ،E. F.Codd and Ralph Kimball considered as one of the pioneers who have contributed to the development of the concept ،The idea of data warehouses spread and researchers in this field multiplied, research and scientific messages entrenchment of the concept and its uses and ways to take advantage of it to support decisions abounded, and created new techniques helped in the development of technique such as Data Mining and analytical processing on-line OLAP and SQL query language techniques In addition to the great advances in in computing and storage media, and tools for user on the desktop has been developed which made it easier to deal with the data and extract information from them in order to help in decision-making.

The modern data warehouses techniques used to support management decisions in general in various functions and levels ،In particular marketing decisions, which represents a special repository of data cloned from various operations systems after the integration in the warehouse database, which will be in isolation from operations databases ،which does not contain All operations data, but the data used to support decisions only, and is being extracted data from operations systems according to a predetermined timetable and according to the summary required by management, and the datastore data are read-only and is not update them to maintain quality of the data and decisions depending on them..

1.2.1 The Concept of Data Warehouse:

Since his appearance on the hand of WHInmon term DWH: Data Warehouse is one of the terms that took space and abundant through many of the writings and research, and has numerous definitions of the researcher to another, the others went to clarify the concept of DWH as the compilation of the data, which are derived directly from the operations systems and some external sources, objective basis to support business decisions and not the organization's operations, any use of the information collected by the organization to help them to respond better and more quickly and intelligently and effectively .(Corey and Abbey, 1996).

the concept of DWH focused on that warehouse of integrated data generated from multiple sources and are used by organization parties as a whole and is a base for the read-only used as a basis for decision support (Pokorny, 2002) ،the Base concept behind the data warehouse is to combine data from multiple databases into a single database, but they are different from the organization database on key points:

1. Keep operational systems - pre-existing - working.
2. clone usual data in the data warehouse..
3. update the data warehouse are not taken immediately after you enter the transaction, but will be updated periodically during the fixed interval..
4. The data will be read-only.

(List et al., 2000) defined the data warehouse as a common queryable source of data for analysis purposes, which is primary used as support for decision processes. It is multidimensional modeled and is used for the storage of historicized, cleansed, validated, synthesized, operative, internal and external data. Stakeholders of a data warehouse system are interested in analyzing their business processes in a comprehensive and flexible way. Mostly they already have a comprehensive understanding of their business processes, which they want to explore and analyze..

From this definition, it is clear that the data that is entered into the DWH being cleaned of redundancy and corrected if errors occur during entered in Daily transactions systems Or in the case of inconsistency Input between different information systems، and the installation of data collected and summarized, and grading is thematically or as per business

dimensions or by time (daily, weekly, monthly, quarterly, annually) (Bischoff and Alexander, 1997) Effective data means that really we need to make decisions or perform various analyzes, as being the exclusion of other data that we do not need it.

As for Ralph Kimball has provided a brief definition of the data warehouse (Kimball, 1998) :

Definition 2 (Ralph Kimball) : A data warehouse is a copy of transaction data specifically structured for query and analysis.

But the owner of the concept, and the innovative of the name William H Inmon has given a scientific definition is the most widespread of the concept of the data warehouse, as defined by (Inmon, 1992):

Definition 3 (W H Inmon): A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process.

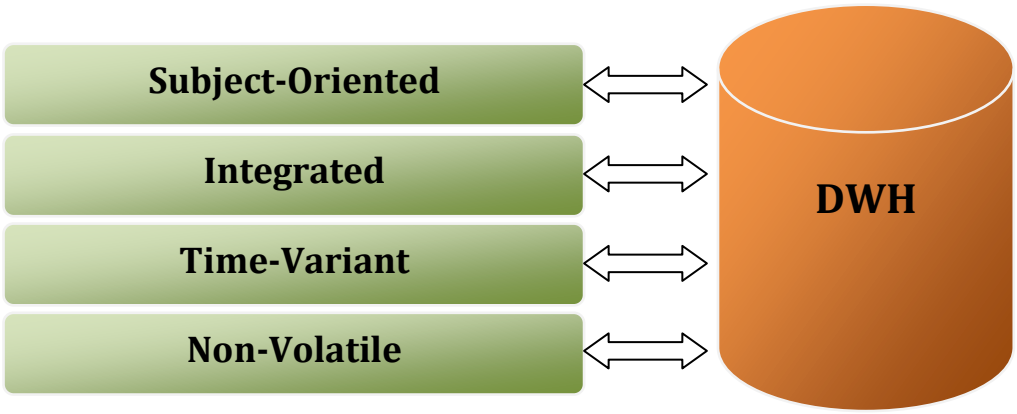

Figure 4 Characteristics of data in data warehouse
Source : (Inmon, 2000)

1.2.2 Data warehouse properties:

Based on the definition of Inmon, the data warehouse has four important properties (Inmon, 2000) (Inmon, 2005) (see Figure 4):

1.2.2.1 Subject Orientation

Data warehouse is organized by subject, this regulation allows to compile the relevant data in a particular subject, WHInmon has identified three main themes: the customer, product and policy or activity.

1.2.2.2 Integration

Data warehouse is created from a heterogeneous group of sources (relational databases and other sources ،(that we must consolidate the structure of the imported data and unification of field labels and format and coding which creates integrated between different data .This requires complex processes and consumes a lot of time for inspection and cleaning and create data.

1.2.2.3 Time Variancy

When you load the data being stamped in time, to become a specific historical data given a time, and this data is grouped on the basis of time (daily, weekly, monthly, quarterly, annual, periodic), so the time difference requires many copies of the details of the time described and the iterative storage different, and so we can summarize or more detailed data, which helps to track changes over time and contribute to the performance analysis as accurate. This property of the most important properties a data warehouse.

1.2.2.4 Non Volatile

Non-volatile means that the data is permanent and stable and unchanging, that is read-only, it cannot be modified or deleted in any way, and it being only by the addition. The data warehouse is a historic store enterprise data, where the data is saved with all their changes over time without getting rid of the previous changes. These and other important status in data warehouse, which leads to a doubling of the size of the warehouse constantly add this to the previous property..

1.2.3 Objectives of the data warehouse:

Through the data warehouse projects, organizations seeking to achieve a set of objectives vary or exceed the goals that were obtained from the operational systems and vary by organization and activity practiced, and below a set of objectives that characterize the data warehouses:

1. Provide the possibility of access to current and historical data of the Organization for analysis and decision-making through a set of tools.
2. Improving the management operations of the Organization for the rapid and reliable access to information that describes the behavior of activity, leading to improved speed and confidence in the stages of decision-making and help to identify additional opportunities for improvement.
3. Increase the speed of reporting and reduce the load on the operating system reports, because the data warehouse operations reproduces the data and put it in a separate database،

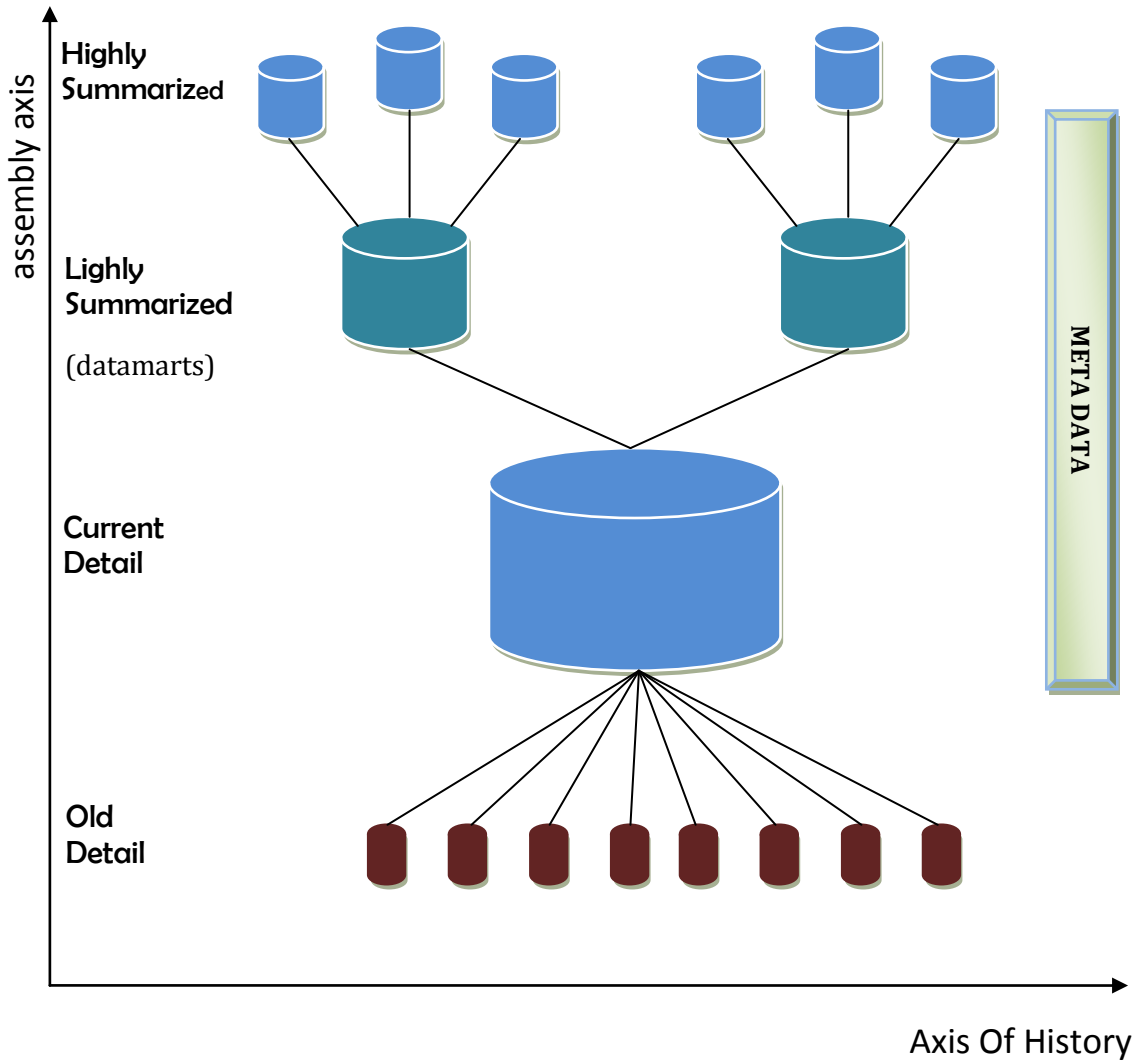

Figure 5 The Structure of the data inside the data warehouse

Source : (Inmon, 2000)

1.2.4 The Structure Of The Data Warehouse:

1.2.4.1 The Structure of the data inside the data warehouse

By (Inmon, 2000), the data inside the warehouse can be divided into several sections, and this according to degree of summarized or detailed, and have it on two axes, the assembly axis: detailed data, aggregated (summarized), and highly summarization and aggregation, and the axis of history: current data or historical (old). Moreover a data containing Description of data METADATA, so we have the following sections of the data (Figure 5):

1.2.4.1.1 Detail Data:

It is the original data have been extracted from the warehouse sources and included in it, which was not carried out any summarize or assemble in any form, and from this data is derived and extract the remaining sections summarized and dated, there are :

1.2.4.1.1.1 Current Detail Data

Which is inserted directly from the source during the update process, and there are :

1.2.4.1.1.2 Old Detail Data

In all the process of updating the data inside the data warehouse, the system does not erasing or changing existing data, but it keeps them to be added to historical development of the data, which is the most important in the data warehouse, Keeping the old data unchanged helps analysts and decision-makers to pursue changes through time ،

1.2.4.1.2 Summarized Data:

Data warehouse designed to serve the organization in different levels, and the diversity of its users, also is designed to benefit from the data for decision-making more appropriate ،but if the data is very detailed ،this is difficult and complicate the process, Which require assembly and summarize data by specific criteria ،the assembly may be by time (day, week, month, year ....) or by region (department, regional, and national .....) or by theme (sales, purchases , customers …). This is according to Requirement and request of analysts and managers and users of the warehouse, These data collected by topic and the summary that we want, kept in a small datawarehouses called DataMarts which is a specialized databases with a single subject other than the data warehouse which collects all the topics and all the data in detail, DataMarts is less expensive and much smaller than the data warehouse, Also, make queries and analyzes it be easier and faster, and the data collected may be Lightly Summarized Data (day or week, department or area) or Highly Summarized Data (a month or a year, regional or national).

1.2.4.1.3 Meta Data

The last component of the data warehouse is the Meta Data, the word Meta-data consists of two syllables: the first (*Meta*) is a Greek word meaning "beside, with, after, following". but In Latin Modern and English current, the word is used to signify the beyond, Or indicate something located behind the scope of human experience **Transcendental**, The second syllable, (*Data*) means data, and then we can say that the metadata is data about data, Or It is a data containing a description and a summary of the data in the warehouse, They are not included in the extracted data from sources and not inferred or collected data of them, MetaData contains all the information about the data warehouse and the data itself, which represents a dictionary of data features easy access to it, which is very important as it contains a description of the structure of the data warehouse, and the algorithms summary of current detailed data and Low or highly summary... and other.

1.2.5 On-Line Analytical Processing OLAP

This term appears the first time in a research paper presented by EFCodd - innovative relational databases (Codd et al., 1993) in 1993, entitled (Providing on-line analytical processing to user analysts), Which He explained the 12 features or rules for on-line analytical processing OLAP, These rules distinguish between simple queries to databases, reporting tools and the most advanced analytical methods, The (OLAP Council), which aims to promote the use of OLAP, has issued a definition of it :

[**On-Line Analytical Processing (OLAP):** is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user]

OLAP provides capabilities of the direct analysis of the data required to answer the questions of decision-makers,

1.2.5.1 Codd's 12 Rules for OLAP:

EFCodd has put in his paper about the term OLAP 12 rules to determine the concept of the term, are:

Rule 1: ***Multidimensional conceptual view*** : User-analysts would view an enterprise as being multidimensional in nature – for example, profits could be viewed by region, product, time period, or scenario (such as actual, budget, or forecast). Multi-dimensional data models enable more straightforward and intuitive manipulation of data by users, including "slicing and dicing".

Rule 2: ***Transparency*** : When OLAP forms part of the users' customary spreadsheet or graphics package, this should be transparent to the user. OLAP should be part of an open systems architecture which can be embedded in any place desired by the user without adversely affecting the functionality of the host tool. The user should not be exposed to the source of the data supplied to the OLAP tool, which may be homogeneous or heterogeneous.

Rule 3: ***Accessibility*** : The OLAP tool should be capable of applying its own logical structure to access heterogeneous sources of data and perform any conversions necessary to present a coherent view to the user. The tool (and not the user) should be concerned with where the physical data comes from.

Rule 4: ***Consistent reporting performance*** : Performance of the OLAP tool should not suffer significantly as the number of dimensions is increased.

Rule 5: ***Client/server architecture*** : The server component of OLAP tools should be sufficiently intelligent that the various clients can be attached with minimum effort. The server should be capable of mapping and consolidating data between disparate databases.

Rule 6: ***Generic Dimensionality*** : Every data dimension should be equivalent in its structure and operational capabilities.

Rule 7: ***Dynamic sparse matrix handling*** : The OLAP server's physical structure should have optimal sparse matrix handling.

Rule 8: ***Multi-user support*** : OLAP tools must provide concurrent retrieval and update access, integrity and security.

Rule 9: ***Unrestricted cross-dimensional operations*** : Computational facilities must allow calculation and data manipulation across any number of data dimensions, and must not restrict any relationship between data cells.

Rule 10: ***Intuitive data manipulation*** : Data manipulation inherent in the consolidation path, such as drilling down or zooming out, should be accomplished via direct action on the analytical model's cells, and not require use of a menu or multiple trips across the user interface.

Rule 11: ***Flexible reporting*** : Reporting facilities should present information in any way the user wants to view it.

Rule 12: ***Unlimited Dimensions and aggregation levels*** : The number of data dimensions supported should, to all intents and purposes, be unlimited. Each generic dimensions should enable an essentially unlimited number of user-defined aggregation levels within any given consolidation path.

Nigel Pendse - founder of OLAP Report - summed the definition of OLAP in five words Gathered in the term FASMI and detailed in the phrase (Pendse, 2003) **:** (F**ast** A**nalysis of** S**hared** M**ultidimensional** I**nformation) ,** This definition is consistent with the standards to simplify the Codd's rules and facilitates the process of evaluating OLAP tools**.**

In Table4 ,we explain the differences between data warehouses properties and operational databases

Properties	Operational databases (OLTP)	Data warehouse (OLAP)
Objectives	Implement an operational process Management and production	Evaluate the operational processes Preview and analysis
Usage	Support the current administration	Decision Support
Design principle	3FN	Multi-dimensional
Data	Current, raw, Data management, Without repetition	Historical, aggregated, Data Analysis, Often with repetition
Data organization	By processing	By subject
Queries	Simple, predefined, detailed data	Complex, private, aggregates ʻgroup by
Operations	Read and write	Read and Add Without delete
Users	Employees, production manager, all A large number of users	Analysts, decision-makers A small number of users
Size	Medium Up to some giga-bytes	Huge Up to some téra- bytes

Table4 data warehouses properties versus operational databases

1.2.6 Basic Concepts

1.2.6.1 Dimensional Modeling:

Dimensional model is most used data structures and the most appropriate inquiries and analysis of users of the data warehouse. They are simple construction, stable and intuitively understandable to the end users. Dimensional model is the real basis for building OLAP cubes. The first to introduce a multi-dimensional model is Ralph Kimball, The model is based on two basic concepts: Fact Table and Dimension Table, We have in the dimensional modeling two main schemas: Star Schema, and Snowflake schema, there are some basic concepts in dimensional modeling of which :

1.2.6.1.1 Dimension :

A category of information. For example, the time dimension.

1.2.6.1.2 Attribute :

A unique level within the dimension. Month is an attribute in the dimension of time.

1.2.6.1.3 Hierarchy :

The specification of levels that represents relationship between different attributes within a dimension. For example, one possible hierarchy in the Time dimension is Year → Quarter → Month → Day...

For the success of dimensional model must perform the following four steps(Kimball and Ross, 2002):

1. Select the business process to model and studied..
2. Identify the degree of precision Granularity and amount of detail of the process in the facts table..
3. Choose the dimensions that apply to each fact table row..
4. Identify the numeric facts that will populate each fact table row.

1.2.6.2 Fact table :

Fact tables are the basis for the data warehouse .They contain the basic measurements of the project, and they are the ultimate target of most data warehouse queries, the Fact table is the primary table in a multi-dimensional model, and it is a table that contains the observable data (facts) that we have on a subject and we want to study, using various analytical axes (dimensions), Thus, the fact table consists of two types of columns. The foreign keys column allows joins with dimension tables, and the measures columns contain the data that is being analyzed.

Generally, the fact table contains a few columns, and the number of records is much more than the dimension table. The Information in the table is characterized by that it is often used for digital aggregates SUM, AVG ..., Data must be additive or semi additive, it's also not without repetition, because of the different time recordings for the preservation of the history of activities.

Figure 6 Fact table model

Source :(Kimball and Ross, 2002)

1.2.6.3 Dimension table :

Dimension table is a table that represents the axis of axles analysis (product, customer, date, region, country ..... etc.), and contrary to the fact table, the dimension contains descriptive attributes which are usually text fields. And the dimension attributes must be :

- lengthy with many of redundancy (labels consist of whole words).
- Descriptive.
- full (No missing values)
- valued separately (and there is only one value for each row).
- guaranteed quality (no misspellings or impossible values).

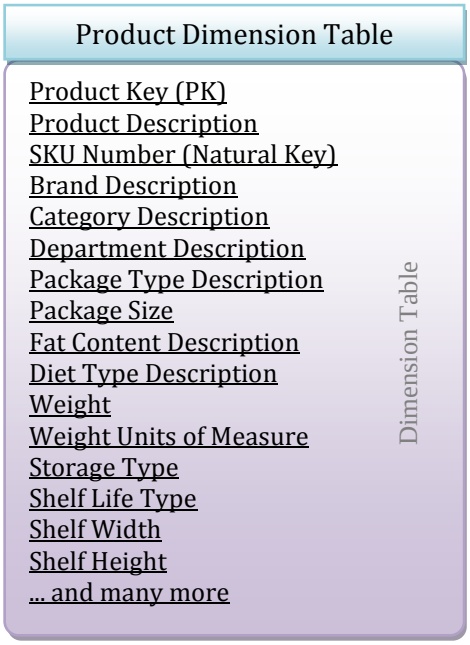

Figure 7 Dimension table model

Source :(Kimball and Ross, 2002)

1.2.7 Data Warehousing Schemas:

There are three basic types of dimensional model are the :

1.2.7.1 Star Schema

It consists of a large central table is a (facts table) and a circle of other tables that contain descriptive elements of fact, is the dimension tables, Each dimension is represented as a single table. all tables are de-normalize, It represents in form of a star from which came the name (Figure 8).

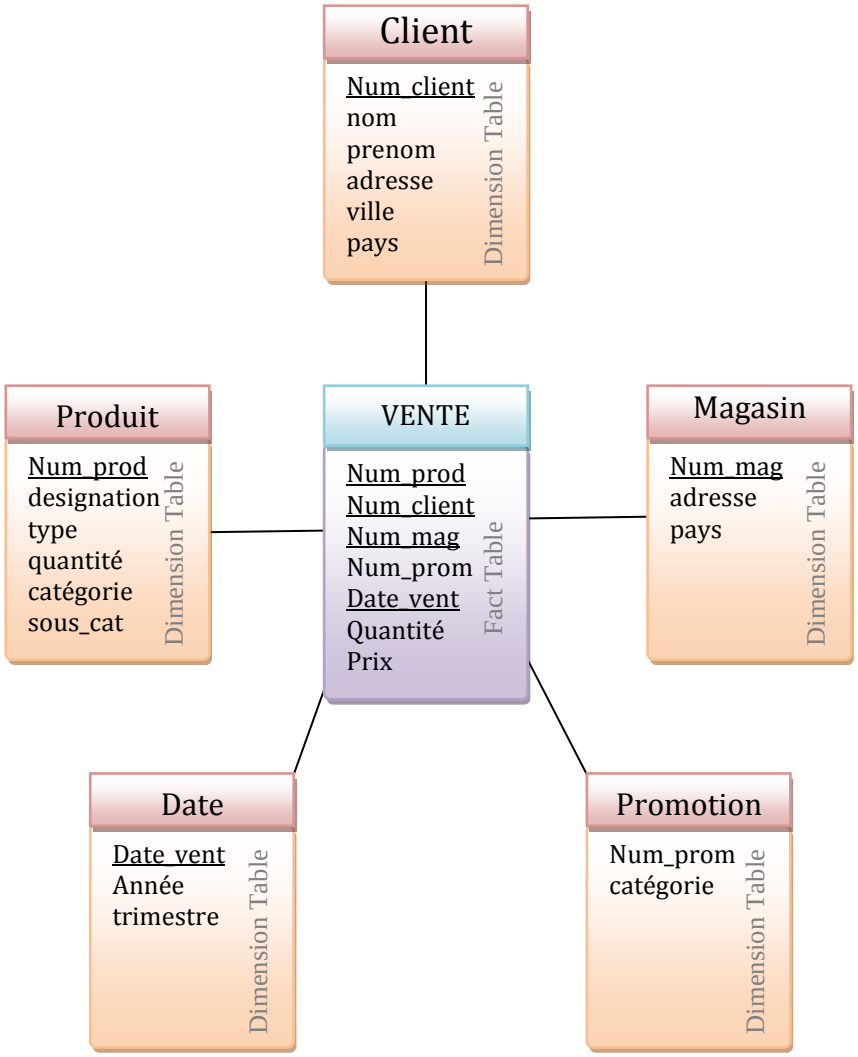

Figure 8 Star Schema

1.2.7.2 Snowflake schema

It Is similar to the previous Star Schema, but the main difference is that the dimension tables in Snowflake schema, all be unified format with data normalize resulting in an additional dimension tables associated with others that applied to it the rules of normalization, Representing the hierarchy to him (Figure 9).

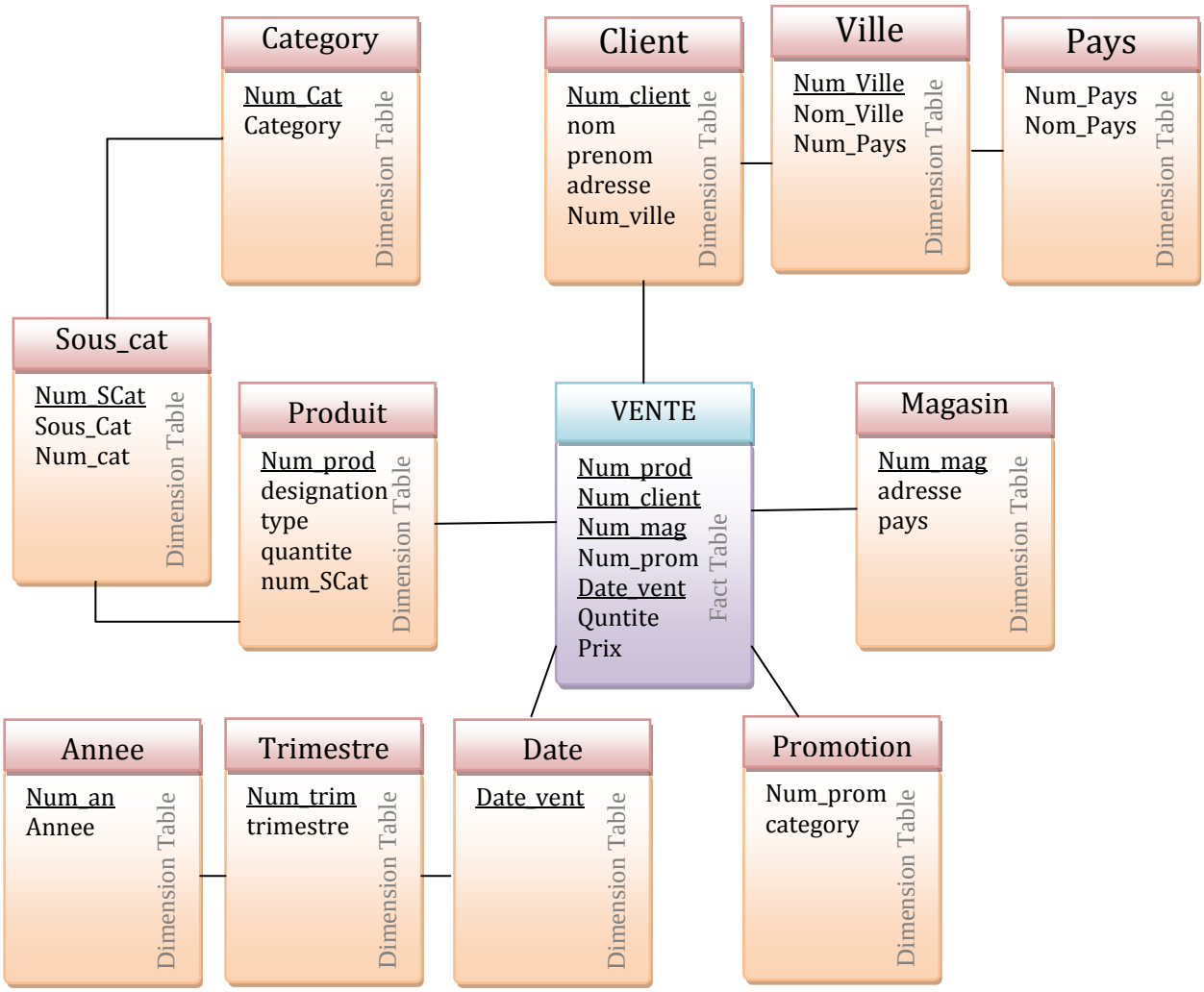

Figure 9 Snowflake schema

1.2.7.3 Constellation schema

is a collection of stellar schemes associated with each other and shared in some of the dimension tables (Figure 10).

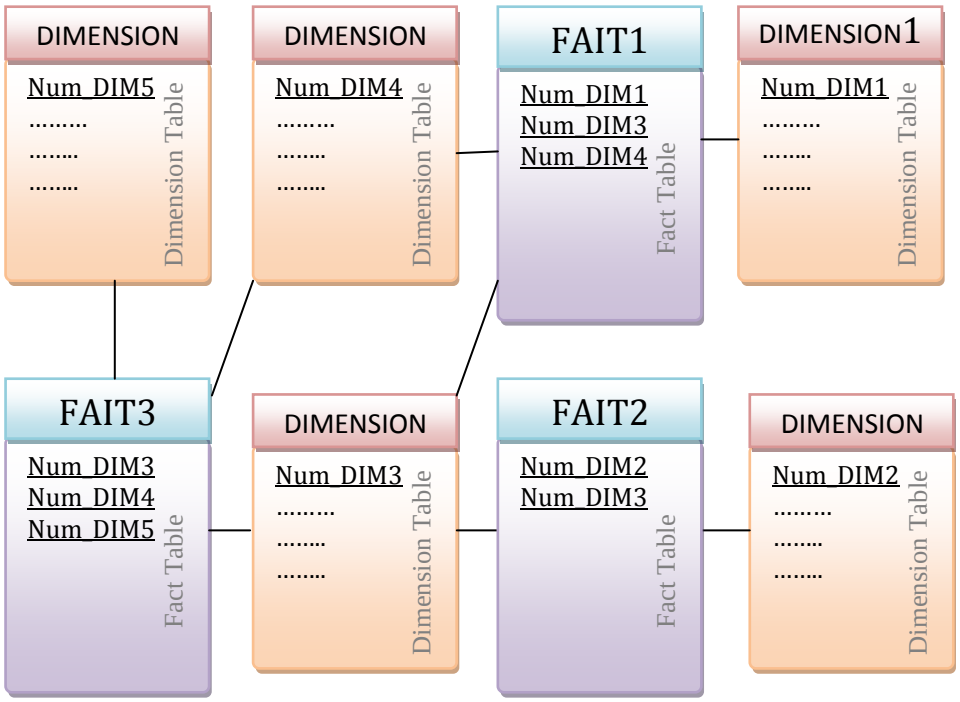

Figure 10 Constellation schema

1.2.8 Data Cube

Multi-dimensional model based on the concept of the data Cube Or hypercube or multi-dimensional table, The cube data provides an abstract idea very close to the viewpoint of analysts of the data and queries around them, Inside the cube, the data is organized in several dimensions, each dimension represents an axis of the analysis axes, **Dimension** consists of a set of **attributes**, Each attribute may take multiple values. The dimensions usually associated with **Hierarchy**, which organizes attributes at different levels to note the data in detail and various gradients (**Granularity**), for example, we have a multi-dimensional table preserve the values of sales according to three dimensions:

Location: it may has the hierarchy:

Country → region → Wilaya → Commune.

Product: it may has the hierarchy:

superfamily → gender → Product → Type

Time: it may has the hierarchy:

Year → season → month → week → Day

The table can be represented in the form of a three-dimensional cube, as shown in Figure 11:

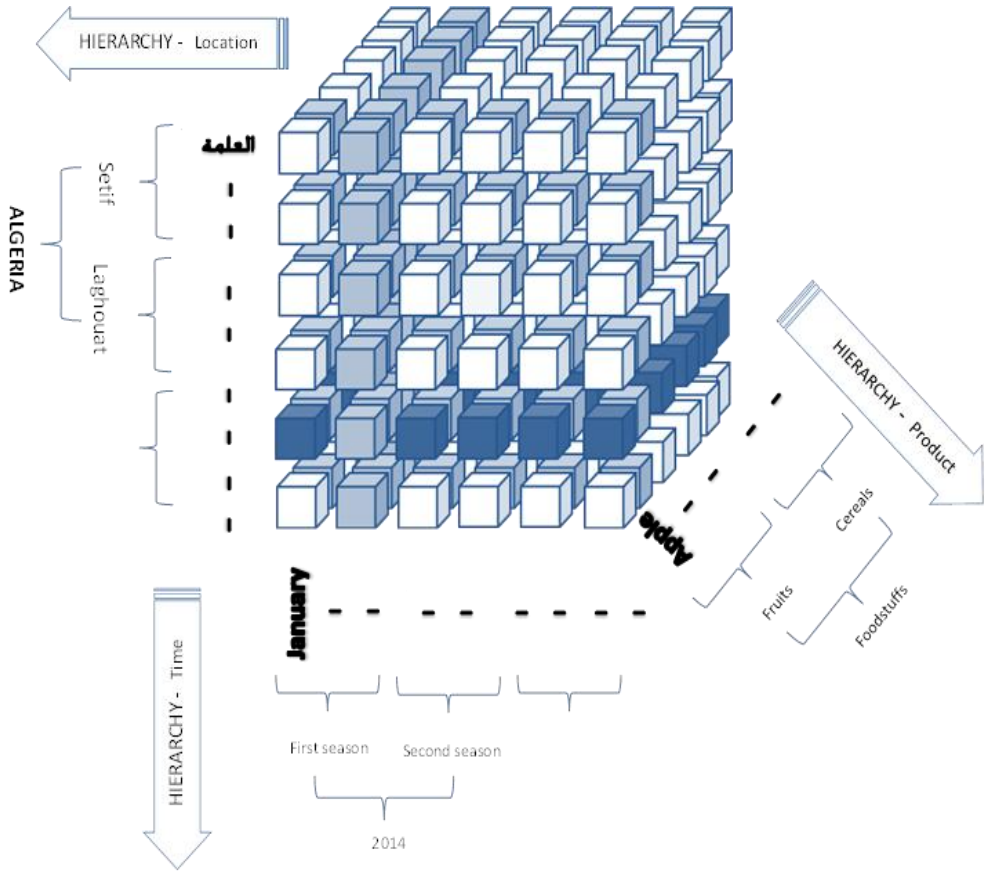

Figure 11 Data Cube

Can be many processes applied to the data cube, as can the cube rotate according to the axis of the axes, can also be truncated a part of the cube or delve deeper into detail or vice versa, and can be divided into these operations into two sections: section linked to structure (Figure 12). The second is linked to a degree accuracy and detail or granularity (Figure 13).

1.2.8.1 Operations related to the Structure:

- ***Rotate/Pivot***: two-dimensional is selected from the dimensions of the cube to show the aggregate data on them, for example, show oil sales in the month of May (with all areas), and

after rotate the axis show oil sales in Algeria (with all time). Rotate determines to us the flip side of the cube, and does not specify the box or one element of it

- ***Slicing*** : that is to work on one slice of the cube, where one-dimensional shorten in only one value..
- ***Dicing*** : Extracting data block And work on a sub-cube of the original cube.

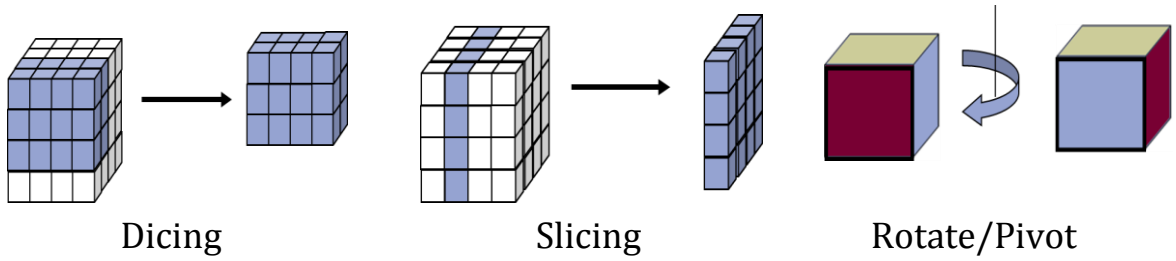

Figure 12 Most important processes related to the structure of the Cube

1.2.8.2 Operations related to the Granularity:

Roll-Up : to get the lowest level of detail, and increase the level of aggregation using aggregate functions

Drill-Down : increase the level of detail, so we have more detailed data,‹

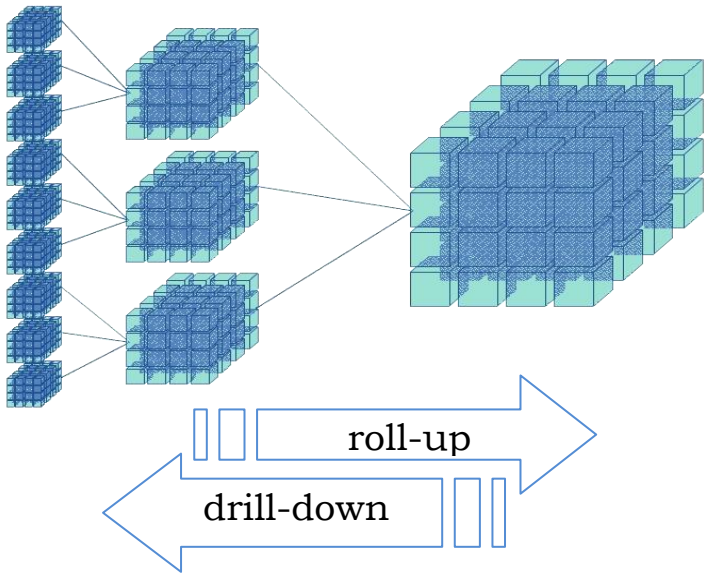

Figure 13 Most important processes related to the Granularity

1.2.9 OLAP storage models

OLAP model can be stored in various ways, the basis of these is Two main models Are *MOLAP* and *ROLAP*, we find the following models:

1.2.9.1 MOLAP:

Which is the true embodiment of the multidimensional modeling, Naming MOLAP is an abbreviation for the meaning of the multidimensional model which Multidimensional OLAP, It is the physical application of the concept of OLAP, a data cube is used instead of relational tables, and these systems are characterized for relational ROLAP speed and better performance, due to the possession of the most sophisticated tools than its predecessor. However, the problem remains at the level of the size that remains limited.

1.2.9.2 ROLAP:

Relational OLAP is in fact a traditional relational database designed to work as an OLAP database. ROLAP rules will be much slower and less efficient than the rules MOLAP. But the big advantage, it is not limited in size.

1.2.9.3 HOLAP:

The third model, is HOLAP model which Hybride OLAP, It combines the advantages of both previous models. The aggregated data are stored in the multidimensional structure, while the detailed data is stored in relational structures.

1.2.9.4 DOLAP:

It is a model characterized by the small size, naming DOLAP means Desktop OLAP because most operations occur on the desktop of the client, as well as it means Dynamic OLAP because cubes created immediately upon request and not previously, such as previous models.

1.3 Conclusion :

Thus we have finished the first chapter, which was talking within about DSS, Which we have defined as an interactive information system that works to collect, transfer, analysis and processing data in order to facilitate the decision-making process (especially unstructured and semi-structured decision) or in order to obtain knowledge. One of the main components of this system is a huge database that gathering all the system data in all its details addition to aggregates and summaries carried out, this huge database, which is called a data warehouse was the focus of the second section of this chapter, as we talked about the data warehouse, which we explained that the data entered is being cleaned of the iteration and corrected if errors occur when you enter it in daily transactions, or in the case of inconsistencies between different information systems entry, As the data are installed or assembled or summarized or grading on an objective basis. And we dealt with the history of warehouses and definitions multiple to him, and types of data within it: historical data and current and detailed and aggregated metadata, and the data warehouse models and schemas and OLAP is applied to it, From among schemes There are Star Schema, Snowflake schema, constellation schema, as we have seen data Cube, and dimensional modeling.

through this chapter Show us the multiple data warehouses features, Show us through this chapter the multiple data warehouses features, the most important of which the huge size And the immense and the diverse quantity Of data on it, which raises the question over the flexibility and responsiveness of queries that pose within it in order to extract the data, and that is what we will try to answer it in the next chapters of the memo.

CHAPTER 2

XML Technology

and Complex Data Warehouse

Summary:

In this chapter, we will work to provide various basic concepts related to XML technology. As we display the foundations of XML data basesas well as the complex data warehouses.

2) Chapter 2: XML and complex data warehouse

2.1 Section I :XML Technology:

2.1.1 Complex Data :

The emergence of databases and then Data warehouse made things very easy for decision makers and companies' managers, but these data is not just a text, it could be voice, images, videos or other types of data. Then we will other type of data, not simple and not less important than those that can be assembled in text tables of different formats. For example, a patient's medical history might be recorded as plain text. Various biological exam results might be indicated in many ways. The file could also include radiographies (images) or echographies (video sequences). We can call this data as **Complex Data**.

In fact, there is no agreed-upon definition of the term complex data, But in)Darmont et al., 2007(authors define complex data, according to them we can consider data complex if it takes one of these forms:

1. **Multiformats**: data may takes different forms and formats (digital data, symbols, text, images, voice, video)
2. **Multistructures**: data could be structured or unstructured or semi-structured(relational databases , binary files, XML documents)
3. **Multisource**: the data is created from different sources(Distributed Databases, Internet)
4. **Multimodal:** The phenomenon is described through several channels, point of views (X-ray and voice diagnosis for a doctor to assess the health status of the patient, Data are expressed in different scales or languages).
5. **Multiversion:** Where the data is scalable in terms of definition or values (temporal databases, periodical surveys...

According to (Darmont et al., 2007), this definition is not sufficient to cover the wide variety of complex data, It could indeed be viewed as an axis of complexity, among other axes dealing with data semantics, processing, syntax. As shown in Figure 14.

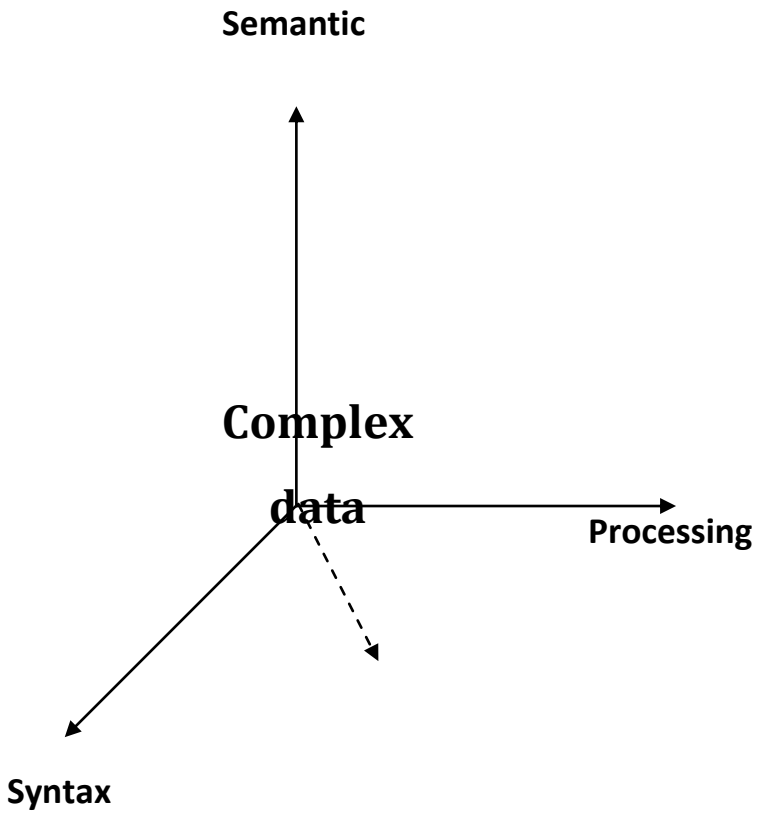

Figure 14 Complex data axes

2.1.2 XML History: Origin And Evolution

XML, (Extensible Markup Language) is markup language which can be defined as languages designed for structuring and coding texts to make them easily handled and processed automatically.

Markup languages were designed first to work with specific programs, but then in 1986, they have been standardized as **SGML**: Standard Generalized Markup Language. And then **HTML**: Hyper Text Markup Language was inspired from SGML which is used to create web pages. In 1998, the W3C recommend using the widget from SGML and later known as the e**X**tensible Markup language XML.

Most markup languages are different from the databases in their ability to identify the elements contained within the text without discrimination of the structured data. However, the XML language has the ability to convert any text to equal a database.

From the above definition, it is clear that the markup languages task is focused in the coding information and formulation of unique structured data easily manageable by all systems and applications, because they focus on the form of information and the procedures that will be applied on that data.

If we take look at the history of markup languages, we will find that the first one was in September 1967. During a meeting at Canadian Government Publications Office. William Tunnicliffe presented a new idea on separation data content from its form, This agrees with other writings in this area and the trend began to separate form from content.

In 1969 **GenCode** project appeared in which **Charles Goldfarb** had worked that was sponsored by IBM and the result of that project was Generalized Markup Language (GML) and after years of work. And After several years of work, the SGML emerged into the light to overcome the GML shortcomings.

In the 1980s, the SGML has spread in most companies and government institutions in the world; it was one of the first languages for encoding and structuring of hyper texts. In 1986, ISO (International Organization for Standardization) approved the SGML and published ISO 8879.

And it did not spread in its early years among users because of its difficulty and complexity as they are mainly designed for the development and deployment of multimedia and its representation to make it easy to work with, so it is a very useful in creating complex and composite pages that combine more than one type and format of information sources. And because of this complexity and difficulty in the language, there was a need for an easier and a simpler language especially to serve the not specialized users who want to create simple pages. And for that, HTML has spread very quickly among users for the ease of using through a variety of text editing programs and tools like Microsoft Word**,** HTML relies on using TAGS to restructure the text in the form of paragraphs, lists and hyper links.

These languages appeared to deal with the so-called Hyper Text, which has become a style that characterizes the published texts on the Internet. Using these languages, items and objects within the document are linked, with easy transition between these elements, either by ascending or descending linking.

With the appearance of Internet and its applications, this method in the structuring and publishing of texts has spread. Especially in the World Wide Web, when Tim Burns Lee, one the Directors of the W3C, created the first protocol of the hyper text that works on the Internet and this is why the invention of the Web is attributed to him.

But the idea of hypertext and text's linking and other media and information sources rooted in the forties of the last century and particularly to July 1945,When Vannevar Bush, who was working at the time as director of the US Office of Scientific Research and Development, published his famous article "As We May Think" (Bush , 1945) in which he described his imaginations and his vision of hyper texts and dubbed it at the time the name "**MEMEX**". He suggested the linking between information with the possibility and ease of modification and adding other user's private information, these ideas became reality with the World Wide Web, which has become heavily dependent on the appearance of hypertext protocol HTTP.

However, with this deployment and continuous development that HTML found, which it was subject until the appearance the latest version with HTML 4.0 in 1997. It has been found that they represent an obstacle to publish multimedia because It works very efficiently with text items. But there is an urgent need to deal with all kinds of media. So DHTML or Dynamic HTML was developed and issued to become more dynamic to deal and publish with multimedia on the Internet.

Previous markup languages were based on tags which they were to facilitate to computers to deal with texts inside tags whether by showing it on the screen, printing, or modification but these tags do not describe efficiently the document itself. For example, it cannot provide description of the word in the document or add its meaning what is its signification. The problem stood out more when documents are retrieved in which many people attributed the search engines failure to provide good results to the problem of form and method of structuring documents

So XML appeared to solve this problem, where we can add a description of any paragraph or word within the document, the document then is transformed into a huge database that we can search and share it with various other applications.Add to that, it also came as a meeting point between my SGML and HTML, it facilitates the exchange of data between sources that are written in both SGML and HTML, it also a s indicates Online Dictionary for Library and Information Science:

"A subset of the SGML markup language in which the tags define the kind of information contained in a data element (i.e., product number, price, etc.), rather than how it is displayed. "Extensible" means that XML tags are not limited and predefined as they are in HTML--they must be created and defined through document analysis by the person producing the electronic document. Designed to meet the needs of large-scale electronic publishing, XML is a flexible text

format that can be used with HTML in the same Web page. Document structure can be defined in a Document Type Definition (DTD) or XML Schema capable of handling document hierarchies."

World Wide Web Consortium (W3C) the organization responsible for the development of XML is the same for the development for the previous languages which is composed of more than 400 member from commercial, academic and government agencies from all over the world. Its activity is concentrated on the development of protocols and standards that will be used on the Web. It released XML in 1996 and it was finally adopted by the W3C in February 1998, which was the year of its deployment on the Internet.

As indicated in W3C website, it took into consideration to achieve several objectives when they have designed this language:

- Legalization of deployment of electronic resources in an independent and a common way.
- Ease of E-commerce data exchange.
- Delivery of information to local programs to be easily processed automatically and deals with it the moment it received.
- Ease of processing and exchange of information and data across different applications and systems at a lower cost.
- Ease of deployment and support the data description (Metadata) for each document which Makes it easier to be retrieved and discovered and then to find a link between a product or an information maker and the final beneficiary.

XML is considered as the second son SGML where it works as a copy it. Unlike its predecessor HTML which was facing many problems regarding a non-precision and sufficient control to describe data. HTML is not interested in the order of the code for the data, which makes it difficult for the program in the browser to conclude the meaning of data, its structure and its order as intended, which makes strongly possible to have multiple translations and interpretations of data by browsers. Especially with complex pages that contain more than one form of information. XML came to eliminate this problem. It is interested in what is the data and what is its purpose. It does not leave the program or user to guess or to form the data. It is a strict language. So it does that by itself and it cares in every detail where the data processing is done inside the document through XML programs and applications.

2.1.3 Rules and basics of the XML

```
<?xml version="1.0"?>
<vente>
<articles>
<article code="1">
<libelle>clavier</libelle>
</article>
<article code="2">
<libelle/>
</article>
</articles>
</vente>
```

- XML documents begin with a line of definitions to determine the version, the used character encoding encoding and the independence of the document (standalone):

<?xml version="..." encoding="..." standalone="..."?>

- XML documents form a tree structure that contains set of elements which starts at the main element "the root".

- Each element begins with a start-tag and ends with an end-tag **<element1> ... .. </ element1>**, the empty element can be represented also with a start-tag that ends with symbol "/": **<element1 />**. Unlike HTML, Start-tags and end-tags should not overlap and between elements.

- XML tags are case sensitive, <Prenom>, <prenom> and <PRENOM> are three different elements from each other. Therefore, Start-tags end end-tags must be written with the same case.

- XML element cannot start with digits, the letters xml (or XML, or Xml, etc) and also it cannot contain spaces.

- XML elements can have attributes, an attribute must have value placed between quotation marks, double or single: <font color = "Red"> ..... </ font>.

- XML document can contain comments as follows:

<!-- This is a comment -->

- It must respect some special symbols in XML and not use them within the language because they generate errors. These characters are : >, <, & , ‘ ," that need to be replaced in XML by other symbols as follows (Table5):

-

Symbol	&	>	<	‘	"
Alternative	& amp;	& lt;	& gt;	'	&quꞌ

Table5 Special characters in XML

For Example: This line is wrong **<message> if salary <1000 then </ message>** and the right is: <message> if salary & lt; **1000 then </ message>.**

- CDATA sections can be used, but the text inside CDATA section will be ignored by the parser without any interpretation.

```
<script language='javaScript'><![CDATA[
    function myFunc()
    {
        if(0 < 1 && 1 < 2)
            alert("Hello");
    }
]]></script>
```

Figure 15 Model shows how to use CDATA sections in an XML document

- XML document in which all the above conditions are applied is Well Formed document.

2.1.4 XML Parser:

The existence of these strict rules to write a well-formed xml documents, is for making the process of information extraction easy. We call the language processor **XML Parser**. Because, it simply analyzes the XML document and passes structured information to an application. Most modern browsers web have implicitly an xml parser.

As XML parser was determined to deal with the information within an XML document, XML parser was also dealing with errors, There are two types of errors, error and fatal errors. An error is a violation of the rules of the specification where the results are undefined. A fatal error, When encountering a fatal error, the XML parser must not continue normal processing and must report such error to the application. That means that any error makes the document not well-formed will considered as a fatal error. Such strict in the writing of XML documents increase the strength, not only to avoid errors parsers but aims to unify the write of a standard documents can

be handled with more than a browser and more than operating platform, Unlike HTML, as there is no stringent in the writing of the texts, which you may find a browser displays the page inappropriately while another browser displays it as you like.

There are several XML Parser, it can be classified according to the approach followed in the Parser into two categories:

2.1.4.1 Tree-based XML Parser

Hierarchical approach Or Object-based, It analyzes the whole document and constructs hierarchical tree where all elements can be accessed through the tree. DOM (Document Object Model) represents the main API for using that approach.

2.1.4.2 Event-based Parser

The parser responds to events (the beginning of an element, the end of the element, the content ....) and returns the result of the application. The main API for using that approach is SAX (Simple API for XML). There are two types of Event-Based Parsers Push Parsing and pull parsing (Figure 16). In Push Parsing, the parser processes a document's contents sequentially and generates events such StartDocument() and endDocument (). This approach provides no help for programmers to access the entire document. SAX is push parsing approach.

In pull parsing, Events are created when we recall some API, the programmer can decide when to generate events, the client gets the data (pull) only when explicitly requests it, Streaming API for XML (StAX) example of cloud analysis, with analysis or express the client controls the cloud the subject of the application, and can call methods on the parser when needed. By contrast, with batch processing, the parser is in control of the subject of the request, Where events are created when we recall some of the API, the programmer can decide when to generate events. The client gets the data (pull) only when explicitly requests it, Streaming API for XML (StAX) example of pull parsing, With pull parsing, the client controls the subject of the application, and can call methods on the parser when needed. By contrast, with Push Parsing, the parser controls the subject of the request,.

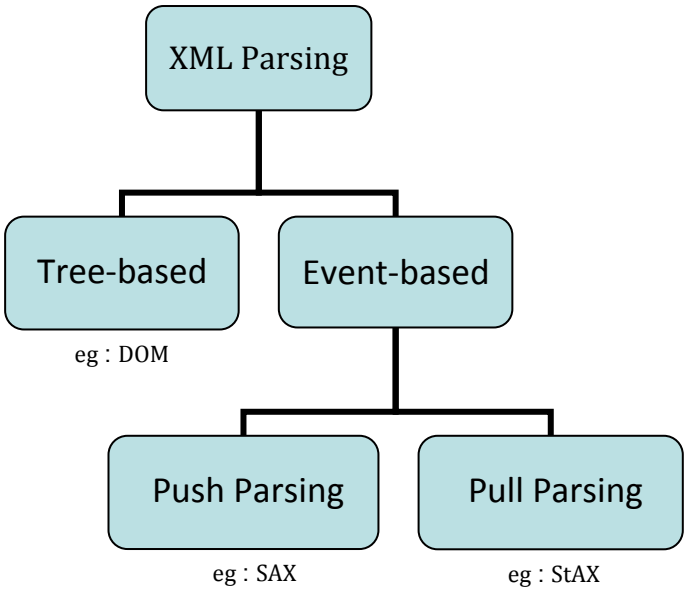

Figure 16 XML parsers and types

2.1.4.3 DOM (Document Object Model) :

DOM was recommended by W3C in late 1998, the parser loads the entire document in memory and constructs hierarchical tree (Figure 17), where the access to all the elements will be direct and easy. Through the DOM tree, we can access all the elements, modify the contents or even deleted them; we can also create new items where these items and properties are treated as nodes. When it must to deal with XML data together and restructure it, it is best to use DOM parsers to take advantage of its flexibility. But for big documents where its size is close to the memory on the machine, DOM becomes insufficient (The use of DOM becomes slow).

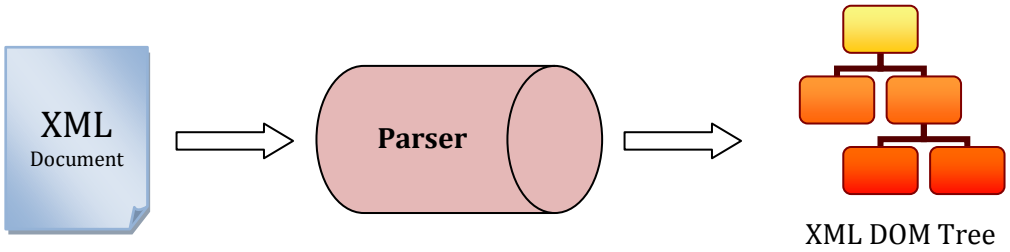

Figure 17 Tree-based Parser model (DOM)

2.1.4.4 SAX (Simple API for XML):

It is a way to deal with XML files, It works differently from the way DOM works. It does not load any XML document into memory. The SAX processor reads the XML file contents consecutively (Sequential access) and it Calls events for each Opened and closed tag, and for any other element in the XML file. This means that we can access any Tag we want using this parser, because SAX will track all tags in the XML file (Figure 18).SAX API is well-suited for processing XML files that does not require to pass the document and more than one time. Or in the case of large amounts of data, so It is not necessary to represent the entire data on the memory

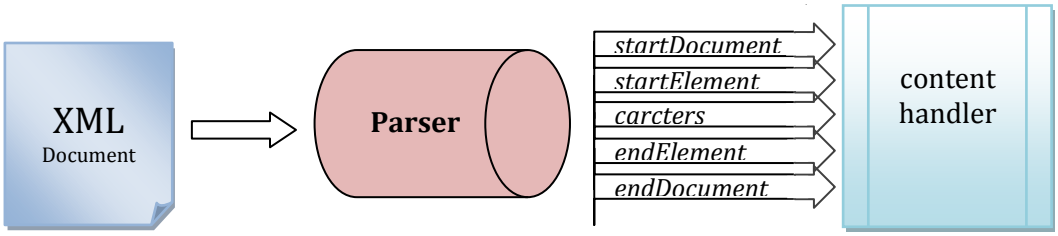

Figure 18 Pull Parsing Event-based Model (SAX)

2.1.5 Valid XML Documents

So the XML document to be valid. First, the body needs to be Well-Formed. Secondly, it should respect the constraints that identified its structure (those related to elements and attributes). These documents must be written, presented and treated depending on the determined structure in Schema. That contains the definition of the structure of these documents.

There are three main types of Schemas, namely:

- **DTD** (Document Type Definition)
- **XSD** (XML Schema Definition)
- **RELAX NG**(REgular LAnguage for XML Next Generation)

2.1.5.1 DTD (Document Type Definition)

The purpose of DTD (Document Type Definition) is to define the document structure with a list of legal elements and attributes. There are many advantages of DTD, including that each XML file can carry its own description. The independent groups of persons may agree to use a common DTD to exchange data In your application, you can use a standard and uniform DTD to make sure that the data you receive from the external world is correct. It can also use DTD to check your data. And there two types of DTD: internal and external. An internal, where declarations commands and conditions of a DTD is built in the XML and the same document and an external, External declaration are located in a separate .dtd file, And it is called within the XML document (Figure 19).

```
<?xml version="1.0"?>

1. <!DOCTYPE vente[
2. <!ELEMENT vente (articles)>
3. <!ELEMENT articles ( article+)>
4. <!ELEMENT article (code,libelle)>
5. <!ELEMENT code (#PCDATA)>
6. <!ELEMENT libelle (#PCDATA)>
7. ]>

<vente>
<articles>
<article>
<code>1</code>
<libelle>clavier</libelle>
</article>
<article>
<code>2</code>
<libelle>Souris</libelle>
</article>
</articles>
```

<!DOCTYPE vente SYSTEM "fichier.dtd">
To call an external DTD

Figure 19 Example of a file DTD

To understand the previous example we need to explain some of the commands in the DTD (see Table6).

<!DOCTYPE	To start the DTD and be attached to the name of the root
!ELEMENT	To specify the child elements of the element
!ATTLIST	To specify the attributes of the element.
؟ * +	To determine the number of repetitions of the child element. (+: more than once, *: zero or more,?: Zero or one(
#PCDATA	The element contains text data are processed by parser
#CDATA	element contains text data exceeded and skipped by parser
EMPTY	Empty element does not contain any thing
ANY	element contains anything you want

Table 6 Some commands DTD

2.1.5.2 XSD (XML Schema Definition)

Due to some criticisms of the DTD, and that was the result of the shortcomings this technology has been unable to fulfill them.

- The DTD is written in XML, Which may result in difficulties in analyzing the document Using XML parsers such as DOM or SAX.
- The DTD does not support Namespaces.
- The number of occurrences of an element cannot be determined precisely, but only using determinants **(+ * ?)**. For example, we cannot set the number of occurrences of an item more than 3 times less than 10 times. This restriction is not possible in DTD.
- The DTD is unable to specify element types, it can only determine if the element contains data or not without specifying its type (Numeric value, integer, decimal number, text or date etc ...).

To avoid these shortcomings, W3C published in May 2nd 2001 **XML Schema**, a recommendation known as **XSD** (XML Schema Definition) that defines the structure of a XML document scheme according to strict conditions and restrictions. Using **XML Schema** (It is written in XML), we can select and define the types of variables and define classes for elements and attributes (Figure 20). The restriction is controlled to repeat elements in terms of the minimum and maximum number of occurrences, it also supports namespaces.

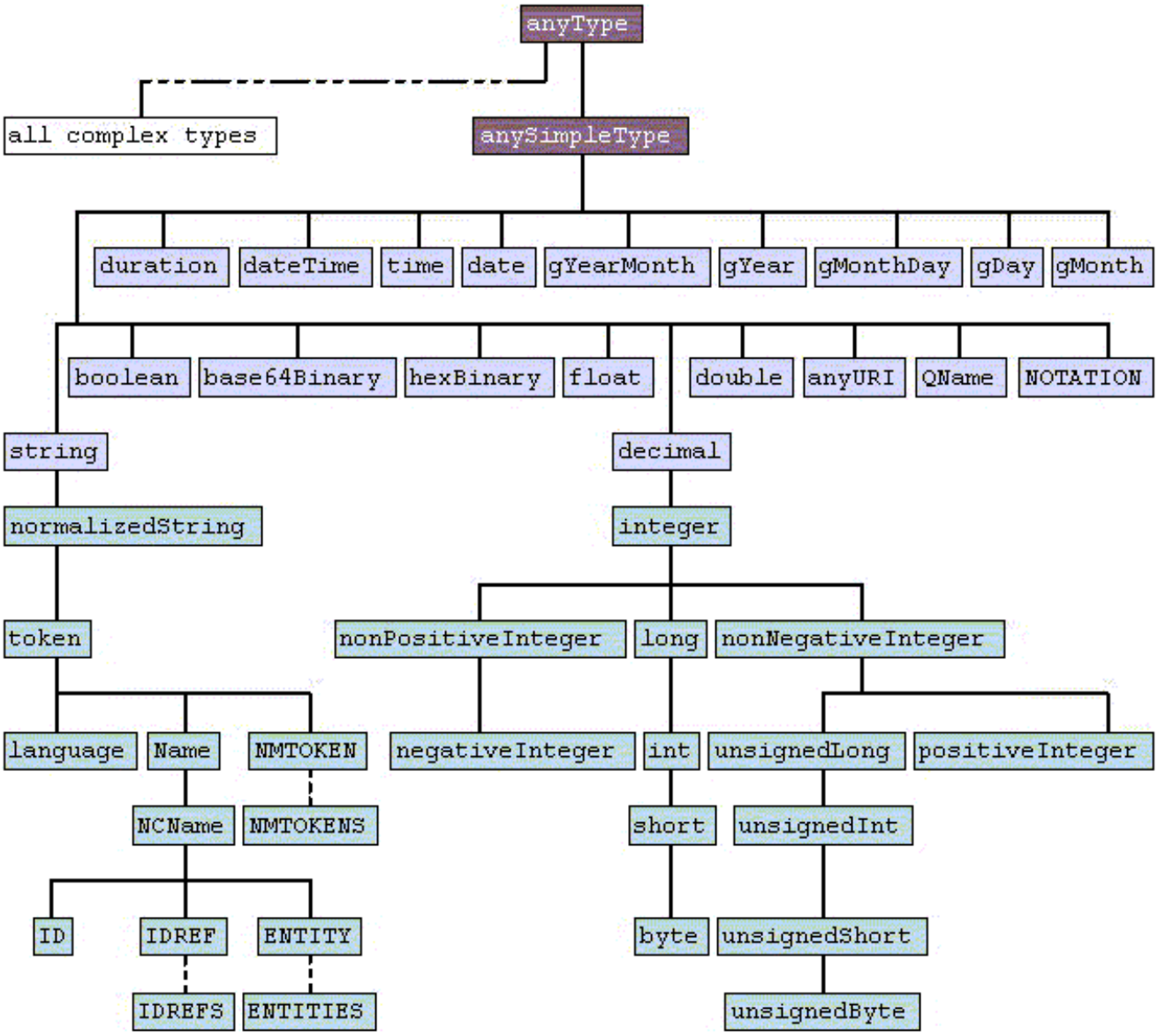

Figure 20 types of variables supported by Xml Schema

In fact, XML is not just a language for writing hyper texts. It is also a Meta language in which many languages can be written and in all disciplines and fields. The tags are not predefined, where but each person can introduce its own corresponding tags- Unlike HTML -. This flexibility makes XML a wide open space that Forms a galaxy of languages, techniques and related standards (Lonjon and Thomasson, 2006) ،

In the heart, we will find the original language (XML 1.0; XML 1.1). Then we find the parser (SAX; DOM ...). Then XML schemas languages to ensure the validity of the document structure (DTD; XML SCHEMA; ....). To view the XML pages in the form, the size and the style we want. We can use many Languages that are interested in the appearance of pages and format of the document in terms of fonts and colors, images position, backgrounds and many other criterias. CSS (Cascading Style Sheets) , XSL (Extensible Stylesheet Language) and which resulted in the (XSLT : XSL Transformations), that we use it to convert an XML file to other formats such as PDF. Or XPATH, which enables us to extract the data and identified it by scanning the document tree, And there are many query languages for xml such as XQuery, Xupdate, XML-QL, XQL, XIRQL, NEXI, Quilt ... etc (see Table 7).

Query languages can be classified into two categories (Lalmas, 2009) : query languages for content-only, and for content and structure. There are three major categories of query languages for the content and the structure namely: tag-based languages, path-based languages and clause-based languages. This space can accommodate markup languages and linking language of various parts of the document with itself or other documents. Like Xlink, Xinclude and XPointer etc… As for the exchange and send XML documents, we find the standards and protocols relating to it like SOAP, XSQL, XFORMS… etc. We also find specialized and descriptive languages like: Open document, SVG, RSS, MathML, XBRL, and CML. There are also standards data and terminology Description such as: RDF, OWL And specialized in Ontology.

Thus XML technology contributed to change the direction of the Web, and it make it more flexible and easier to handle with. And also print texts and display them became not affected if we change the platform or device or operating system(It does not matter if is a computer, phone, Tablet....etc). Document or text are written once and publish everywhere and in all styles. XML technology is designed to save and exchange of information and not to be presented unlike HTML, XML content is separated from appearance.

	Description	standards
REFERENCE	Language for building the structure and Schema of SGML and XML documents	**DTD**
	To identify the models of metadata and to characterize the terminology	**RDF**
	Language for building the structure and Schema of XML documents by the ISO organization	**Relax NG**
	Language for building the structure and Schema of XML documents	**Schematron**
	XML vocabulary to represent 2D illustrations.	**SVG**
	XML vocabulary to represent 3D illustrations.	**VRML . X3D**
	To integrate documents among them.	**Xinclude**
	link Language inside the document	**Xlink**
	parent Language.	**XML 1.0 et 1.1**
	To identify the information units allowed in the document XML.	**XML Infoset**
	Namespaces: grammatical rules applied to names of XML models.	**XML Namespace**
	Language for building the structure and Schema of XML documents by the W3C organization	**XML Schema**
	Language for labeling the information units from outside the document XML..	**Xpath**
	Targeting language and pointing units of information from inside the document XML	**Xpointer**
	XML vocabulary to describe the thematic links between resources	**XTM**
EXCHANGE AND TREATMENT	Language that specifies business processes.	**BPEL**
	Languages of the web services.	**SOAP, UDDI et WSDL**
	documents Model of a commercial nature	**UBL**
	ML vocabulary to describe the UML Model.	**XMI**
	Language to transform XML documents.	**XSLT**
	query Languages XML	**Xquery et Xupdate**
	XML vocabulary related to WAP for mobile phone.	**WML**
	Pattern display documents in browsers	**xHTML**
SECTORIAL	Aerospace	ATA 2200, S1000D, AECMA 2000M
	Cars	J2008
	Audiovisual	MPEG 4 et MPEG 7
	Banks	SwiftML, OFX, IFX, FpML
	Commerce	EbXML, UBL, xCBL, cXML
	Health	HL7
	Training	SCORM
	Mathematics	MathML
	Astronomy	AIML

Table 7 Examples of XML standards

2.2 Section II : XML Databases

Before going into the databases using XML technology, it is necessary to emphasize an important thing witch is that the xml technology separates the content and presentation. It is a basically a technique for information exchange, and for that reason we find two types of xml documents, which we will talk about it in the next paragraph.

2.2.1 Types of XML documents

XML documents can be classified according to their content in two main categories:

2.2.1.1 Data-Centric XML

In such documents, the data content is structured in fields such as tables in relational databases (Fields and properties and different categories). And the query on the data of is done using query languages, some of which we mentioned in the first section of this chapter, for example, XQuery. And sometimes is not required to keep this data or files stored always, this is due to the type of the data, the application or the user who requested it. Examples of data-centric documents are sales orders, flight schedules, scientific data, and stock quotes (Figure 21).

This type of Documents is characterized by:

- Fairly regular structure,
- Fine-grained data
- No mixed content.
- The order in which sibling elements occurs is generally not significant.

```
<OrdreDeVentes NumeroOrdreDeVentes="12345">
    <Client NumeroClient="543">
      <NomClient>ABC Industries</NomClient>
      <Rue>123 Main St.</Rue>
      <Ville>Chicago</Ville>
      <Etat>IL</Etat>
      <CodePostal>60609</CodePostal>
    </Client>
    <DateOrdre>981215</DateOrdre>
    <Item NumeroItem="1">
      <Lot NumeroLot="123">
        <Description>
          <p><b>Turkey wrench:</b><br />
          Stainless steel, one-piece construction,
          lifetime guarantee.</p>
        </Description>
        <Prix>9.95</Prix>
      </Lot>
      <Quantite>10</Quantite>
    </Item>
    <Item NumeroItem="2">
      <Lot NumeroLot="456">
        <Description>
          <p><b>Stuffing separator:<b><br />
          Aluminum, one-year guarantee.</p>
        </Description>
        <Prix>13.27</Prix>
      </Lot>
      <Quantite>5</Quantite>
    </Item>
</OrdreDeVentes>
```

Figure 21 data centric document Model

2.2.1.2 Text-Centric (Document-Centric) XML

Document-centric documents are (usually) documents that are designed for human consumption designed to save the text to be displayed on different browsers, just like HTML. And of course we don't need strict structuring of the data, such as those that oblige us in Data-Centric Documents. These documents are also characterized by a lot of these elements with mixed content. And then we need to Style Sheets languages like css or XSL, to format the appearance and show the final product (Figure 22).

This type of Documents is characterized by:

- Less regular or irregular structure.
- Lots of elements with mixed content.
- Granularity fine
- The order in which sibling elements occurs is generally not significant

```
<Produit>
  <Intro>
  The <ProductName>Turkey Wrench</ProductName> from <Developer>Full
  Fabrication Labs, Inc.</Developer> is <Summary>like a monkey wrench,
  but not as big.</Summary>
  </Intro>
  <Description>
  <Para>The turkey wrench, which comes in <i>both right- and left-
  handed versions (skyhook optional)</i>, is made of the <b>finest
  stainless steel</b>. The Readi-grip rubberized handle quickly adapts
  to your hands, even in the greasiest situations. Adjustment is
  possible through a variety of custom dials.</Para>
  <Para>You can:</Para>
  <Liste>
  <Item><Link URL="Order.html">Order your own turkey wrench</Link></Item>
  <Item><Link URL="Wrenches.htm">Read more about wrenches</Link></Item>
  <Item><Link URL="Catalog.zip">Download the catalog</Link></Item>
  </Liste>
  <Para>The turkey wrench costs <b>just $19.99</b> and, if you
  order now, comes with a <b>hand-crafted shrimp hammer</b> as a
  bonus gift.</Para>
  </Description>
</Produit>
```

Figure 22 Text centric document Model

2.2.2 Databases and XML

Due to the wide spread of xml technologies, and the expansion of fields for which can be used. There were more and more dependencies on networks and in practically on WEB for this technology.

with the emergence and development of standards, search techniques, querying and linking in XML documents Like Xquery, Xlink, Xpath, Xupdate and other techniques that are interested and data classification and exploitation. It appeared in conjunction with that the urgent need to save the data in databases until they will exploited optimally. And with different types of XML documents (text-centric or data-centric), this will lead us to decide what database solutions to use. There are two main solutions,

- XML-Enabled Databases
- Native XML database

2.2.2.1 XML-Enabled Databases

This type of DBMS uses relational database to store XML data (Table8), The motivation for this is that:

– Storage techniques and effective query languages have been developed for relational databases largely. And for that reason, it should not neglect this aspect and it should be exploited.
– The Economic challenge due to the relational nature of most existing databases, like Oracle, DB2 and Microsoft SQL server etc…

To store XML data in relational database, we must use third-party middleware to convert these data into relational tables and also to convert the XML queries used by XML query languages (ex: XQuery) to SQL queries and vice versa. This is done through linking, transforming and mapping from XML document schema to database schema and vice versa. We call a total mapping the process of transforming the entire XML document and all its data flow into relational database -This is in the Pure relational databases - or it could partial where some of the data flow is transformed into relational database According to the application demand. This is done in the databases that support XML - knowing that that XML is available in most of the existing DBMS- and we call this partial mapping where part of the XML document is stored

in relational table and the other part keeps the same schema and stored in files or BLOBs (Figure 23).

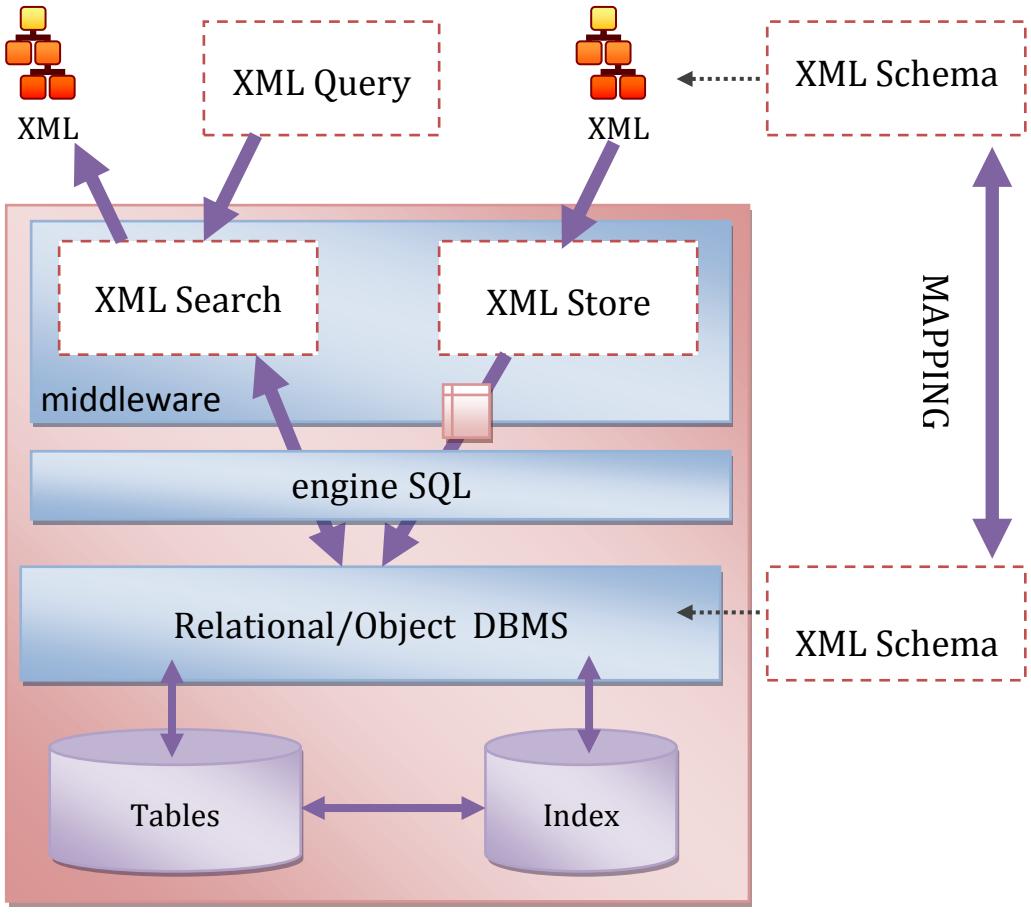

Figure 23 How to store XML sources in a relational database

To generate a relational schema from an XML schema:

1. For each complex element type, create a table and a primary key column.
2. For each element type with mixed content, create a separate table in which to store the PCDATA, linked to the parent table through the parent table's primary key.
3. For each single-valued attribute of that element type, and for each singly-occurring simple child element, create a column in that table.
   - If the XML schema has data type information, then set the data type of the column to the corresponding type.
   - If the XML schema has not a data type information, then set the data type to a pre-determined type, such as CLOB or VARCHAR(255).

- If the child element type or attribute is optional, make the column nullable.

4. For each multi-valued attribute and for each multiply-occurring simple child element, create a separate table to store values, linked to the parent table through the parent table's primary key.

5. For each complex child element, link the parent element type's table to the child element type's table with the parent table's primary key.

To generate an XML schema from a relational schema:

1. For each table, create an element type.
2. For each data (non-key) column in that table, as well as for the primary key column(s), add an attribute to the element type or a PCDATA-only child element to its content model.
3. For each table to which the primary key is exported, add a child element to the content model and process the table recursively.
4. For each foreign key, add a child element to the content model and process the foreign key table recursively.

To query an XML (XQuery...) in such a system, it will be done as follows

1. Transform the XML Query (Xquery …) to SQL statement.
2. Execute the generated SQL statements on the relational database.
3. Convert the returned result into XML document.

2.2.2.2 Native XML database:

The term "native XML database" first gained prominence in the marketing campaign for Tamino, due to the success of this campaign, the term came into common usage among companies developing similar products (Table 9).

Native XML databases must verify the following:

- Defines a logical model for an XML document that and stores and retrieves documents, it must include elements, attributes, PCDATA, and also it must take into account document order.

- Has an XML document as its fundamental unit of (logical) storage, just as a relational database has a row in a table as its fundamental unit of (logical) storage.
- Is not required to have any particular underlying physical storage model. For example, it can be built on a relational or use a proprietary storage format such as indexed, compressed files.

Product	developer	license	database type
Access 2007	Microsoft	Commercial	Relational
Cache	InterSystems Corp.	Commercial	Post-relational
DB2	IBM	Commercial	Relational, native XML
eXtremeDB	McObject	Commercial	Object-oriented
FileMaker	FileMaker	Commercial	FileMaker
FoxPro	Microsoft	Commercial	Relational
Informix	IBM	Commercial	Relational
Matisse	Matisse Software	Commercial	Object-oriented
MonetDB/SQL	CWI Database Group	Open Source	Relational
MySQL	Sun Microsystems	Open Source	Relational
Objectivity/DB	Objectivity	Commercial	Object-oriented
OpenInsight	Revelation Software	Commercial	Multi-valued
Oracle	Oracle	Commercial	Relational, native XML
Orient ODBMS	Orient Technologies	Open Source	Object-oriented
PostgreSQL	PostgreSQL Global Development Group	Open Source	Relational
RDM Embedded	Raima, Inc.	Commercial	Network, relational
RDM Server	Raima, Inc.	Commercial	Network, relational
Sentences	Lazy Software, Ltd.	Free	Associative
SQL Server	Microsoft	Commercial	Relational, native XML
Sybase ASE	Sybase	Commercial	Relational
UniData	IBM	Commercial	Nested relational
UniVerse	IBM	Commercial	Nested relational
Versant Object Database	Versant Corp.	Commercial	Object-oriented
ViewDS	eB2Bcom	Commercial	Proprietary (LDAP)

Table 8 Most important databases that support XML

2.2.2.3 Native XML Database Architectures

The architectures of native XML databases fall into two categories (Ronald, 2003): text-based and model-based:

2.2.2.3.1 Text-Based Native XML Databases

A text-based native XML database is one that stores XML as text. This might be a file in a file system, a BLOB in a relational database, or a proprietary text format. (It is worth noting that a relational database that has added XML-aware processing of CLOB (Character Large OBject) columns is, in fact, a native XML database with respect to these abilities.).

Common to all text-based native XML databases are indexes, which allow the query engine to easily jump to any point in any XML document. This gives such databases a tremendous speed advantage when retrieving entire documents or document fragments. This is because the database can perform a single index lookup, position the disk head once, and, assuming that the necessary fragment is stored in contiguous bytes on the disk, retrieve the entire document or fragment in a single read. In contrast, reassembling a document from pieces, as is done in a relational database and some model-based native XML databases, requires multiple index lookups and multiple disk reads.

2.2.2.3.2 Model-Based Native XML Databases

Rather than storing the XML document as text, they build an internal object model from the document and store this model. How the model is stored depends on the database. Some databases store the model in a relational or object-oriented database. Other databases use a proprietary storage format optimized for their model.

Model-based native XML databases that use a proprietary storage format are likely to have performance similar to text-based native XML databases when retrieving data in the order in which it is stored. This is because they use physical pointers between nodes.

Text-based systems are obviously faster at returning documents as text, while model-based systems are obviously faster at returning documents as DOM trees. Like text-based native XML databases, model-based native XML databases are likely to encounter performance problems when retrieving and returning data in any form other than that in which it is stored.

2.2.2.4 Features of Native XML Databases

In this section, we briefly discuss a number of the features found in native XML databases. And at the end, we will have an idea of what features are available today and what features to expect in the future.

2.2.2.4.1 Document Collection

Many native XML databases support the notion of a collection. This plays a role similar to a table in a relational database or a directory in a file system. For example, suppose you are using a native XML database to store sales orders. In this case, you might want to define a sales order collection so that queries over sales orders could be limited to documents in that collection.Whether collections can be nested depends on the database.

2.2.2.4.2 Query Languages

Almost all native XML databases support one or more query languages. The most popular of these are XPath and XQuery.

2.2.2.4.3 Updates and Deletes :

Native XML databases have a variety of strategies for updating and deleting documents, from simply replacing or deleting the existing document to modifications through a live DOM tree to languages that specify how to modify fragments of a document. Xupdate represents the most important standard language for updating XML.

2.2.2.4.4 Transactions, Locking, and Concurrency

Virtually all native XML databases support transactions and presumably support rollbacks. However, locking is often at the level of entire documents, rather than at the level of individual nodes, so multi-user concurrency can be relatively low. Whether this is an issue depends on the application and what constitutes a document. For example: A document is a chapter of a user's guide and writers edit chapters. Document-level locking is unlikely to cause concurrency problems, as two writers updating the same chapter at the same time is unlikely. In

another example, we have: A document contains the data used in a workflow, such as a financial contract. Each step of the workflow reads data from the document and adds its own data. For example, one step might perform a credit check and add a credit score to the document. Another step might check for outstanding balances on other contracts with the same customer and add the total outstanding balance. If node-level locking is used, some of these steps may be executed in parallel. If document-level locking is used, they must be executed serially to avoid write conflicts. This may cause unacceptable delays in high-volume applications.

The problem with node-level locking is implementing it. Locking a node usually requires locking its parent, which in turn requires locking its parent, and so on back to the root, effectively locking the entire document. To see why this is true, consider a transaction that reads a leaf node. If the transaction does not acquire locks on the ancestors of the leaf node, another transaction can delete an ancestor of the leaf node, in turn deleting the leaf node. However, it is also clear that another transaction should be able to update those parts of the document not on the direct path from the root to the leaf node.

A partial solution for this problem is proposed by Stijn Dekeyser, et al.While they do not entirely avoid the problem of locking the ancestors of a target node, they do make these locks more flexible by annotating them with the query defining the path from the locked node to the target node. This allows other transactions to determine whether they conflict with transactions already holding locks.

2.2.2.4.5 Application Programming Interfaces (APIs)

Almost all native XML databases offer programmatic APIs. These are usually in the form of an ODBC-like interface, with methods for connecting to the database, exploring metadata, executing queries, and retrieving results. Results are usually returned as an XML string, a DOM tree, or a SAX Parser over the returned document. If queries can return multiple documents, then methods for iterating through the result set are available as well. Among the developed APIs, we have: XML:DB API from XML:DB.org and JSR 225: XQuery API for Java (XQJ).

Most native XML databases also offer the ability to execute queries and return results over HTTP.

2.2.2.4.6 : Round-Tripping

One important feature of native XML databases is that they can round-trip XML documents. That is, you can store an XML document in a native XML database and get the "same" document back again. This is important to document-centric applications, for which things like CDATA sections, entity usage, comments, and processing instructions form an integral part of the document. It is also vital to many legal and medical applications, which are required by law to keep exact copies of documents.

Round-tripping is less important to data-centric applications, which generally care only about elements, attributes, text, and hierarchical order. All software that transfers data between XML documents and databases can round-trip these.

All native XML databases can round-trip documents at the level of elements, attributes, PCDATA, and document order. How much more they can round-trip depends on the database. Text-based native XML databases round-trip XML documents exactly, while model-based native XML databases round-trip XML documents at the level of their document model.

Since the level of round-tripping you need depends entirely on your application, you may have a choice of many native XML databases or be restricted to only a few.

2.2.2.4.7 : Remote Data

Some native XML databases can include remote data in documents stored in the database. Usually, this is data retrieved from a relational database with ODBC, OLE DB, or JDBC and modeled using the table-based mapping or an object-relational mapping.

To realize the concept of remote data, That is, any updates to the document in the native XML database must be updated in the rational database. Because it is considered as the original database, where the data has been retrieved from it. Most native XML databases will probably support this concept.

2.2.2.4.8 Indexes

All native XML databases support indexes as a way to increase query speed. These are three types of indexes. Value indexes, Structural indexes and full-text indexes.

2.2.2.4.9 External Entity Storage

A difficult question when storing XML documents is how to handle external entities. That is, should they be expanded and their value stored with the rest of the document, or should the entity reference be left in place? There is no single answer to this question.

For example, suppose a document includes an external general entity that calls a CGI program for the current weather report. If the document is used as a Web page to give current weather reports, it would be a mistake to expand the entity reference, as the Web page would no longer return live data. On the other hand, if the document were part of a collection of historical weather data, it would be a mistake *not* to expand the entity reference, as the document would always retrieve the current data rather than containing the historic data.

As another example, consider a product manual that consisted of nothing but references to external entities that point to the chapters of the manual. If some of these chapters were used in other documents, such as manuals for differents model of the same product, it would be a mistake to expand these references, as this would result in multiple copies of the same chapters.

Product	Developer	License	Database Type
4Suite, 4Suite Server	FourThought	Open Source	Object-oriented
BaseX	University of Konstanz	Open Source	Proprietary
Berkeley DB XML	Oracle (formerly owned by Sleepycat Software)	Open Source	Key-value
DBDOM	K. Ari Krupnikov	Open Source	Relational
dbXML	dbXML Group	Open Source	Proprietary
Dieselpoint	Dieselpoint, Inc.	Commercial	None (indexes only)
DOMSafeXML	Ellipsis	Commercial	File system(?)
EMC Documentum xDB (formerly X-Hive/DB)	EMC Corporation	Commercial	Proprietary
eXist	Wolfgang Meier, et al	Open Source	Proprietary
eXtc	M/Gateway Developments Ltd.	Free	Post-relational
Extraway	3D Informatica	Commercial	Files plus indexes
Infonyte DB (formerly PDOM)	Infonyte	Commercial	Proprietary (Model-based)
Ipedo XML Database	Ipedo	Commercial	Proprietary
Lore	Stanford University	scientific	Semi-structured
MarkLogic Server	Mark Logic Corp.	Commercial	Proprietary
M/DB:X	M/Gateway Developments Ltd.	Open Source	Hierarchical (GT.M)
MonetDB/XQuery	CWI Database Group	Open Source	Proprietary
myXMLDB	Mladen Adamovic	Open Source	MySQL
Natix	University of Mannheim	Free	Proprietary
ozone	ozone-db.org	Open Source	Object-oriented
Qizx	XMLMind, a division of Pixware	Commercial	Proprietary
Sedna XML DBMS	ISP RAS MODIS	Free	Proprietary
Sekaiju (known as Yggdrasill in Japan)	Media Fusion	Commercial	Proprietary
SQL/XML-IMDB	QuiLogic	Commercial	Proprietary XML store plus relational store
Sonic XML Server (formerly eXcelon)	Sonic Software (who bought eXcelon Corp.)	Commercial	Object-oriented
Tamino	Software AG	Commercial	Proprietary. Relational through ODBC.
TEXTML Server	IXIASOFT, Inc.	Commercial	Proprietary (Document-based)
TigerLogic XML Data Management Server (XDMS)	Raining Data	Commercial	Pick
Timber	University of Michigan	Open Source	Shore, Berkeley DB
TOTAL XML (formerly Socrates XML)	Cincom	Commercial	Object-relational, external relational through ODBC
Virtuoso	OpenLink Software	Commercial	Proprietary. Relational through ODBC
Xindice (see also dbXML)	Apache Software Foundation	Open Source	Proprietary (Node-based)
xml.gax.com (formerly NaX Base)	GAX Technologies (bought from Naxoft)	Commercial	Proprietary
Xpriori XMS	Xpriori	Commercial	Proprietary
XStreamDB Native XML Database	Bluestream Database Software Corp.	Commercial	Proprietary (Node-based)
Xyleme Zone Server	Xyleme SA	Commercial	Proprietary (Natix)

Table 9 Most important Native XML databases Products

2.3 Section II : XML Data warehouses.

We talked earlier in this thesis, about data warehouses (its definition, its importance, its classification and its modeling .....). Then in the first section of the second chapter, we detailed the XML Technology (bases, rules and the different standards of this technology .....). then in the second section of the same chapter about XML databases and its types (XML-Enabled Databases and native XML database). And now we come to merger all these technologies and sciences under the name **XML data warehouse** which will cover all of the XML technologies. With the expansion of the use of networks and in particular the World Wide Web, and the adoption of this technologies by many companies and institutions to communicate with their clients, their customers and even with their staff and users, and information delivery to Concerned on the products and services offered by these companies. However, the size of the mobile data increased across the network, these data are mostly in complex form (text messages, images, audio clips, video ....).

And what we already knew about the XML technology and its flexibility and the best solution for dealing with complex data (Darmont et al., 2007), especially with the emergence of many XML standards that is specialized in certain types of non-simple file (MPEG 4, MPEG 7).

XML data warehouse is a data warehouse that meets the following characteristics (Ouaret, 2008) :

- Unlike traditional data warehouse, the input of data sources is XML files (documents XML, XML Schema, DTD …etc).
- Storage can be done in the original XML database or in relational database.
- The Output data or XML document must be in a suitable form for OLAP queries.
- The ability to exchange information with HTTP in the intranet or the Internet.
- The support for the XML query technologies (XQuery, XSLT, XPATH…etc).
- Data collect from different sources must be VALID and respect the storage schemas.

(Mahboubi and Darmont, 2009) defined XML warehouse data as a set of XML data which is organized in a multi-dimensional manner in order to support XML queries and decision-making queries.

We also have two types of XML documents, Data-Centric and Text-centric documents. This means that according to the content and the type of the document, we find two types of XML warehouses: XML data warehouses and XML documents warehouses.

2.3.1 Related works (State of the art)

Since the beginning of the year 2000, and researchers have cared (Abdelhédi, 2014)) on Modeling of the XML data warehouses, and they have made significant contributions. We mention here some of previous research in the area:

❖ In (Golfarelli et al., 2001), the authors define a semi-automatic approach to create XML data stores and to develop of traditional multi-dimensional from the original XML documents, which they have the same DTD. This approach consists of a semi-automatic four steps:

1. Simplifying the DTD.
2. Creating a DTD graph.
3. Choosing facts by the designer.
4. Defining dimensions and measures.

Following the same principle, the authors in (Vrdoljak et al., 2006) suggested to work on XSchemas instead of DTD , because these schemes provide a richer semantic.

❖ Authors in **(Pokorny, 2001)**, defined an approach based on simple star schemes with explicit hierarchy, the Hierarchy is clear and explicit if it contains a set of referential constraints. This approach, which is called «**XML-Star**» (Figure 24), allows modeling and querying on data warehouse based on data source which is a set of XML documents. This approach is also based on Views or virtual tables and DTDs.

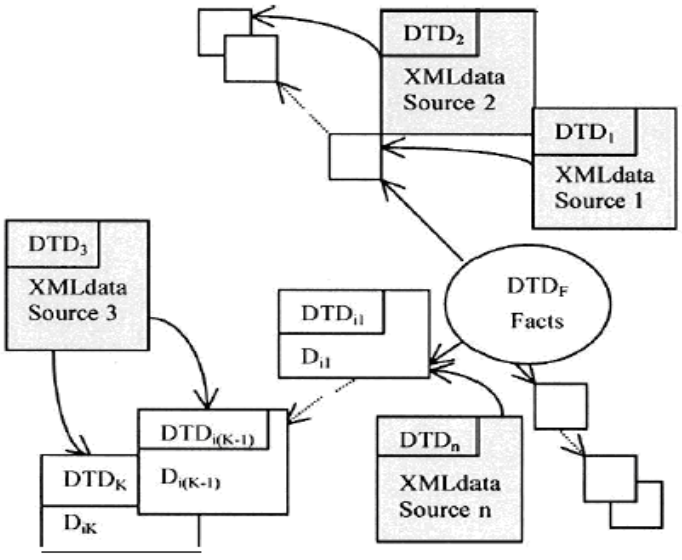

Figure 24 XML-Star Model

❖ In **(Niemi et al., 2002),** The authors propose a system based on XML to collect data. The system is well-defined and targeted because the information it needs is well known, and also measures, dimensions and constraints for the cube are known through queries and XML documents. This system is based on the star schema. The authors studied how to view OLAP cube using XML by methodological presentation of the various elements in the relational model. the proposed data by users are defined using MDX, a Declarative query language for multidimensional databases (Figure 25).

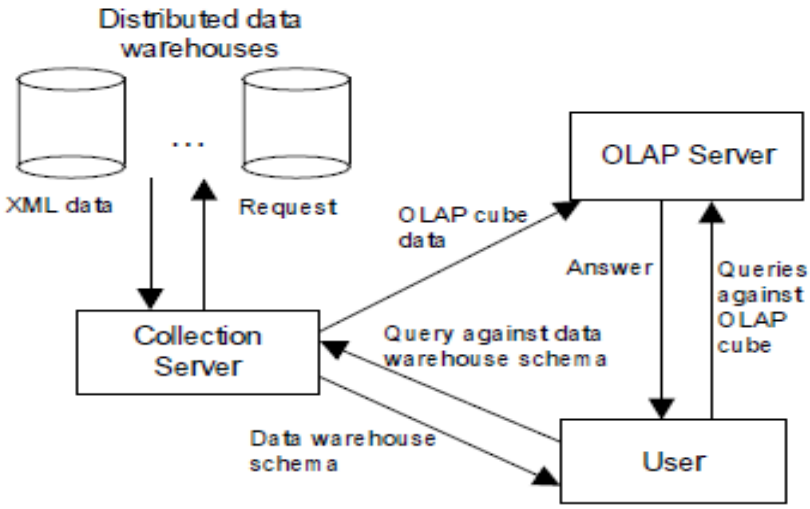

Figure 25 Niemi et al., 2002 System archtecture

❖ In **(Vrdoljak et al., 2003),** The authors propose a semi-automated methodology for designing web warehouses from XML sources modeled by XML Schemas. In the proposed methodology, design is carried out by first creating a schema graph, then navigating its arcs in order to derive a correct multidimensional representation. The authors gave a particular relevance to the problem of detecting shared hierarchies and convergence of dependencies, and of modeling many-to-many relationships. The approach is implemented in a prototype that reads first an XML Schema and produces in output the logical schema of the warehouse.

❖ In **(Hümmer et al., 2003)**, The authors propose an approach called **XCube** which uses three basic XML schemas for the storage and exchange of data and information queries from one or more data warehouse. This is a multidimensional schema and dimension schema and facts values schema (Figure 26). which are:

- **XCubeSchema:** to describe the multidimensional structure of the data cube. It also defines the facts and dimensions under the node **cubeSchema**, while **classSchema** describes the classification levels of the dimensions involved.
- **XCubeDimension:** Is meant to formalize the dimensional structures, where every dimension is described through its classification **level** , **attribute** and the reference to the highest classification level.
- **XCubeFact :** Which determines the facts, through the cube, or cell branching with it.

The authors gave each module separated so it will be possible to re-use in many cubes in the same application and analysis of multi-dimensional different specificities.

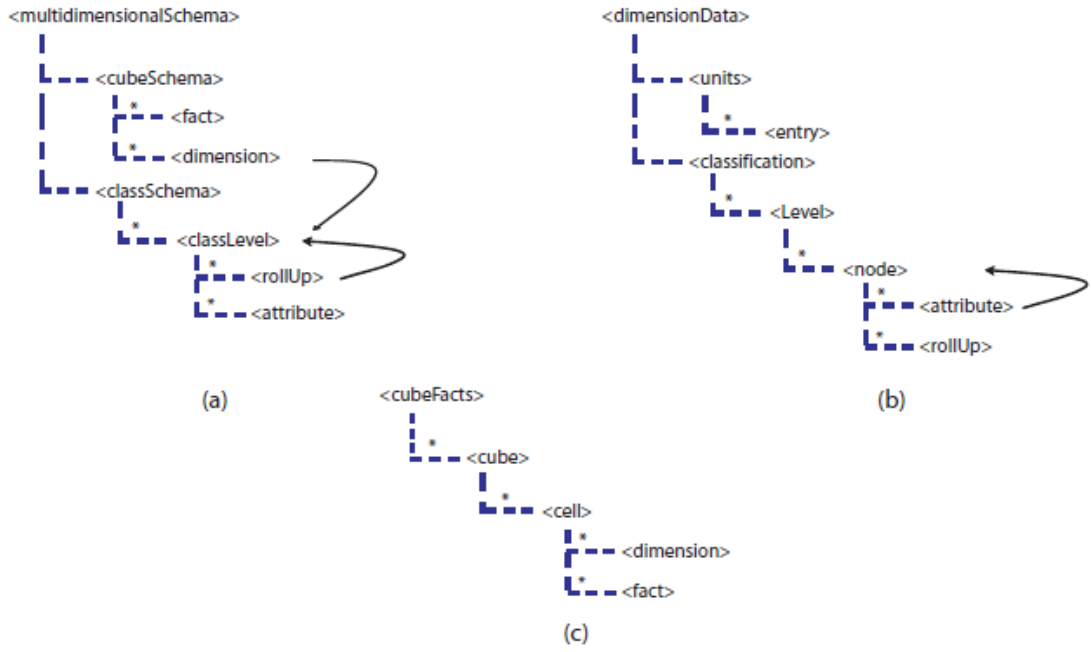

Figure 26 XCube model and the three schemas

❖ In **(Zhang et al., 2003),** authors propose an approach to materialize XML data warehouses based on frequent query patterns discovered from historical queries issued by users. This approach allows dealing with distributed XML data sources that are represented as DTD. It starts by determining the XML data sources that are more frequently accessed by users. To integrate XML data, the authors used hierarchical clustering technique that allows access to XML documents and maintain them. The proposed approach used in the input two trees from two different sources of data, the first from XML sources, and the second from users' queries.

❖ In **(Nassis et al., 2004)**, The authors provide a multi-dimensional model based on a set of documents XML (xFACT) (Figure 27). This approach is based on the Unified Modeling Language (UML) To build XML document warehouse. To design VDIM (Virtual DIMensions), logically conceptual views on the Data Warehouse will be defined, and from that point the name virtual dimensions came. This method consists of three levels:

1. User Requirement Level,
2. Conceptual Level,
3. XML schema level.

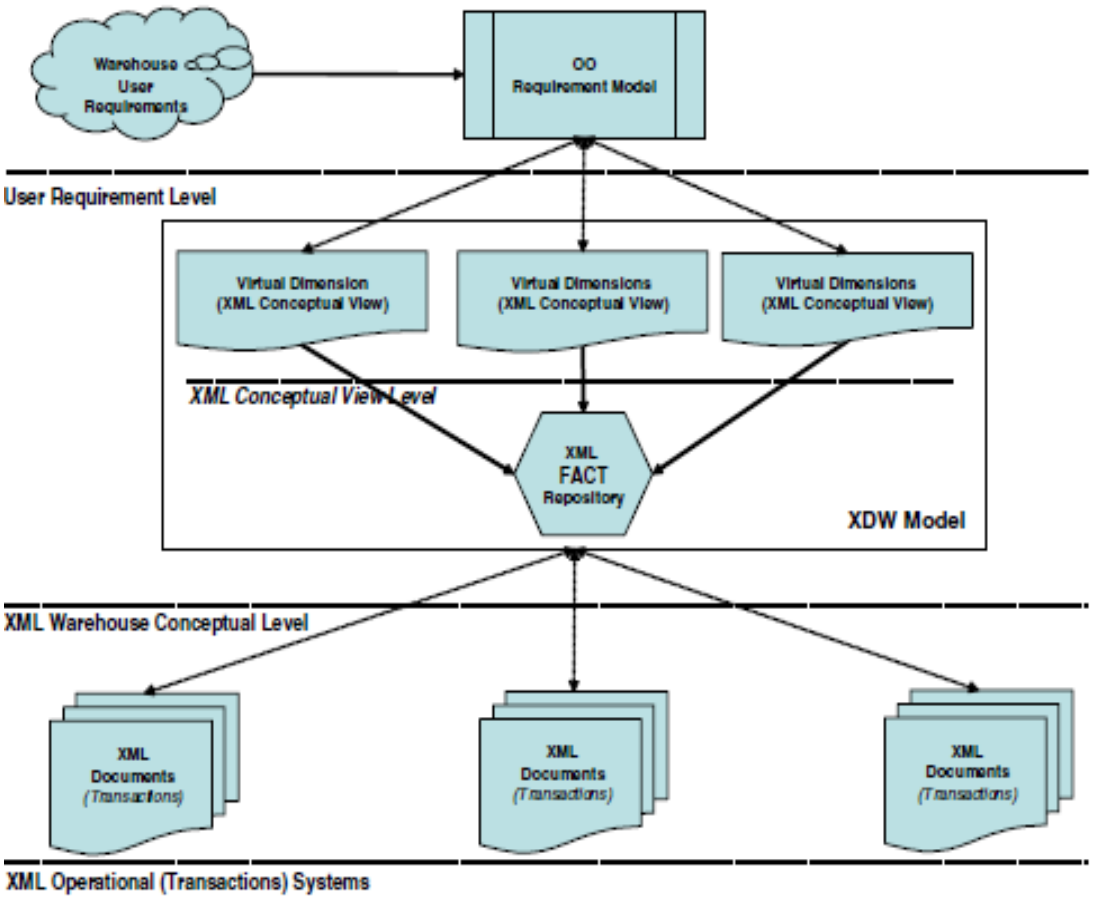

Figure 27 xFACT model and its levels

❖ In **(Rusu et al., 2005)** , authors propose a practical methodology for building XML documents data warehouses. This proposed approach covers processes including data extraction, conversion, integration, aggregation of XML documents according to the following steps:

- Semi-automatic method to solve the problem of increasing conflict in integration through data cleaning.
- Summarize the data, which includes creating dimensions and creating the corresponding XML documents, establish links between the dimensions to create fact document.

In these different phases and with the process of creating a data warehouse, the authors use XQuery to generate different XML documents of dimensions and facts.

❖ In **(Zhang et al., 2005)**, authors propose an approach to materialize XML data warehouses based on historical queries issued by users, called X-Warehouse (Figure 28). The data warehouse is constructed by discovering frequent query paths.

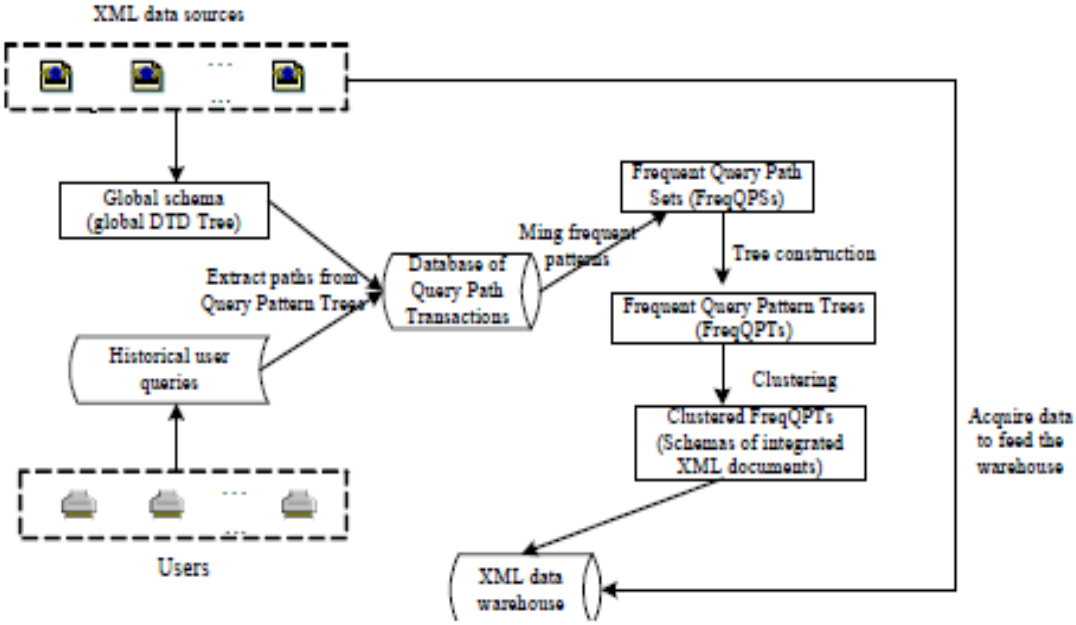

Figure 28 Overview of X-Warehouse

❖ In **(Rajugan et al., 2005)**, Authors propose an approach for conceptually modeling and designing for a Global XML FACT repository based on XML FACT repositories (xFACT) proposed in (Nassis et al., 2004). Using MDA (Model-Driven Architecture), a platform independent models that play a vital role in system development and data engineering (Figure 29), the authors grouped logically a geographically dispersed XML document warehouses and Document. Under the MDA, initiative, first the model of a system is specified via an abstract notation independent of the technical or deployment specifications (i.e. Platform Independent Model or PIM) and then the PIM is mapped or transformed into a deployment model (i.e. Platform Specific Model or PSM) by adding platform or deployment specific information into the PIM.

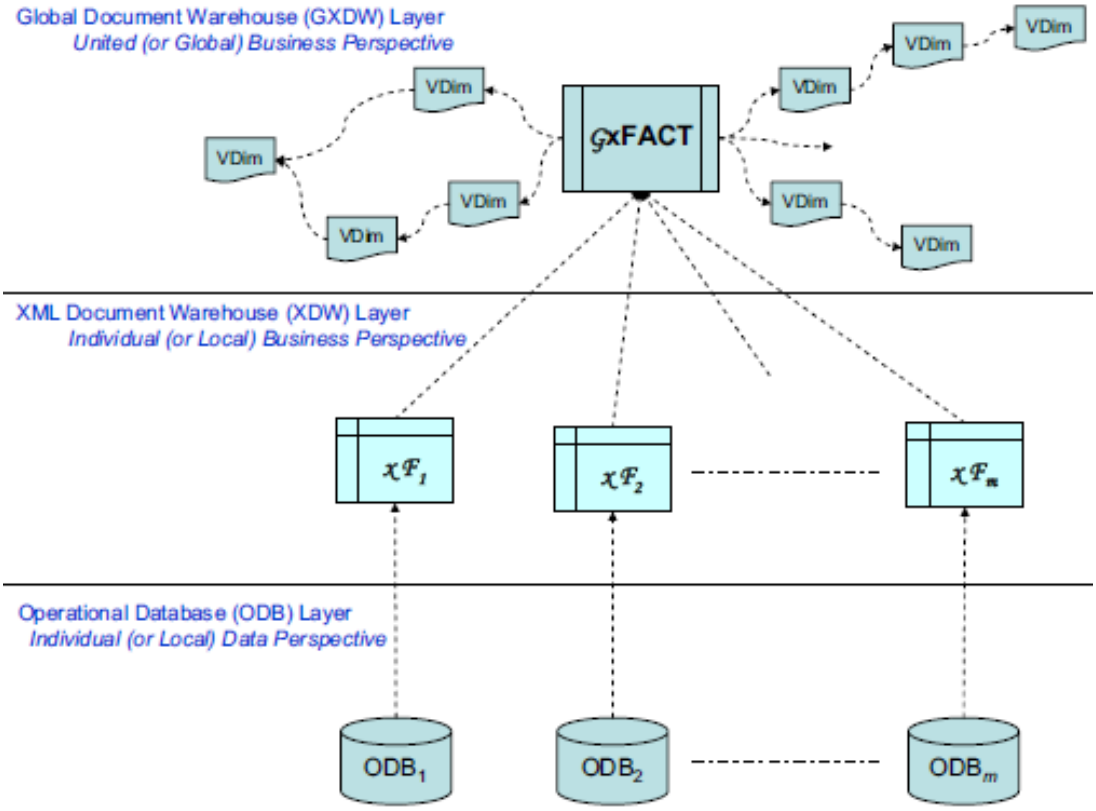

Figure 29 GxFact , Context Diagram

❖ In **(Li and An, 2005),** authors propose an XML Schema-based approach to construct an XML data warehouse based on Snowflake schema. in the following manner (Figure 31):

- Transform the XML schema into UML class diagrams.
- Transform UML diagram into UML snowflake diagram.
- Transform UML to XML schema.

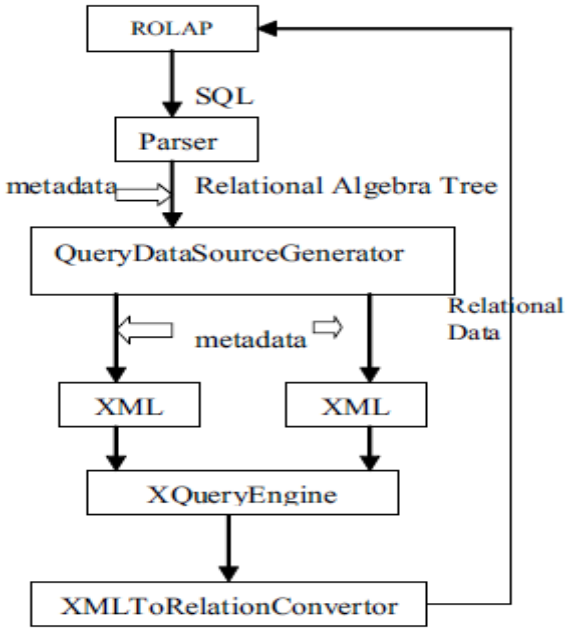

Figure 31 Implementation Architecture in Li and An, 2005

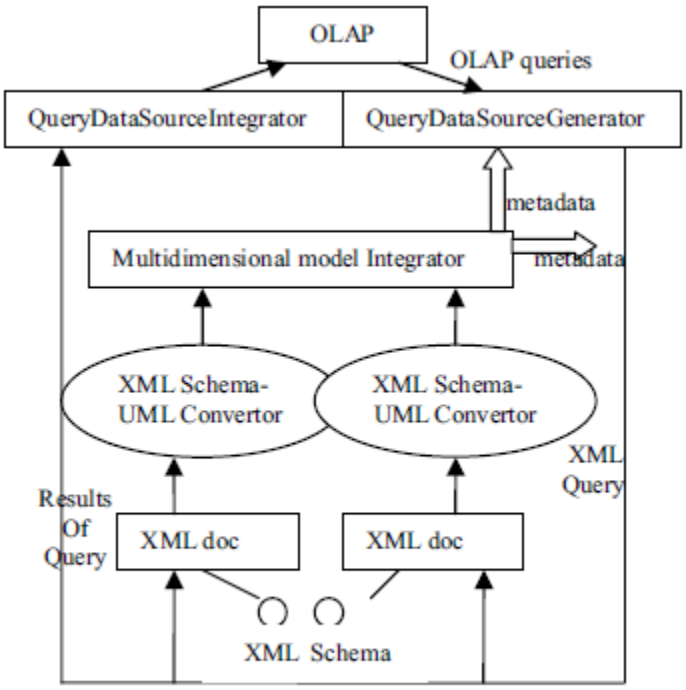

Figure 31 Integration Architecture in Li and An, 2005

- **In (Boukraâ et al., 2006a)** Based on a logical model represented in a single XML schema and proposed in(Boussaid et al., 2006a) ‹The authors suggested the separation of facts and dimensions, and thus, will be to collect all the facts in an XML document, as sub-elements of the root element, specify by names in the plural.each dimension includes a separate XML document, and links between XML documents (dimensions and facts) are represented by virtual keys mechanism that serve as primary keys and foreign keys, similar to those that are in the relational model

- **In (Boussaid et al., 2006b)** The authors propose a systematic approach built on the basis of the analysis needs,for multi-dimensional complex data modeling called it «X-Warehousing», Based on the objectives of the user analysis which is represented in the (MCM) Multidimensional Conceptual Model (Figure 32), expressed in XML schema and a set XML documents for analysis, produces a homogeneous set of data with the stringent restrictions on the contents. And organize this data in documents XML, each containing a Fact with measures and dimensions. This collection of XML documents represent OLAP cube. It is directed towards the direct analysis. On the whole this approach is entirely based on XML, it

allows the design data warehouse (datamart), to represent a conceptual scheme using XML Schema, finally, is fed a multi-dimensional structure of by data stored in XML documents.

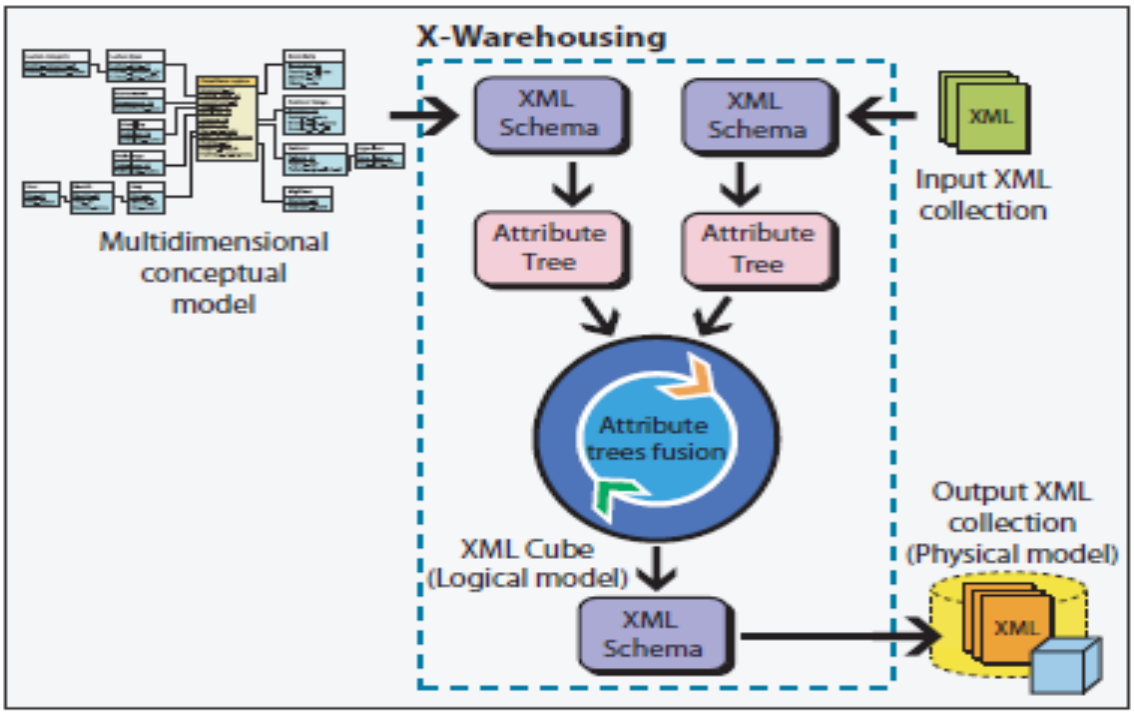

Figure 32 X-Warehousing Model

❖ **In (Tseng and Chou, 2006),** the authors propose a general architecture for constructing document warehouses, where the document are from internal and external data sources. These data will be processed (like texts' summaries), In order to extract the elements that will be saved as descriptive data. The system constructs document cubes with a star schema through the creation of cubes and indexes (Figure 33). The authors distinguished the dimensions into three types:

1. **Ordinary dimension:** that contains a set of keywords to allow analyzing the documents.
2. **Metadata dimension:** that includes descriptive data extracted from documents (title, creator, date).
3. **Category dimension:** that contains external data related and متصلة to the document.

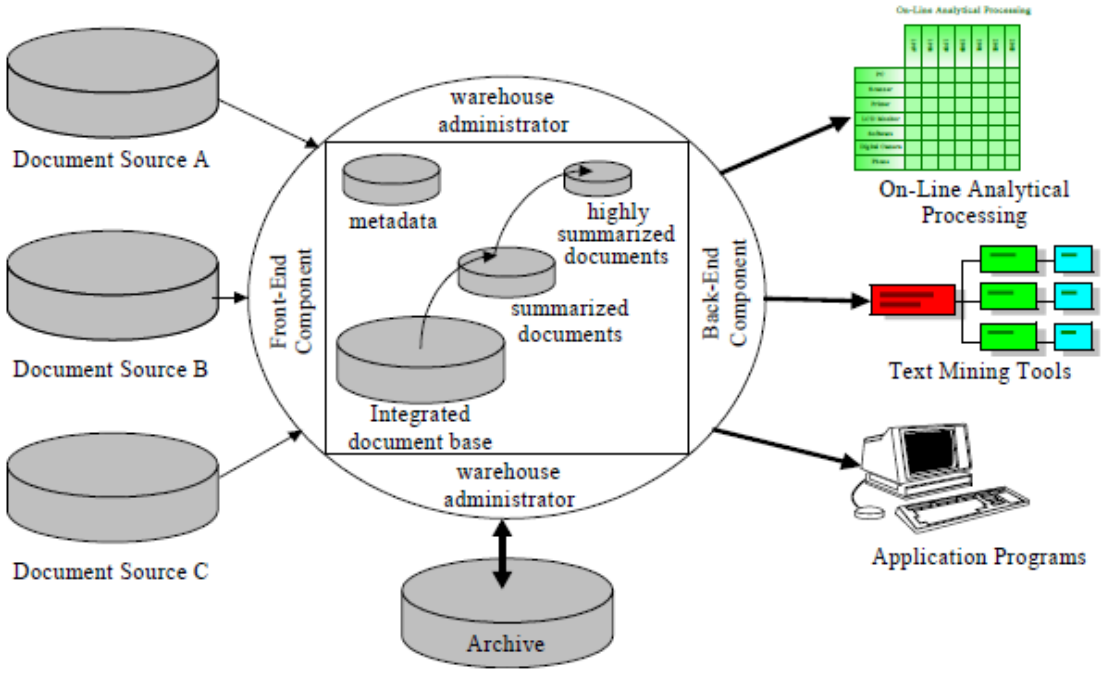

Figure 33 Tseng and Chou, 2006 Model

❖ **In (Song et al., 2008)**, The authors propose a new semi-automatic approach called **«SAMSTAR»**, based Structural Heuristics research, for automatic extraction of candidate fact from an ERD (Entity-Relationship Diagrams). By analyzing the number of connections of an entity through **M:1** relationships. Thus the dimensions of each fact or the candidate fact are generated based on the links that are directly connected to the fact entity. This approach aims to simplify the work of the designer to develop a subset of the multi-dimensional candidate schemas. This approach do not take into account the dimensions hierarchy, it only generates fact and dimensions. In (Kim et al., 2009), authors propose a variations of this approach called «**SAMSTARplus**» (Figure 34)

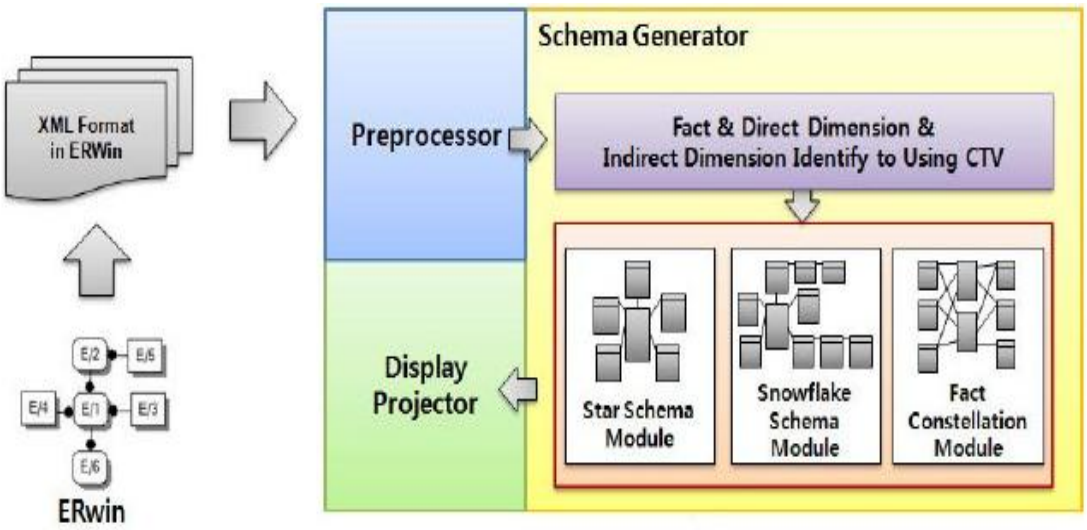

Figure 34 SAMSTARplus System Architecture

❖ **In (Pinet and Schneider, 2009)**, The authors propose a new model multi-dimensional schema from a UML conceptual schema. This model represents the source classes in directed acyclic graph, the decision maker represent the nodes as facts, and all the nodes related to the chosen fact represent the possible dimension for that fact.

❖ **In (Hachaichi et al., 2010)**, the authors present an automatic method to design a multidimensional schemas from DTD, This method processes the most relevant elements to the analysis in these Schemas to guide the decision maker to express his needs. It is based on three steps implemented in a tool called CAME-XML (Conception Assistée de Magasin de données en Etoile à partir de source XML) (Figure 35). These three steps are:

1. Preprocessing,
2. Construction datastore schemas (MD Magasin de données),
3. Adjustment / Validation.

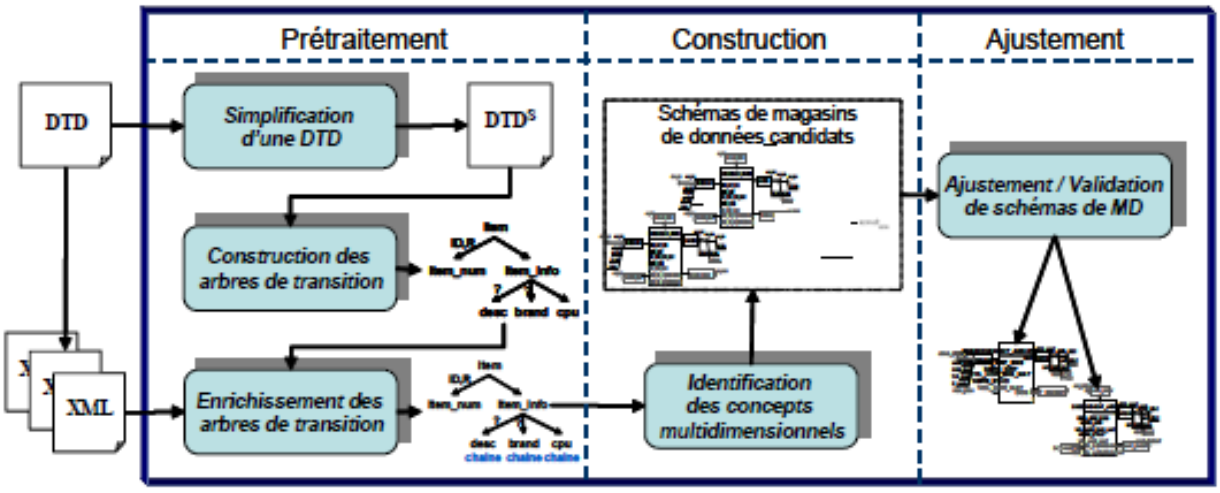

Figure 35 CAME-XML Steps

❖ **In (Carmè et al., 2010)**, The authors propose an approach based on models to support designers in the process of creating a data warehouse. From relational data sources, the system automatically detects the tables that model facts. according to them, the table may be a fact if the following three inferences are true:

1. If it contains a higher number of instances than other tables.
2. If it has a large ratio of numerical attributes compared to non-numerical attributes.
3. If it has a few foreign keys than other tables.

The result is a set of candidate multidimensional schemas for the discovered (Figure 36)

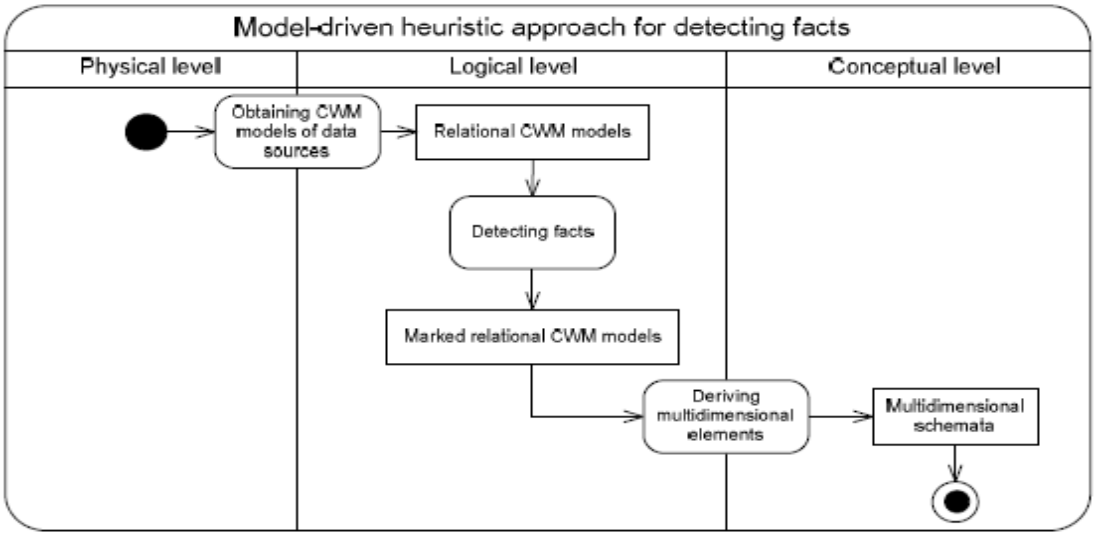

Figure 36 Model-driven heuristic Approch for detecting facts

❖ **In (Pujolle et al., 2011),**The authors propose a new multidimensional model for analyzing data extracted from XML text-rich documents, they called it **«A Galaxy»** (Figure 37), the galaxy is based on:

- a unique dimension concept that represents an analysis axis, but also a possible analysis subject;
- Groupings of these dimensions to show their compatibility for analysis specification. The model also allows linking attributes together.

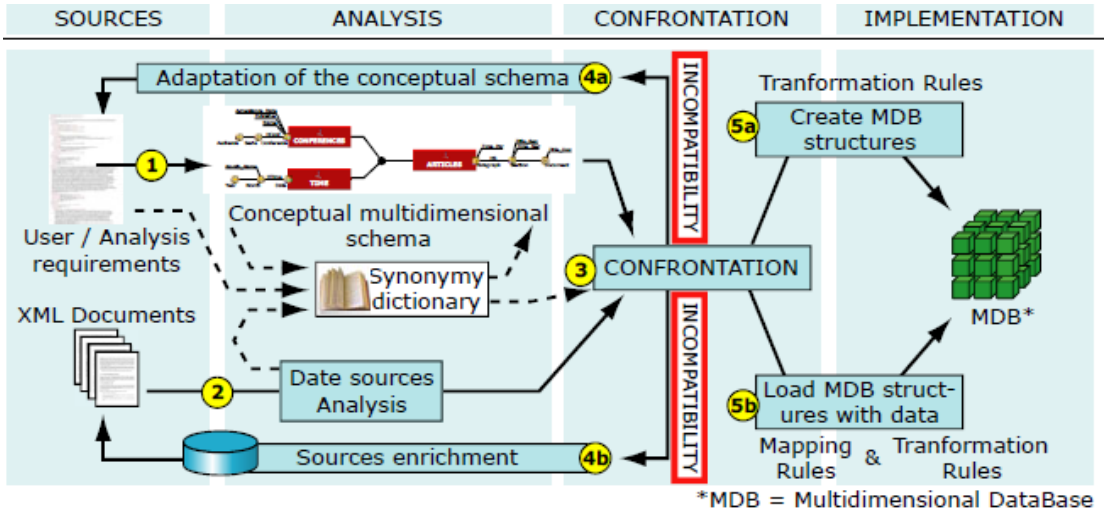

Figure 37 General overview of A Galaxy composed of 5 stages

❖ **In (Boukraâ et al., 2013)**, The authors propose a multidimensional model (Figure 38), determined by a meta-model which is created by UML. The proposed model consists of three layers:

- The first layer class diagram describes complex objects and captures the hierarchical organization of their attributes.
- The second layer package of classes describes the multidimensional model as a set of complex objects that are connected by relationships and some of which are organized in hierarchies.
- The multidimensional model allows layer the users to design complex cubes using the cubic projection.

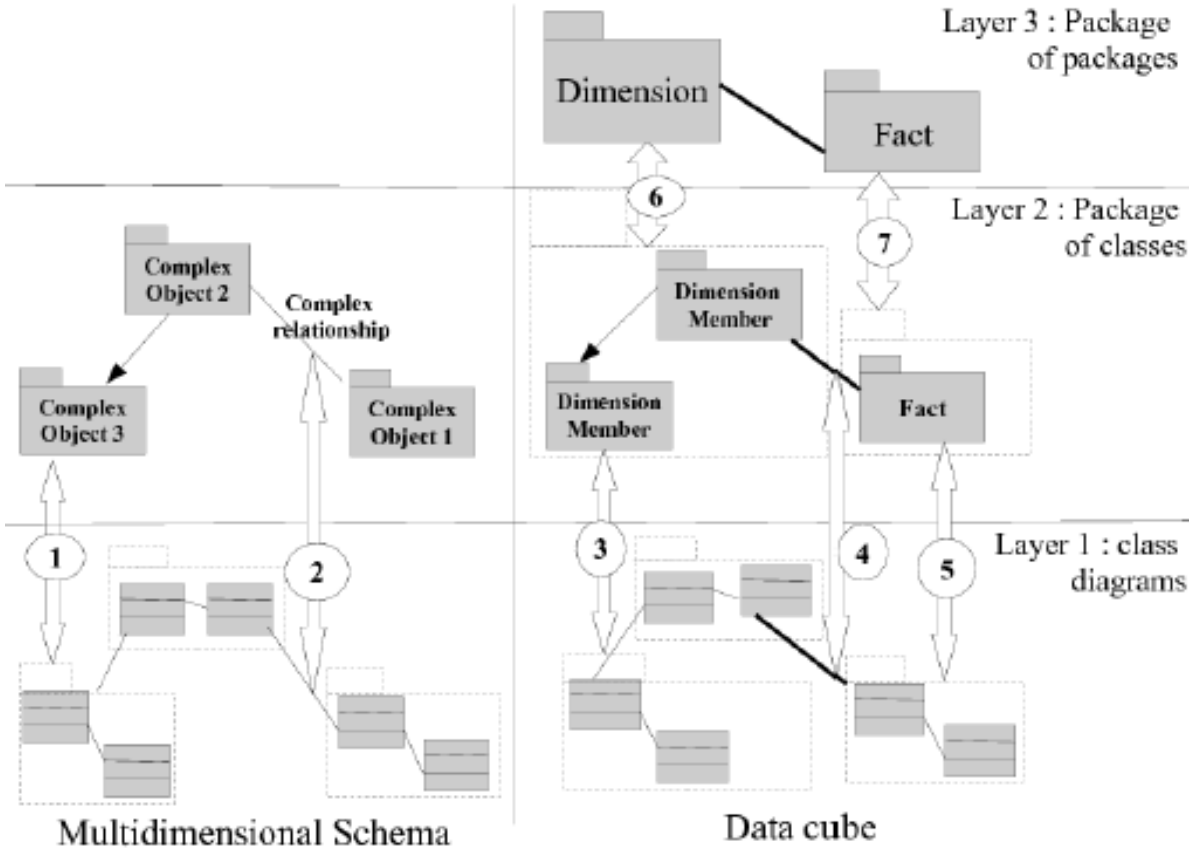

Figure 38 Genaral principle of Boukraâ et al., 2013 Approach

2.4 Conclusion::

This chapter shows us the essence of XML technology, and its diversity, starting by the main language, parsers, schemes and other, As we passed on the use of XML documents in databases and data warehouses, and also saw a large number of products to manage this last, which is a clear evidence of the XML technology and its role in keeping and circulation of data, creating a lot of research that is studying the development of scientific approaches to operate databases and XML data warehouses, and especially those related multi-dimensional model.

CHAPTER 3

Techniques To Improve Performance In Data Warehouses

Summary:

In this chapter, we will work on various basic techniques that improve performance for databases and data warehouses.

3) Chapter 3: Techniques to improve performance in data warehouses

Data warehouses are very large databases, and its size is constantly increasing due to the information streaming in every moment, The institutions do not stop working and conducting various operations, and the need for more information in order to make appropriate decisions do not stop, this information stored in warehouses mainly oriented to the process of decision-making, Is a process that of great importance for business evolution and companies, it is also a complex process and requires a careful analysis and thorough of all information and data available and collected in the data warehouse, This analysis is carried out by a complex queries, which contain a lot of aggregation and joins operation on large tables, this makes treatment of these queries take too much time, which negatively affects the decision-making process, which is essentially an interactive process between the decision makers and the results of the analysis, Whenever Increased the time it takes to receive query results, Whenever the impact negatively on the enterprise, requiring search techniques to speed up the process of query

The increasing size and the great for data warehouses, in addition to complex queries, with the importance of time requires thinking about the means and techniques to accelerate to get the desired response from the queries required, in fact,The researchers did not overlooked this aspect, but we find great interest in it, which explains the diversity of techniques contribute to speed up access to the results in the least possible time, these techniques are called: Performance optimization techniques, they are derived from the techniques used in relational databases, and these techniques can be classified into two categories, depending on what is produced by the technique of carry and plus size because of the data redundancy or this technique be without data redundancy, the categories are (Bellatreche et al., 2007) (Abdelhédi, 2014) (Kerkad, 2013) (Bouchakri and Boukhalfa, 2009):

1. **Redundant Structures** : Techniques are working to optimize queries, but impose additional cost in storage size and in the maintenance, we find in this class :
    - Vertical Fragmentation
    - Mixed Fragmentation
    - Materialized Views
    - Index
    - Management Of Buffer

2. **no redundant structures** :Contrary the redundant technology, these techniques do not require any additional overhead or extra cost does not in storage size and not in maintenance, we find in this class:

   – Horizontal Fragmentation
   – Parallel Processing
   – Scheduling Queries.

The Figure 39 summarizes the existing performance improvement techniques

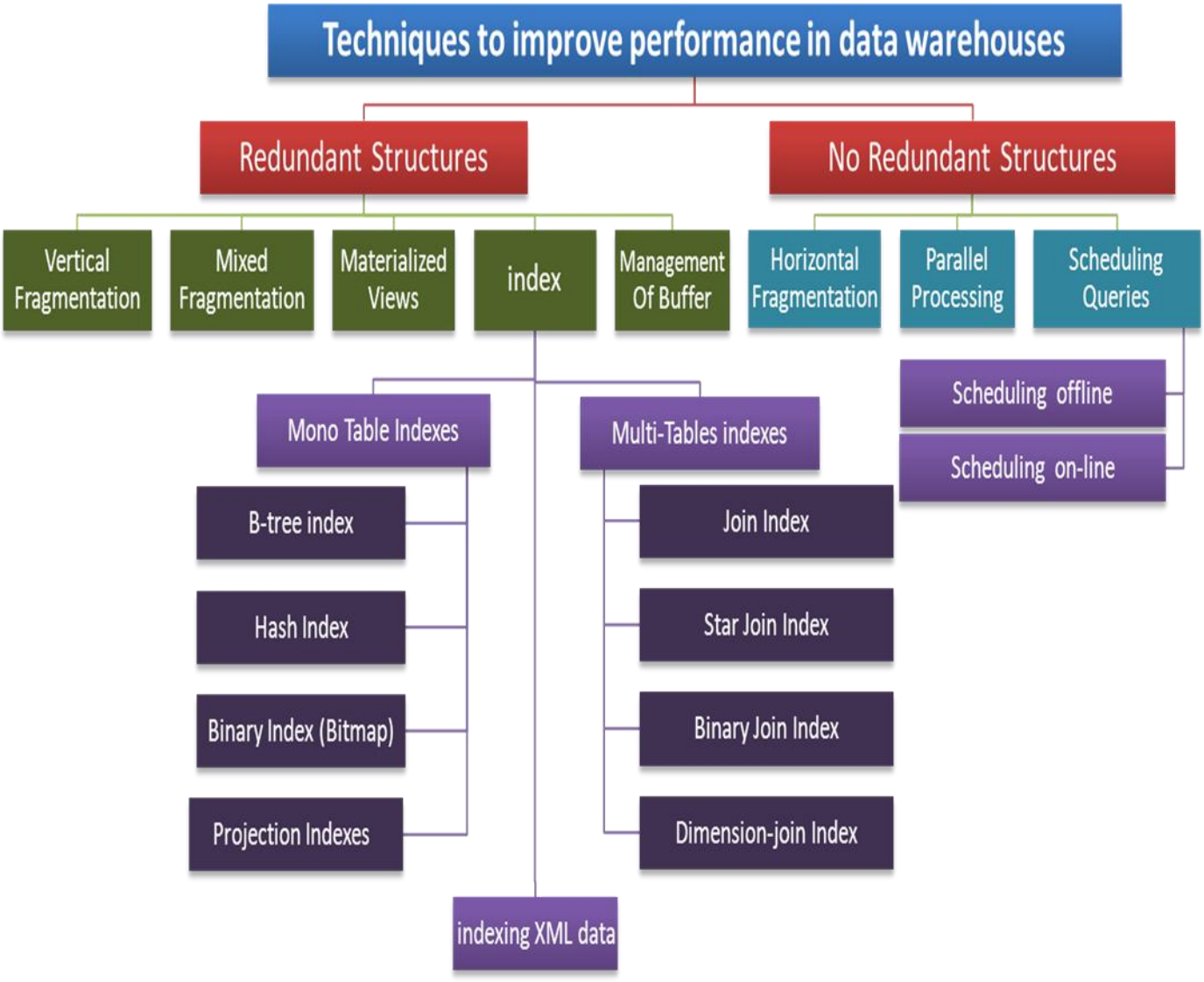

Figure 39 the different Techniques to improve performance in data warehouses

3.1 Redundant Structures

Techniques to improve performance in data warehouses or in relational databases, Improve performance in data warehouses or in relational databases, which require when working with more storage size and techniques as well as to increase it further at the cost of maintenance,This costly techniques called Redundant techniques, wherein the data is repeated, and many techniques falls under this section, are:

3.1.1 Index

Indexes is a data structure added to the database for quick access to information and data that we are looking for, through the master key, and the indicators by which is found on the physical location of the ranks of the tuples that we are looking for, indexes significantly reduce the read operations and access on the disk, especially since it came out search for properties require linking two or more tables, which contributes to accelerate the query process and improve performance. And if the indexing applied to the field or more and are all on the same table, we classify this index as Mono-TABLE, but if the gathering and linking two or more tables is called the index Multi-Tables.

From among the proposed indexing techniques in the context of traditional databases,(Boukhalfa, 2009) we can mention *the B-tree index, hash index, projection index*, and Join Index …Etc. Most of these indexes are also used in the context of relational DW. Some indexing techniques also appeared in the context of DWs such as *bitmap index, binary join index* and *Star join indexes*.. Etc., this is taken up in this part of memorandum which index techniques used in relational data bases and data warehouses.

3.1.1.1 Mono Table Indexes (Simple)

As described in the figure, there are two categories of the indexes, Simple indexes *Mono Table* ; that are defined on one or more of properties or fields from the same table, And the second category is a Multi-Tables or Join Index, that are defined on a set of properties or fields from two or more tables, Are doing the merger process between the involved tables, and in both categories, we find several types of indexes that fall under this category or that.

With regard to the Simple indexes, we find:

3.1.1.1.1 B-tree index

The default index for most commercial database systems (Boukhalfa, 2009). It is a binary tree of several levels, the lower level has a Leaf nodes, which contain the index entries with the cursor to the physical location of the record associated with it, and the elements associated with the elements of the bottom ones (children) called a level nodes, and each level consists of pages and page consists of a set of elements (the number does not exceed the specified grade of the tree) and are arranged in ascending order; The tree has a single root consists of only one element represents the top level of the tree, The search process in the index are very fast, on the whole Bayer tree is a tree data structure, allow searches, and sequential access, and input, and deletions in logarithmic time. Bayer trees is a generalization of the binary search trees, and these trees are often used in databases and in filesystems.

An example of a tree-b:

Let's start with B-Tree empty and then turn up the keys in the following order:

1 ،and then 12, then 8, 2.25, 5 and 14.28, 17, 7.52, 16, 48.68, 3, 26, 29, 53, 55.45

We want to build a B-tree of degree 5, first we take the first four elements for composition of the root, then evolve it as it is representative of the Figure 40.

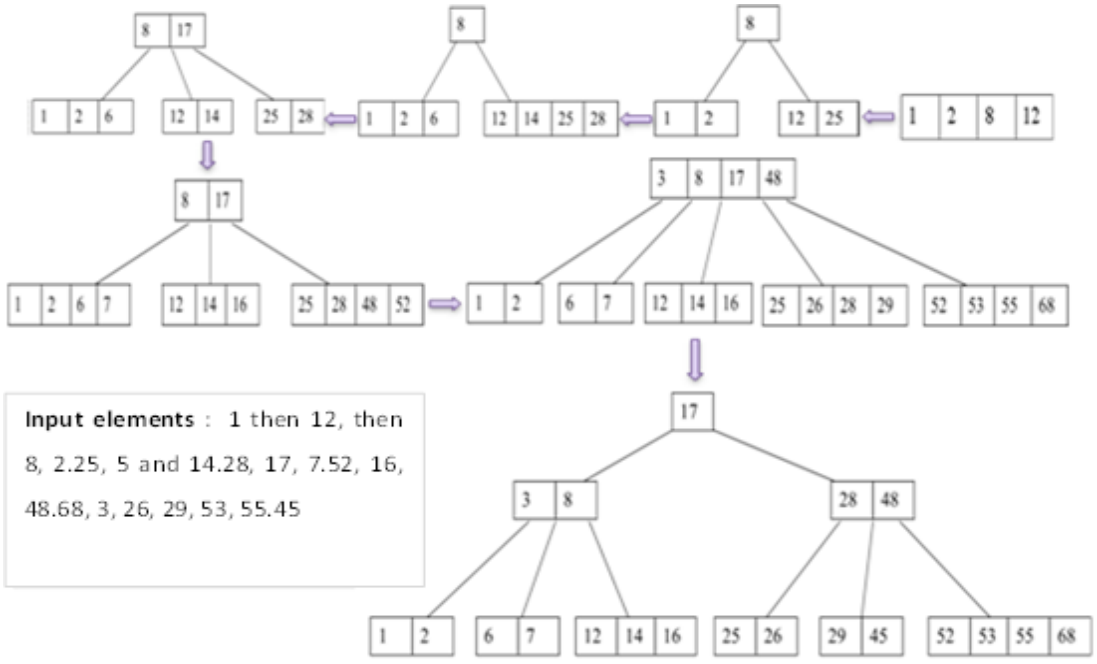

Figure 40 B-tree exemple

3.1.1.1.2 Hash index:

It is a mono index, based on a Hash Function, Based on the key element and using this function, determines the physical location of the selected item title (Figure 41), The choice of hash function is very important to ensure good performance of the Index (Figure 42).

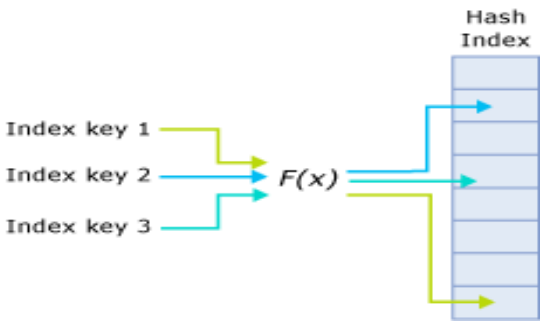

Figure 41 HASH INDEX

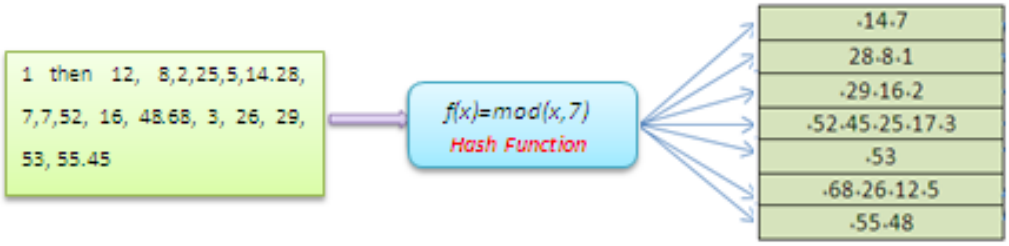

Figure 42 HASH INDEX - Exemple -

3.1.1.1.3 BINARY INDEX (Bitmap) :

Binary index is based on the use of a range of binary tables (with values 0 or 1) to record all rows from the table concerned where are stored bit Bit for each value of the property or field indexer, this index is used on the fields and attributes of confined values such as: (sex : a, y, years: limited values, ...) (Figure 43).

Persons Table T			
ID_c	Gender	Eye_Color	Other
100	Male	Broun	.....
101	Female	Black	.....
102	Male	Black	.....
103	Male	Blue	.....
104	Female	Black	.....
105	Female	Green	.....
106	Female	Green	.....
107	Male	Broun	.....
108	Female	Blue	.....
109	Female	Blue	.....

Bitmap Index : *Gendre*	
Male	Female
1	0
0	1
1	0
1	0
0	1
0	1
0	1
1	0
0	1
0	1

Bitmap Index : Eye_Color			
Broun	Black	Blue	Green
1	0	0	0
0	1	0	0
0	1	0	0
0	0	1	0
0	1	0	0
0	0	0	1
0	0	0	1
1	0	0	0
0	0	1	0
0	0	1	0

Figure 43 BINARY INDEX (Bitmap)

3.1.1.1.4 Projection Indexes :

It is a simple index (O'Neil and Quass, 1997) , The projection index is defined as one or more of the attributes or fields of the table, it saves a copy of the values of those attributes in the same order that they appear in the original table, as it represents a complete copy of the entire column of a particular attribute (Figure 44). This type of indexes is very useful to improve the condition queries with GROUP BY.

Persons Table T				
ID_c	Name	Year-B	Gender	Other
100	Ahmad	1998	Male	.....
101	Salma	1977	Female	.....
102	Amer	1980	Male	.....
103	Mahmoud	1997	Male	.....
104	Aicha	1964	Female	.....
105	Habiba	1985	Female	.....
106	Mariam	2006	Female	.....
107	Qutaiba	2009	Male	.....
108	Sirine	2014	Female	.....
109	Khadija	1951	Female	.....

Projection Index on : Name and Year-B	
Name	Year-B
Ahmad	1998
Salma	1977
Amer	1980
Mahmoud	1997
Aicha	1964
Habiba	1985
Mariam	2006
Qutaiba	2009
Sirine	2014
Khadija	1951

Figure 44 Projection Indexes

3.1.1.2 Multi-Tables indexes (JOIN)

The queries addressed to data warehouses, are complex queries, because it is often inquiry on data extracted from multiple tables and not from one table, which require linking and jointing them, to get the correct and desired result, of course, the joint process is too costly, especially in huge databases.

3.1.1.2.1 JOIN INDEX :

(Valduriez, 1987) defined Join Indices which calculates prior to the links and the join of two tables, it embodies materialized ties and relations between them through a table of two fields, Each of them represents a row identifier RID of tables of them (Figure 45).

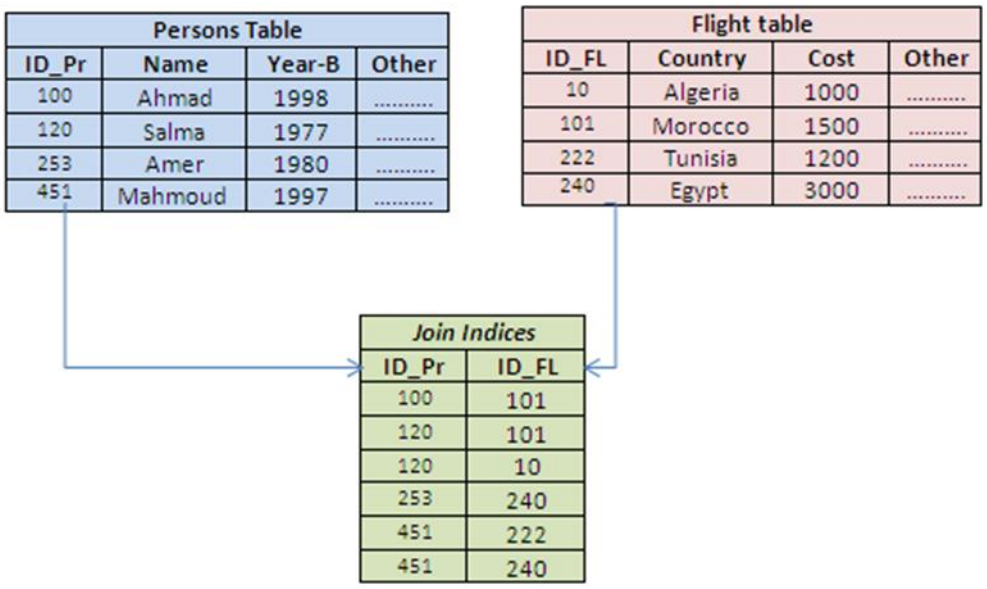

Figure 45 Join Indices

3.1.1.2.2 STAR JOIN INDEX :

In data warehouse, we find that the OLAP queries require several join process between the fact table and dimensional tables, if we apply the join Index for Valduriez, we will repeat the index with all tables two by two, so the number of indexes required is N! Where N is the number of tables, to avoid this problematic, RedBrick proposed a special join Index (Boukhalfa, 2009) ، STAR JOIN INDEX, it is more convenient to dataMarts modeled by star schema. the Star Join Index contains all the possible combinations between the ID of Fact table and foreign keys of dimension tables.

3.1.1.2.3 BINARY JOIN INDEX :

The Binary join Index Is another form of the join indexes, which combines the binary indexes and the join indexes, it has been proposed in order to pre-account the linkages between fact table and dimensional tables In data warehouses that have been modeled star scheme (O'Neil and Graefe, 1995) (O'Neil and Quass, 1997) (Boukhalfa, 2009)

.

3.1.1.2.4 DIMENSION-JOIN INDEX :

the Binary join Index is not applicable to the data warehouses modeled by snowflake scheme, because the relations in the hierarchy of dimensional tables are non-specific or pre-calculated, for this Bizarro(Bizarro and Madeira, 2001) proposed an index that takes into account queries applied to the data warehouses modeled by snowflake scheme, this index is called DIMENSION-JOIN INDEX

3.1.1.3 Techniques for indexing XML data

Native XML databases are stored XML documents, his data is queried using languages based on Path Expression.

Path expression aims to find an item or a node in the graph or XML tree by scanning the path described by the expression. And to ensure a reasonable time to deal with queries in path statements, several techniques have emerged for indexing XML documents data , this index may be in itself an XML document, and there are many studies in the field of data XML and documents indexing, we can divide the proposed indexes In studies interested in this field into 3 sections (Hammerschmidt, 2006) :

- Section I. includes approaches that index the document structure without regard to the values of the elements or attributes, call this section label: *Structural Indexes* or *Pure-Path Indexes*.

- Section II: Unlike the first, there are indexes to index the attributes and elements without looking at the path to it, called this group: *Value Indexes*.

Section III includes advanced approaches that are interested and indexing all path and values together, leading to greater speed up queries, called this group: *Hybrid Indexe* (mixed).

Before beginning to explain the three sections, we have Figure 46 ; which contains the XML document, with the DOM tree related, where we will need to explain the types of indexes

```
1 <site>
2  <regions>
3      <asia>
4          <item id=" item1">
5           <location>Singapur</location>
6            <quantity>2</quantity>
7            <name>512 MB USB Stick</name>
8             <payment>Money order</payment>
9            <payment>Cash</payment>
10         </item>
11    </asia>
12    <europe>
13        <item id=" item3">
14          <location>Hamburg</location>
15          <quantity>1</quantity>
16          <name>Beuys Sculpture </name>
17        </item>
18        <item id=" item4">
19          <location>Paris</location>
20          <quantity>2</quantity>
21          <name>Louvre Tickets</name>
22          <payment>Cash</payment>
23        </item>
24    </europe>
25  </regions>
26  <people>
27      <person id="person0">
28        <name>Huei Demke</name>
29        <address>
30           <street>95 Grinter St</street>
31           <city>Luebeck</city>
32            <country>Germany</country>
33        </address>
34     </person>
35        <person id="person1">
36         <name>Daishiro Juric</name>
37         <address>
38            <street>5 Pinet St</street>
39            <city>Athens</city>
40             <country>Greece</country>
41        </address>
42    </person>
43  </people>
44 </site>
```

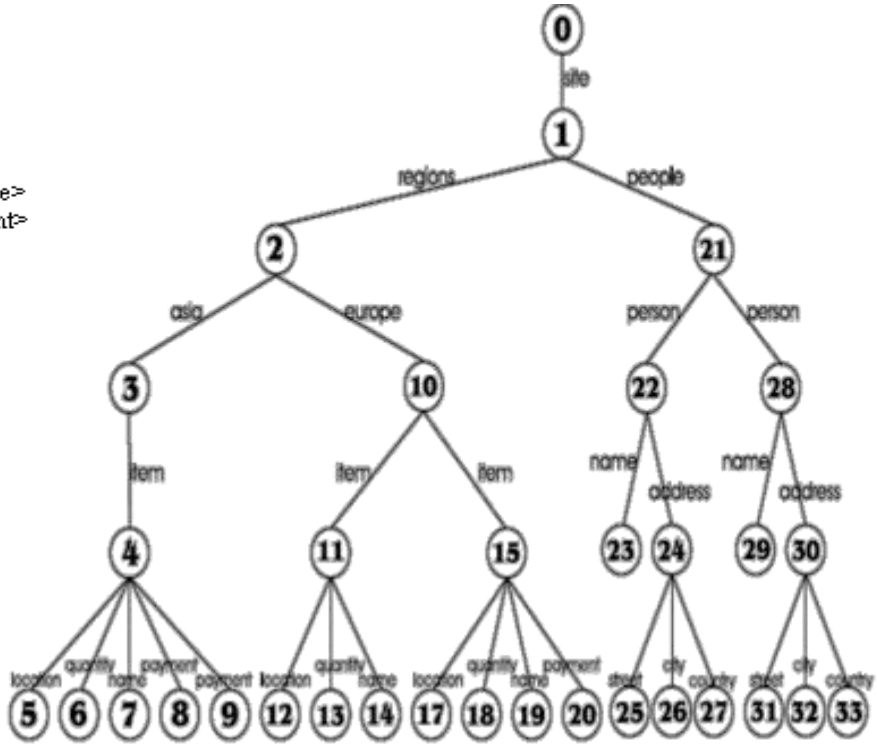

Figure 46 XML document with the DOM tree related to it

3.1.1.3.1 Structural Indexes

Structural indexes reflect the structure of the XML data with its element labels. The values of the elements are not kept in the index structure. Therefore, only pure path expressions are supported. From these indexes we mention the following:

3.1.1.3.1.1 Strong DataGuide:

DataGuide is a summary structure To represent semi-structured data and data XML, It was proposed by (Goldman and Widom, 1997) The index structure works on the description of all the nodes that have the same label, the definition of DataGuide is based on the target groups. The target group of the path is a collection of all the objects (nodes) with the ability to access and which chose that path (Yeh et al., 2004)., Figure 47 illustrates the working principle of the data directory.

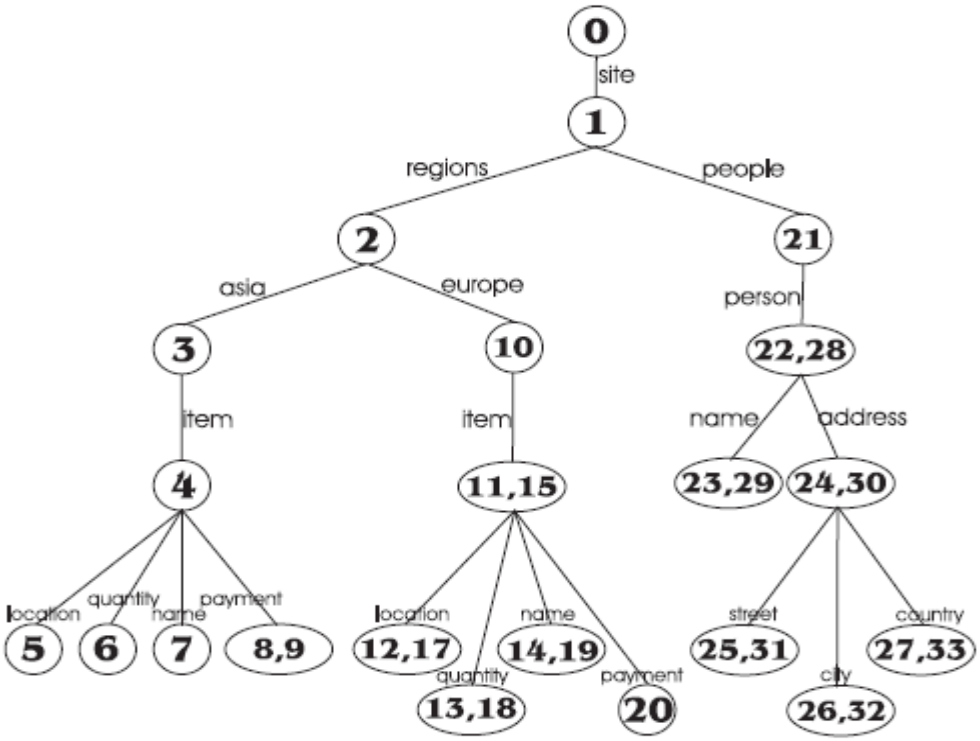

Figure 47 Strong DataGuide

(DataGuide) requires a long time to construction and enormous size may exceed the size of the data (Yeh et al., 2004).

3.1.1.3.1.2 1-Index:

This index works on assembling nodes that have the same path in the graph data XML, this is done through the concept of Bisimulation, ‹(Aouiche, 2005) ‹in Figure 48 illustrate the difference between the index 1-Index (diagram b) and the data directory (diagram c) The (diagram a) represents the XML data tree (Yeh et al., 2004) (Milo and Suciu, 1999)

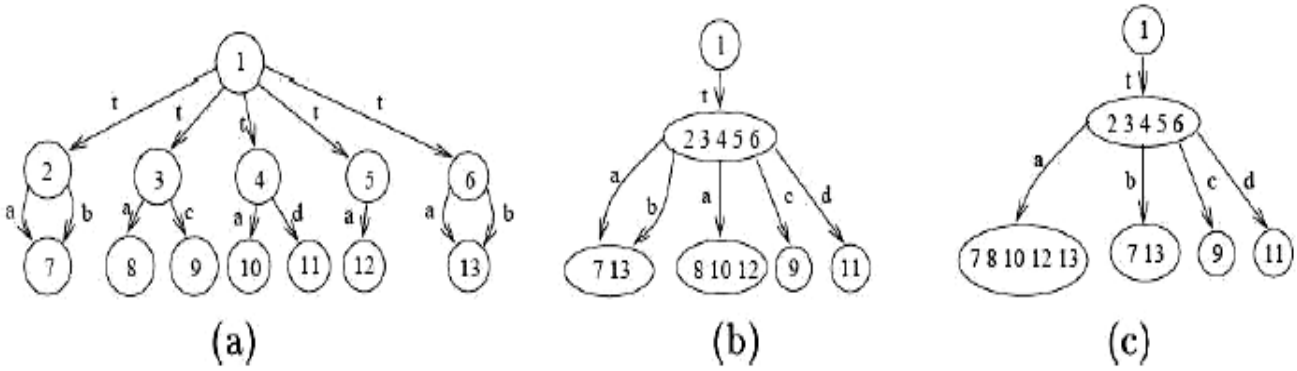

Figure 48 1-Index

3.1.1.3.1.3 2-Index:

The 2-Index (Milo and Suciu, 1999) supports absolute and relative path expressions between nodes that are connected by parent-child relationships. The 2-Index is also based on extents that contain nodes that are reachable by similar path expressions, Figure 49 represents 2-Index of the sample mentioned earlier.

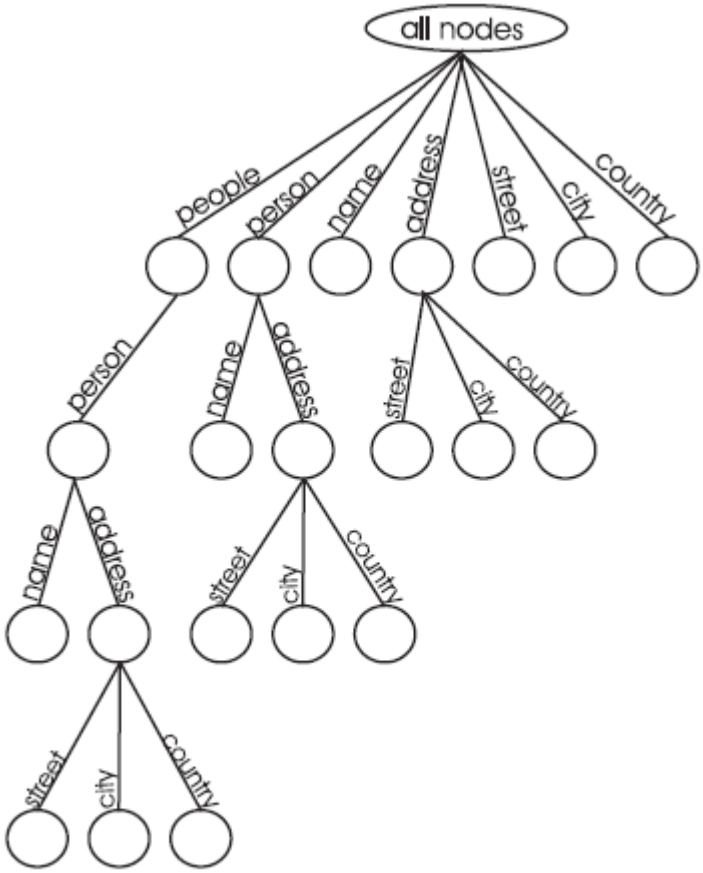

Figure 49 2-Index

3.1.1.3.1.4 T-Index:

It is Abbreviation for Template-based Indexing, proposed by (Milo and Suciu, 1999)، The idea is not to build indexes on all tracks, but in the specified path templates only. The T-Index is a generalization and specialization of the 1-Index and the 2-Index. It is a generalization because the 1-Index and the 2-Index are special cases of the T-Index. This means, that every 1-Index and 2-Index is a T-Index, too. On the other hand, the T-Index is a specialization because it is tailored to answer specific queries whereas the 1-Index and the 2-Index can be used to process any path expressions (Hammerschmidt, 2006). Figure 50 shows the T-Index on the previous sample.

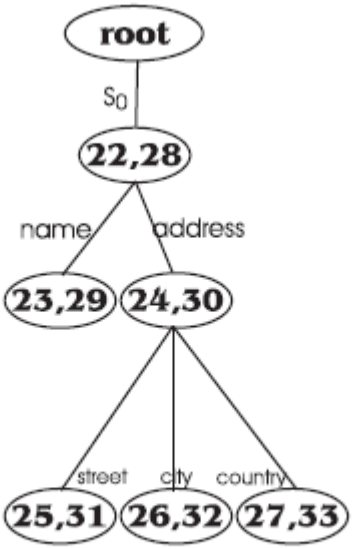

Figure 50 T-Index

3.1.1.3.1.5 APEX :

In General, the index stores all paths from the root element, the size of these structures is linked to the depth of the data, Whenever; the depth of the data Increased ,the size of the structure Increased, which negatively affects performance.

APEX is an index to adaptive path to the data XML (Chung et al., 2002) (Aouiche, 2005), It provides a compromise between size and efficiency. Instead of indexing all the paths from the root, APEX index, in the structure, paths that are frequently used, and on another structure, it keeps the summits of source structure. Which greatly reduces the size of exploited. In the Figure 51, the representation of an index APEX, this consists of two structures:

- Hash tree H_{APEX}, representing the inside Path to a node In the graph G_{APEX} .
- the graph G_{APEX} ,it is a structure to save the XML data

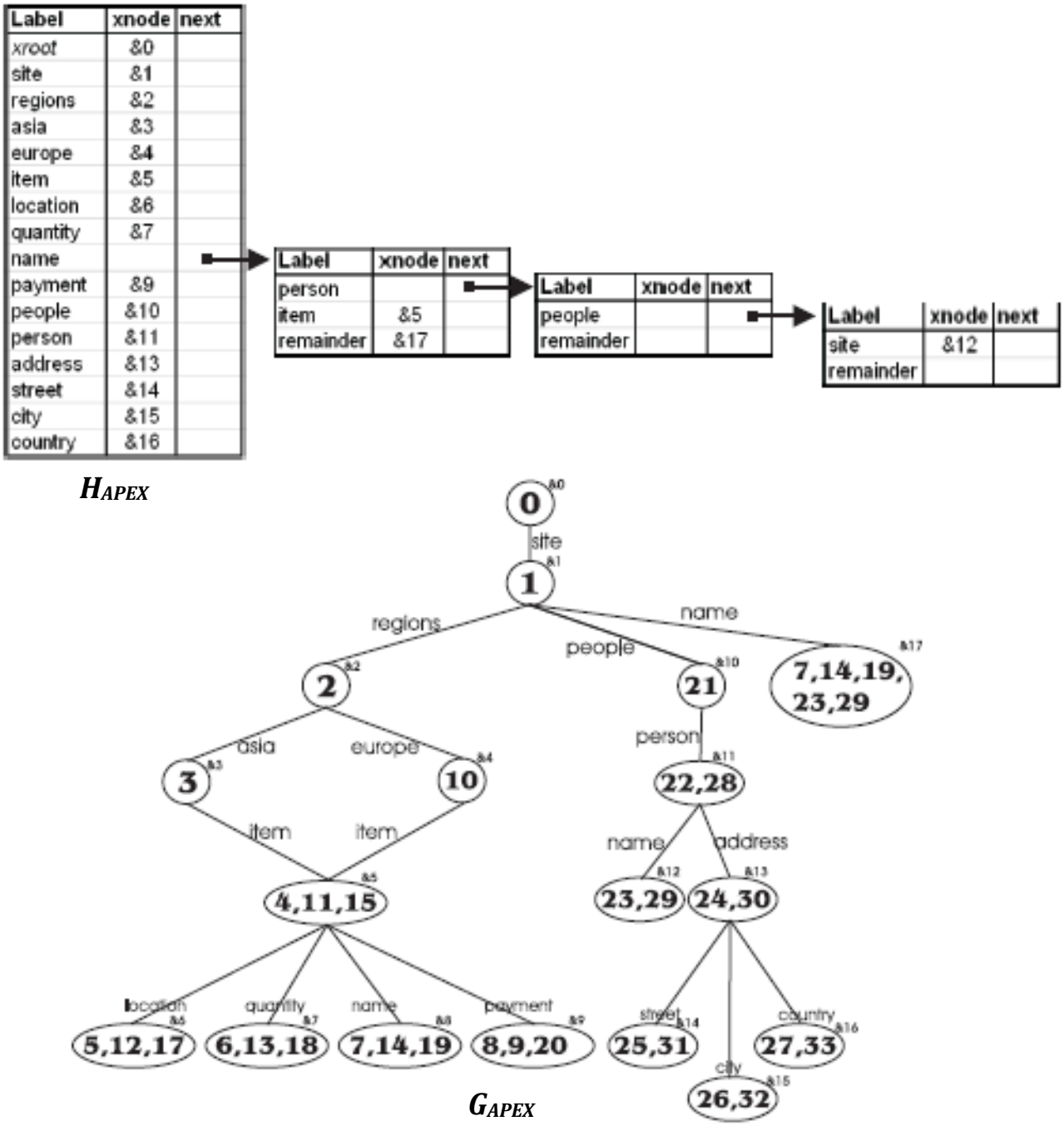

Figure 51 APEX index

3.1.1.3.1.6 Numbering Schemes and Tree Signatures :

Numbering Schemes (Grust, 2002) (Haifeng et al., 2003) map each element of the XML data to one or more numbers that are mostly determined by post/preorder XML-tree traversing algorithms. The numbers are used for a faster retrieval of relationships between elements, .The work) Kha et al., 2001(proposes a numbering schema that is optimized for updates leading to

less number recalculation when the XML data is modified. In general, numbering schemes are not selective and do not cover key-queries. On the other hand, they are comparable to a structural summary because the structure is implicitly expressed in the numbers ..

Approaches that use tree signatures to process XML queries (Zezula et al., 2003) can be compared to the numbering schemes with the difference that the function creation signatures from XML data is not bijective. Therefore, signatures may lead to wrong hits that have to be filtered in a further step (Hammerschmidt, 2006).

3.1.1.3.1.7 Structural Summaries:

Structural summaries are the main data structure in further works ‹(Hammerschmidt, 2006) ‹including the Forward-and-Backward-Index (Abiteboul et al., 2000) ‹the D(K)-Index (Chen et al., 2003) ‹Covering Indexes (Kaushik et al., 2002)‹ and HOPI (Schenkel et al., 2004) ‹ a so-called connection index . HOPI is tailored for queries with long paths especially with wildcards and the XPath axes descendant and ancestor

which cannot be supported efficiently by several structural summaries.

All structural summaries have in common that they require navigation with several steps in their internal data structure when evaluating a query. A major disadvantage is that structural summaries ignore the values of elements and attributes, so that path expressions with value conditions cannot be executed efficiently‹ .

3.1.1.3.2 Value Indexes

3.1.1.3.2.1 Inverted Lists:

Some XML index approaches only index the value of elements without paying attention to the full leading path. Those indexes can only support very restricted XPath queries like //*[.='Dan']. Because any structural conditions are ignored the relevance of these approaches is questionable (Hammerschmidt, 2006) . Accelerating key value queries for XML data by information retrieval techniques is one popular proposal that reuses known implementations like Inverted Lists or Tries (Scholer et al., 2002) (Kaushik et al., 2004) ‹An inverted list enumerates all values appearing in the XML data in a search structure and references the position of their appearance. Each element of inverted list elements has at least one reference. Inverted list takes time logarithmic to recover any element if the value of the item is known. In Figure 52, is displayed an Inverted list for our sample list above

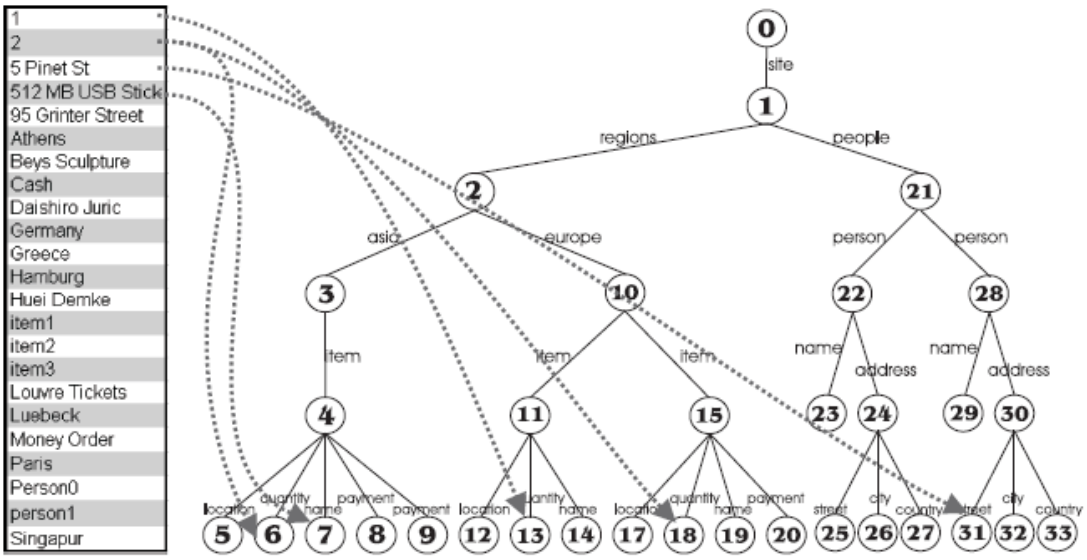

Figure 52 Inverted Lists

3.1.1.3.2.2 Lore ‹Lore Value Index:

The Value Index of the Lore Database Management System (McHugh et al., 1997) is a selective inverted list for element values. Selective means that the label of an element is reflected and that a set of labels to be indexed can be defined by a database administrator..

3.1.1.3.2.3 SEQL Index:

A more sophisticated approach is presented by SEQL (Search Engine Query Language) (Naughton et al., 2001) managing an additional inverted list for the element labels. Nevertheless, because SEQL regards XML data as a text document and not as a tree of nodes, it returns text positions instead of nodes, so that further navigation is uncomfortable.

3.1.1.3.3 Hybrid Indexes

(Hammerschmidt, 2006) think that XML indexes that support both structural queries and value queries are the most relevant indexes because they cover most XML query capabilities.
Here we list some of the work that had focused on hybrid indexes.

3.1.1.3.3.1 Structural Summary plus Inverted List:

An obvious approach to support both structural and value queries is to combine two separate data structures like a structural summary and an inverted list. This is done in the work of (Kaushik et al., 2004) (Halverson et al., 2003) (Barg and Wong, 2003), A query is separated into the structural part processed by the structural summary and the value part that is executed by use of an inverted list . Both data structures return relevant nodes that have to be intersected at runtime to retrieve the final result set. In general, this intersection operation may produce enormous costs for removing wrong hits.

3.1.1.3.3.2 Index ViST:

With the Virtual Suffix Tree (ViST) (Wang et al., 2003) introduce an approach that encodes and represents XML data and path expressions as structure-encoded sequences. XML data is represented by the preorder sequence of its tree structure produced by a depth-first traversal of the XML data. The value of elements and attributes and the labels of all elements are combined to one large sequence. Therefore, ViST is comparable to a numbering schema. Since isomorphic trees may produce different preorder sequences an order among sibling nodes is enforced using the lexicographic order of the labels.

3.1.1.3.3.3 Index FABRIC:

In FABRIC Index XML data is indexed) Cooper et al., 2001(Using encryption paths, from the root to the leaves nodes, and these operations are done by designators that operate on the data path encryption in the form of special characters or literal strings. Accordingly, it is one indicator is set for each tag. Symbols or obtained from the encoding characters are inserted into the PATRICIA tree where treated as simple characters،

Index FABRIC allows efficient management of a large number of complex key, Therefore, it can be used to search through complex paths of data XML. A Dictionary of designators is put in order to ensure consistency and congruence between the indicators and the names of tags, This dictionary is created during the traffic on the source data to create the index, whenever, it add a new item, a new designator is added automatically, The names of the tags in the query are converted to pointers using the dictionary key to search through the index structure, Search in the index is determined by the search for a string keys, which starts from the root node and then compares the series with labels in the structure, update operations can be treated effectively with FABRIC index. Addition a node in the index structure is by adding one node or by adding a limit at the current node.

. In the Figure 53 mention an example of index FABRIC on the previous sample, where the table (a) represents a Dictionary of the indicators, and the table (b) contains paths encryption

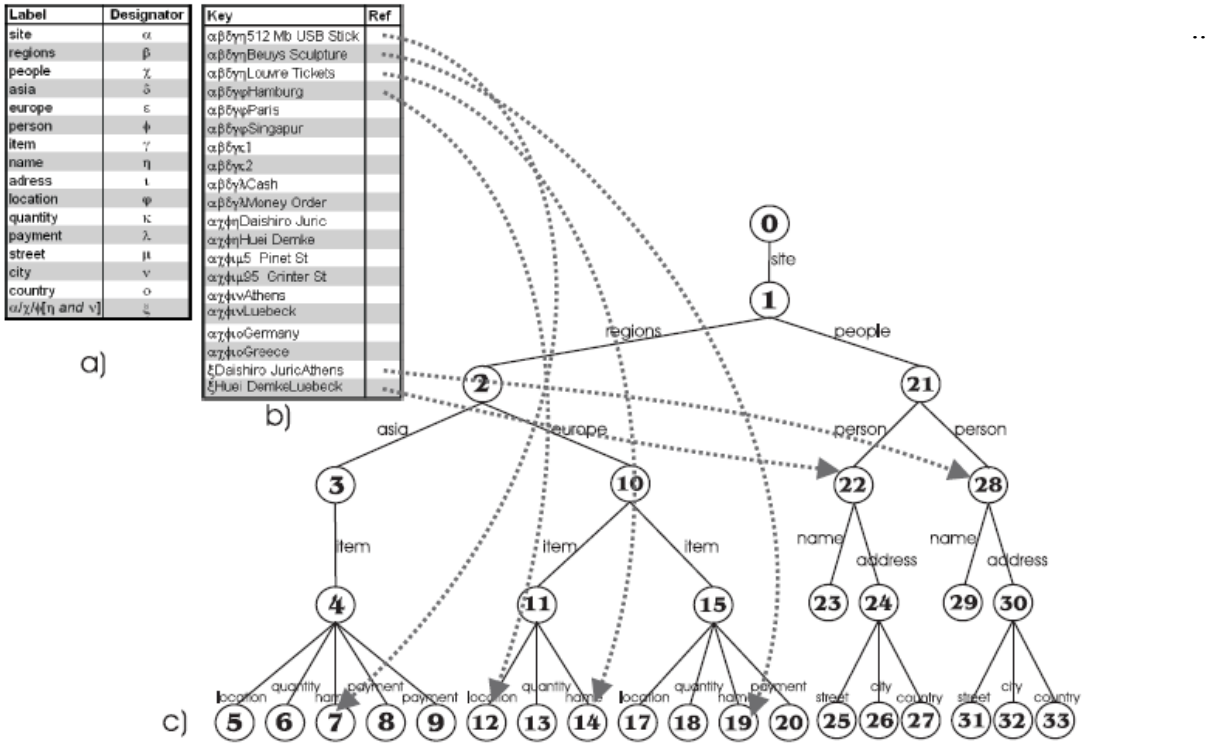

Figure 53 FABRIC index

3.1.2 Materialized view

Materialized view is the table that contains the result of executing a particular query, It improves the execution time of queries, By pre-calculation of operations costly that its implementation takes a long time, such as joins and aggregations, where query results are saved on the Storage volume (operations which we call materialization), Therefore, the execution of these queries only requires access to these Materialized view preserved, rather than access to the source data, Which greatly reduces processing time and costs, because the size of the Materialized view much less than the size of the source tables, these view is marked to simplify source tables structure, and facilitates re-use of queries process repeatedly, and one of the important things which contribute Materialized view are that it helps to protect the original data and secure the access.

However, this technique has some disadvantages related to the cost of maintenance, it must be updated at each update and change in the source data, as it will occupy an additional space in the storage units.

Similarly, in the context of an XML data warehouse, it can be the Materialization of Views in the form of an XML document, the Figure 54 shows an example of a Materialized view XML. This view is made up of cells (Cell element). Each cell is characterized by a number of dimensions (Dimension elements) and measurements marked on these dimensions (Fact element).

```
<viewXML>
    <view id='mv'>
        <Cell>
            <fact name='sold' value ='1500' />
            <dimension name='pord_weight' value='10' />
            <dimension name='cust_gender' value='F' />
        </Cell>
        <Cell>
            <fact name='sold' value ='800' />
            <dimension name='pord_weight' value='50' />
            <dimension name='cust_gender' value='F' />
        </Cell>
        <Cell>
            <fact name='sold' value ='30' />
            <dimension name='pord_weight' value='10' />
            <dimension name='cust_gender' value='M' />
        </Cell>
        <Cell>
            <fact name='sold' value ='100' \>
            <dimension name='pord_weight' value='50' />
            <dimension name='cust_gender' value='M' />
        </Cell>
    </view>
</viewXML>
```

Figure 54 Exemple of an XML Materialized view

There are many research and work focused on addressing the problems of Materialized view, we can distinguish two axes are important in such research (Boukhalfa, 2009) (Aouiche, 2005):

- The issue of increasing maintenance of Materialized view, when the source data is updated.
- A matter of choice and determine the view required to Materialize (Gupta, 1997)، there are three choices: first Materialized view for each queries, this is expensive, very much, the second not Materialize any of the queries, and is direct access to the source data at a time, the third and is an account and the Materialize of the most common and widely used only visuals.

3.1.3 Management Of Buffer

Buffer contains databases copies of some of the data on secondary memory (Kerkad, 2013) (Garcia-Molina, 2008) These copies are stored in the form of pages and the results of data processing by DBMS, Put the data in the cache is characterized by being able to reuse them in other queries which limits many access operations In secondary memory where permanently Data is stored (HDD, Flash, Cloud ...). Re-use of this data can increase system performance by the physical characteristics of central memory which makes the cost of reading and writing small compared with the time of arrival of the data on the disk. Tuning buffer is a series of treatment to adjust the various parameters of the buffer required to maintain the good performance of the system.

3.1.4 Fragmentation

Among the most important techniques used to improve the performance of the relational databases and data warehouse, which received a great deal of interest and study in a lot of research are: *Fragmentation,* which means fragmentation and division of the warehouse or tables into several sections and parts,

***The* fragmentation** is a division of a set of data to multiple partitions, called fragments, where the combination of Fragments resulting in the entire source data without any loss or add information, (Mahboubi, 2008).

So Fragmentation is to divide a set of data, In context and relational framework, this group is the tables (relations) In the context of XML data warehouse, these groups represented in the document or documents XML, There are three types of Fragmentation and are:

- Vertical fragmentation
- Horizontal fragmentation
- Hybrid fragmentation

3.1.4.1 Vertical Fragmentation

Vertical fragmentation is to divide the data set on the basis of the fields or attributes into several sections called these sections: vertical fragments، the formation of these fragments by projections of one or more of the characteristics that are frequently used by the queries applied to the initial set of properties, fragmentation vertical enhance projection processing queries on the characteristics used in the retail process, limiting the number of fragments that are accessed (Figure 55).

we have a relational table T (data set) - talking here as an example in the context of databases relational -, This table consists of a group A of the fields - attributes or features - from T, the vertical fragmentation of the table T, is to find a set of parts (fragments) F associated with each other by table Key, as:

- $F = \{F_1 \dots F_k\}$، And k is the number of parts Constituent the fragmentation.
- $\forall F_i\ (i = 1..k), F_i = \prod_{A_i}(T)$ as Ai are the set of properties constituent the part of Fi And content in A set of properties A which: $Ai \subset A$ ، $\prod$ is Projection Operator.
- The property Key is added to all parts Fi
- The intersection of all fragments gives us the key property ، $F_1 \cap \dots \cap F_k = Key$
- Inclusion of all the fragments resulting the origin table T: $T = F_1 \bowtie \dots \bowtie F_k$ as $\bowtie$ is Join Operator

Customers Table T

ID_c	Name	Year-B	Gender	Level	Place-B	Debt
100	Ahmad	1998	Male	Secondary	Laghouat	1000
101	Salma	1977	Female	Medium	M'sila	5400
102	Amer	1980	Male	Without	Setif	5450
103	Mahmoud	1997	Male	Medium	M'sila	9800
104	Aicha	1964	Female	University	M'sila	2150
105	Habiba	1985	Female	Secondary	Djelfa	0
106	Mariam	2006	Female	Primary	Algeria	1000
107	Qutaiba	2009	Male	Without	M'sila	1500
108	Sirine	2014	Female	Without	Algeria	1654
109	Khadija	1951	Female	Secondary	Setif	1200

Vertical fragmentation

Fragment F_1

ID_c	Name	Gender
100	Ahmad	Male
101	Salma	Female
102	Amer	Male
103	Mahmoud	Male
104	Aicha	Female
105	Habiba	Female
106	Mariam	Female
107	Qutaiba	Male
108	Sirine	Female
109	Khadija	Female

Fragment F2

ID_c	Year-B	Place-B	Level
100	1998	Laghouat	Secondary
101	1977	M'sila	Medium
102	1980	Setif	Without
103	1997	M'sila	Medium
104	1964	M'sila	University
105	1985	Djelfa	Secondary
106	2006	Algeria	Primary
107	2009	M'sila	Without
108	2014	Algeria	Without
109	1951	Setif	Secondary

Fragment F_3

ID_c	Debt
100	1000
101	5400
102	5450
103	9800
104	2150
105	0
106	1000
107	1500
108	1654
109	1200

Figure 55 Vertical Fragmentation exemple

Vertical fragmentation is one of the techniques that improve performance in data warehouses and databases, it is a redundant techniques because fields and properties of key are repeated in every part of the resulting portions, which imposes the need for additional storage space.

3.2 No Redundant Structures

3.2.1.1 Horizontal Fragmentation

If the partition in the vertical fragmentation according to fields or attributes, without paying attention to the rows and records, the horizontal fragmentation on the contrary, quite, as they are split Objects (table, view, index) into groups and parts, each part representing a subset of rows and records in the parent table (Figure 56), these parts are called: horizontal fragments which are obtained by conducting a Restriction Operation on the records and rows of the parent data set, Each horizontal fragment meets the condition of the predicate (A combination of predicates with Boolean operators "and, $\wedge$" _ "or,$\vee$"), Completely Reconstruction of this horizontal fragments obtained by the process of Union between them. Horizontal fragmentation facilitate the treatment of restriction queries by reducing unnecessary access to data,

Even horizontal fragmentation for a table T to be valid, it must meet the following three rules: completeness, separation and reconstruction, and this is to maintain the cohesion of fragmented objects (table, view, index) (Boukhalfa, 2009):

1- **Completion**: we mean that the table will be divided entirety without leaving any Case or record without putting it in a part of the parts, Completion ensures no loss of any information from the table. If **T** is a table, T_1, T_2, ..., the $\mathbf{T_n}$ is the horizontal fragments obtained after fragmentation, and **t** is a record from the table, then:

$$\forall t \epsilon T \Rightarrow \exists T_i \in (T_1, T_2, .., T_n) \wedge t \in T_i$$

2- **Disjunction**: We mean that a record **t** cannot be found in more than only one part T_k,

$$t \epsilon T_k \Rightarrow \nexists T_i \in \left((T_1, T_2, .., T_n) - T_k\right) \wedge t \in T_i$$

3- **Reconstruction**: We mean that it can rebuild the original table T, by combining all the parts T_i, this ensures that the fragmentation operation is a reverse process, This so don't miss the original objects or tables, if the OP was the process by which combine the parts and rebuild it:

$$T = OP(T_1, T_2, .., T_n)$$

There are two types of horizontal fragmentation:

- Primary Fragmentation
- Derived Fragmentation.

3.2.1.1.1 Primary Fragmentation

Primary fragmentation for a table T, is done by the application of operations and fragmentation conditions on records and rows of the table itself, it can select parts on several levels, the maximum levels of Division varies by DBMS (Figure 57).

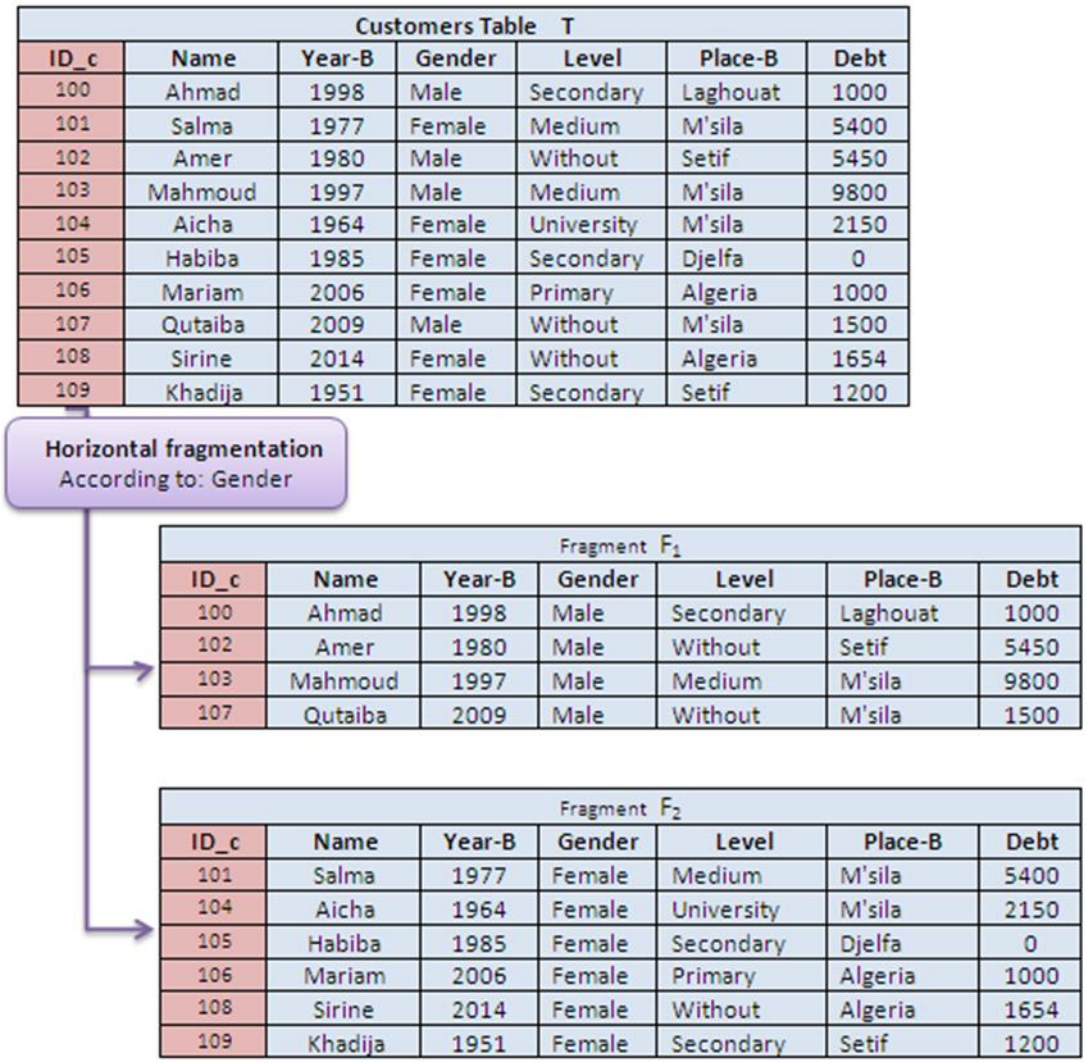

Figure 56 Horizontal Fragmentation exemple

3.2.1.1.2 Derived Fragmentation

Derived fragmentation of the table T, is done by the application of operations and fragmentation conditions on records and rows of the table T1 associated with the table T, and applied to the facts table.

Example: We have the data warehouse in star model, where. Facts table is: sales, and dimension tables: Customers table. The following figure shows the primary fragmentation (on a dimension table: customers, by property sex, so the resulting two parts: male customers and female customers), and fragmentation derived (the facts table: sales, based on the customers table) (Figure 58)

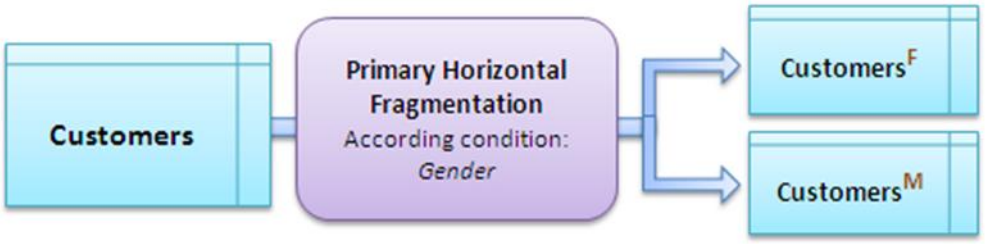

Figure 57 Primary Fragmentation

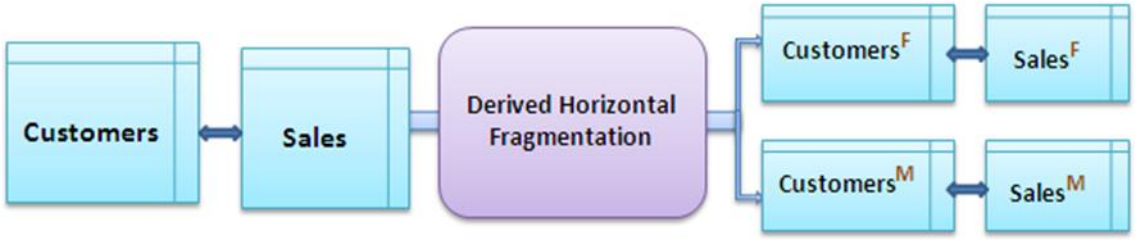

Figure 58 Derived Fragmentation

3.2.1.2 Mixed Fragmentation

Mixed fragmentation involves applying a horizontal fragmentation on the table T, followed by a vertical fragmentation applied to each horizontal fragment, or vice versa.

3.2.1.3 Fragmentation of XML data

In this section, we present some works in the context of the fragmentation of XML data, typically the data are stored in an XML document or a collection of XML documents. The proposed methods fragment the XML data into a set of XML documents called fragments. Their common goal is to improve query performance by reducing processing time. Among these studies we cite :

There are many approaches in the literature that present definitions of XML fragments (Andrade et al., 2006; Bremer and Gertz, 2003; Kling et al., 2010) and XML approaches for fragmentation design in general (Bremer and Gertz, 2003; Ma and Schewe, 2003; Pagnamenta, 2005). However, regarding the analysis stage of the fragmentation design, there is no work that details the criteria that need to be taken into consideration in that stage to generate a good fragmentation design (a good fragmentation design in one that improves query performance for the most frequent application queries). Thus, there lacks a consistent decision-making method to assess which type of fragmentation is more applicable in each scenario, causing fragmentation to be ad-hoc, typically based on the designers' experience.

(Ma and Schewe, 2003) emphasize the importance of considering the frequent queries in the definition of the fragments. Moreover, in their work, they describe heuristics for horizontal XML fragmentation. The heuristics are based on a cost model, where the biggest offender of horizontal XML fragmentation is the transport time of pieces of the query answer between nodes in the network. However, (Ma and Schewe, 2003) do not provide experimental results to evaluate the efficiency of the proposed heuristics.

(Bonifati and Cuzzocrea, 2007) describe a structure-driven fragmentation methodology. That is, issues such as size, width and depth of the XML subtrees are considered in the fragmentation process. The authors propose a set of heuristics for XML fragmentation, called SimpleX. These heuristics seek to determine the maximum values for the variables size, width and depth of the subtrees, and fragmentation is done so not to overcome these values.

(Kurita et al., 2007) propose a method for query processing over distributed XML databases. The method focuses on XML data fragmentation and dynamic relocation of fragments on the network nodes. The authors consider that for an efficient large-scale XML query processing system, we must follow four steps: data partitioning, data distribution, distributed query processing and dynamic data reallocation. However, their work is restricted to vertical fragmentation only. During the fragmentation, the authors do consider the frequent queries, but the fragment size instead. Their goal is to obtain homogeneous fragment sizes.

(Mahboubi and Darmont, 2009b) propose to apply horizontal XML fragmentation over a data warehouse. They explore two methods for horizontal fragmentation: predicate construction and affinity based fragmentation. The authors mention that several studies suggest that derived horizontal fragmentation is the best way to shred XML data. However, according to them, this statement does not necessarily apply to the XML data warehouse architecture. As a conclusion of their experiment, the authors present fragmentation affinity attributes as being the best way to shred an XML data warehouse.

(Kling et al., 2011) focus on exploring the distribution in the context of XML database systems as a way to solve the problem of effectiveness and efficiency in accessing large-scale XML data. Their proposed solution addresses two problems of XML query processing in distributed environments: localization, which is the conversion of a single query execution plan into several ones, so they can be executed in a distributed environment; and pruning, which eliminates parts of the execution plan that not contribute to the query result. In other words, only the fragments that contain relevant data to the query would be accessed. Specifically, for horizontal fragmentation, the authors use the same technique they apply in the relational model. That is, the fragmentation is obtained by composing minterms extracted from the frequent queries.

In summary, there are several different approaches to fragment documents horizontally, vertically or in a hybrid way (Bonifati and Cuzzocrea, 2007; Boukraâ et al., 2011, 2006b; Braganholo and Mattoso, 2014; Darmont, 2006; Kechar and Bahloul, 2014; Kling et al., 2011, 2010; Koong et al., 2015; Mahboubi and Darmont, 2009b, 2007; Ma and Schewe, 2003; Schroeder et al., 2012; Slimani et al., 2007). However, none of them focuses on the analysis phase, i.e. they do not give any hints or suggestions to help the designer to choose a horizontal or vertical fragmentation, for example. All related work is focused on phases (i) extraction of relevant data and phase (ii) fragmentation itself, lacking a direction to the analysis phase.

3.3 Conclusion:

Improve the performance of databases or data warehouses, generally, is using several methods, including techniques require a repeat of the data, we call it "Redundant techniques", such as vertical fragmentation and indexes, the second category of ways of improving the performance does not require redundancy is called "No Redundant techniques", the most important is horizontal fragmentation, the same thing for complex data warehouses XML, we focused in Our thesis on two basic techniques " fragmentation and indexing", due to the focus of the majority of research on them.

CHAPTER 4

XHaFrag

Improving Performance

Of XML Data Warehouse

By Hash Fragmentation

(Contribution)

Summary:

In this chapter, we will propose a new approach to improve the performance of complex data warehouses XML, using the technique of horizontal Fragmentation by hash function, I called this approach XHaFrag..

4) Chapter 4 :Improving performance of XML Data Warehouse by Hash Fragmentation:

The interest in technologies XML, especially with regard to data warehouses, is increasing day after day, due to the newness of this technology and the emergence of its efficiency and its important role in the exchange of information throughout the world, this is what we have pointed out in previous chapters of this thesis, however, the research in the field of improving the performance of complex data warehouses XML is still insufficient, that is why we have entered this area and try to contribute in it.

In this chapter we will propose a new approach to improve the performance of complex data warehouses XML, by horizontal Fragmentation, using hash functions.

4.1 Section I : XHaFrag Approch:

Most of the studies and research on the issue (improve the performance of data warehouses) whether relational or complex data XML, presented approaches based on Evolutionary Algorithm or Heuristics, such as genetic algorithm, , K-Means, Ant Colony Algorithm ... and others, (Andrade et al., 2006; Bonifati and Cuzzocrea, 2007; Boukraâ et al., 2011; Braganholo and Mattoso, 2014; da Silva et al., 2013; Kechar and Bahloul, 2014; Mahboubi, 2008b; Mahboubi and Darmont, 2009b) but in our approach, I suggested fragmentation of complex data warehouse, by primary horizontal fragmentation, using a simple method, known in the field of indexing files and data structure, this method based on the *hash function*, which divide a large set of data into several groups, in order to facilitate the search process and the storage.

The approach I developed is based on the Hash Fragmentation, and I have called it : *XHaFrag*, Shorthand for its meaning "XML Hash Fragmentation", and in the following points I will explain in detail how does it work and the principles on which it is based (see Figure 59):

1. **We fragment only the facts document**, not the dimensions documents, this is because the facts document often represents the largest bloc of the data warehouse, and is the most renewed and updated, unlike dimensions documents, which are usually limited in size, and its update happens rarely. The greatest burden on queries comes from the facts document, and the fragmentation of the latter reduces the burden and improves performance.

2. **Choice of hash function xF (x)** : It is the function accordance to which the fragmentation works, and it determines the number of resulting fragments, it also works only on the simple fields and properties (numbers or text strings), such as sequential key or totals, Among the most common hash functions, we find the correct remainder function *Modulo*, the function that we would use in our experiments to analyze the performance of the proposed approach xF (x) = Mod (x).

3. **Select the field or attribute**: - fields or properties — which we would use in the hash function, as mentioned in the previous point, the fields used must be simple. And it does not require the use of the table key only.

4. After selecting the field and choosing the function, we fragment the facts document, which will produce a set of files - the number of resulting files controlled by the function used -, these files (parts) is mostly identical in size. It will also be completely separate from each other, and relationships and links with documents dimensions remain unchanged.

5. After applying the fragmentation, and the production of new documents, we load them into the warehouse instead of the original facts document, and apply queries on them.

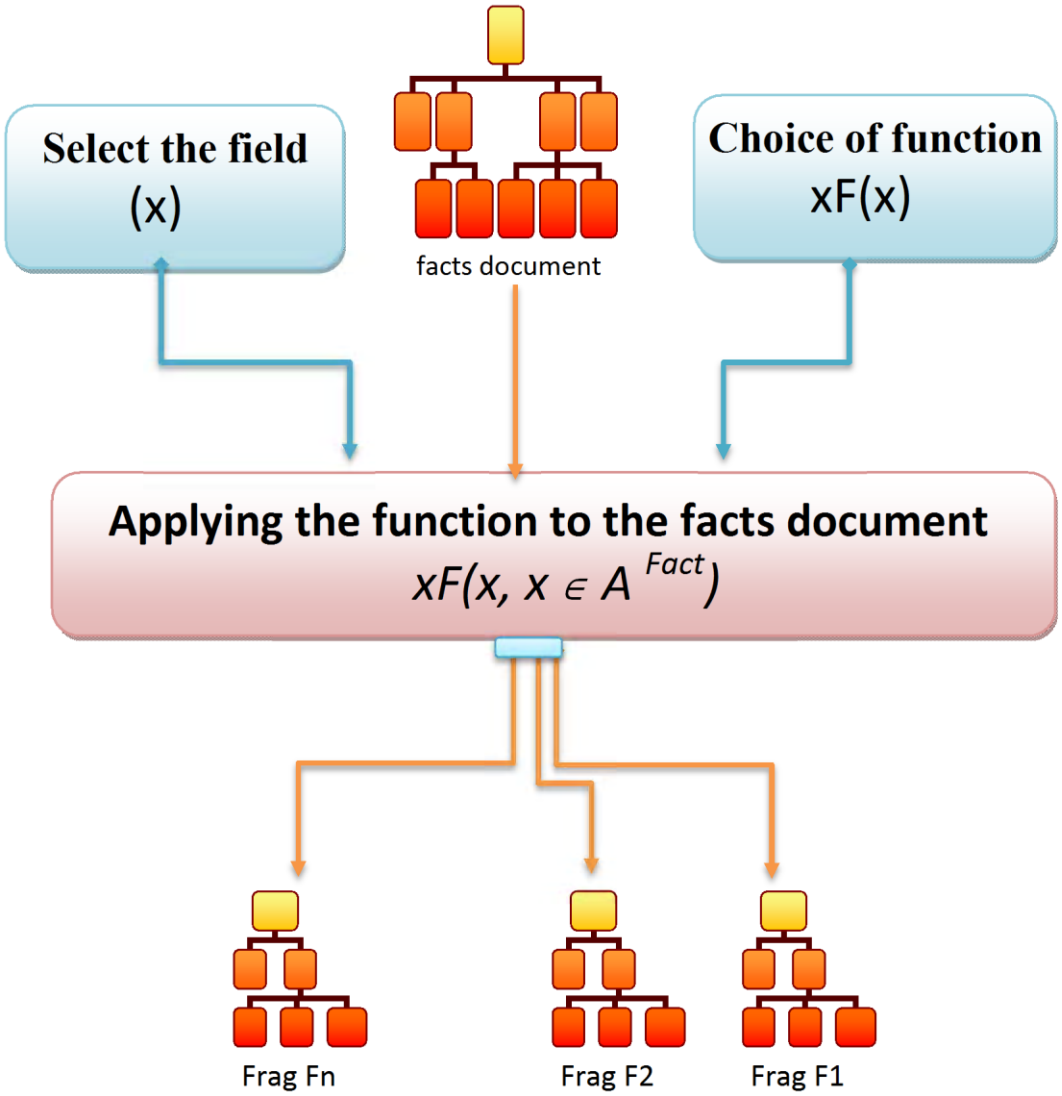

Figure 59 the working principle of the proposed approach "XHaFrag"

Due to the simplicity of the principle upon which the proposed approach is based, this fragmentation can be applied by using different programming tools, with various database systems compatible with XML files, both relational or Native XML.

From our perspective XHaFrag approach is characterized by the following advantages:

- The simplicity and clarity of principle.
- Ease of application and implementation.
- Diversity of function, and non-compliance with one hash function.
- The possibility of working with various fields or properties of facts document.

- It works only on the facts document, no other document from the warehouse (dimensions).
- Parts (fragments) resulting shall be proximate in size - generally -.

In our experiment of this approach, we worked with the Hash function: MOD 7, with the f_quantity property, located in the facts table *sales*, in the document *Fact.xml*, The output that we have obtained is 7 parts of a document. We will show the findings of the experiment, in the following section of this chapter in details.

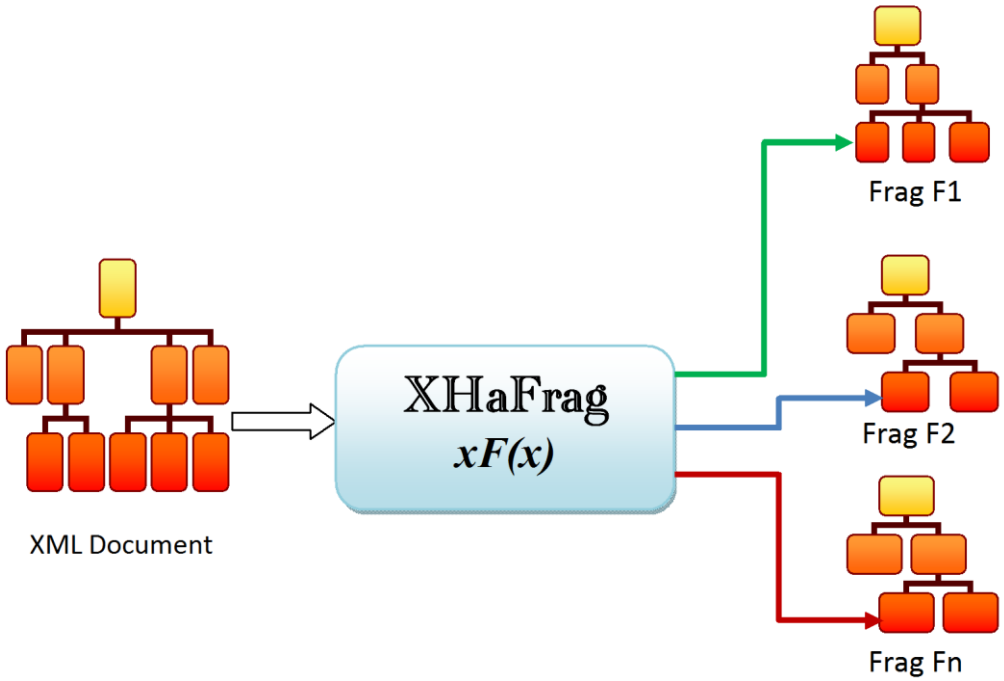

Figure 60 XHaFrag , XML Hash Fragmentation

4.2 Section II : Benchmarking And Experiences

4.2.1 Benchmark:

The benchmark is used to evaluate the performance of DBMS in general, and to test the effectiveness of the performance optimization techniques, in particular. Many benchmarks have been proposed to evaluate the performance of XML data management systems, Including XMach1 (Böhme and Rahm, 2001) ,XMark (Schmidt et al., 2002), XOO7 (Bressan et al., 2003), XBench (Yao et al., 2004), MemBeR (Afanasiev et al., 2005) and the Michigan Benchmark (Runapongsa et al., 2006) , These benchmarks also differ in:

- The fixed or flexible nature of the XML schema (one or several Document Type Definitions or XML schemas);
- The number of XML documents used to model the database at the physical level (one or several);
- The inclusion or the exclusion of update operations in the workload.

In this sense, Mahboubi and Darmont propose a test bench for XML data warehouses called XWeB (XML Data Warehouse Benchmark) (Mahboubi et al., 2006), Their main motive comes from the fact that existing XML benchmarks do not allow to evaluate the performance of BI applications, These systems propose transactional databases and their Workloads are simple queries. Mahboubi and Darmont therefore present through XWeB A decision tool based on a warehouse modeled multidimensionally which operates with more complex BI queries XQuery. the data warehouse of XWeB is composed of fact table *sales* store in the document *Fact_Sale.xml*, and four dimensions: *products, customers, supplies and date* stored in documents $dimension_{nom\_dimension}.xml$ (Figure 61).

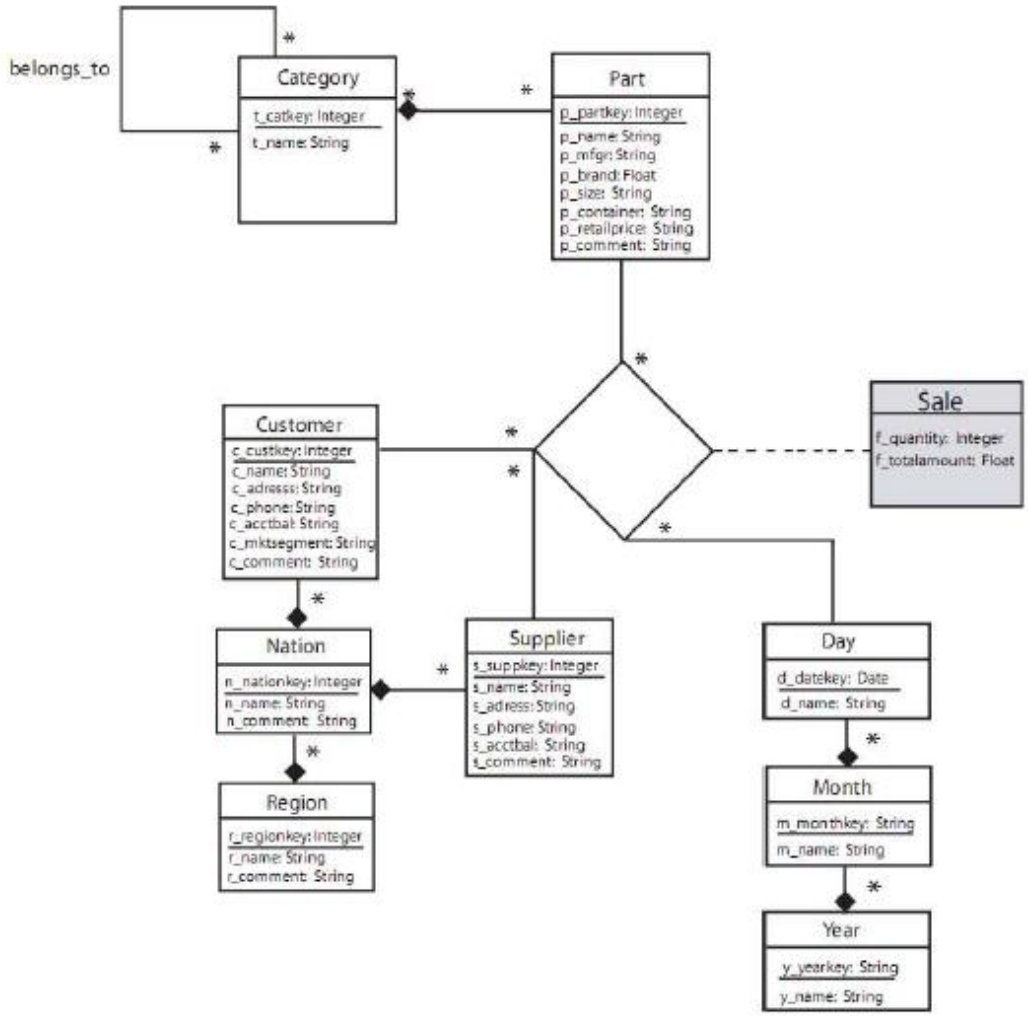

Figure 61 Conceptual Schema of the warehouse XWeB

XWeB's **workload** is currently composed of Twenty decision-support queries in XQuery Language labeled Q1 to Q20. Structured in increasing order of query complexity, These queries exploit the warehouse schema through join, selection and grouping operations. Subdivided into five categories: simple reporting queries, 1, 2 and 3-dimension cubes; and complex hierarchy cubes ,then there are the Boolean execution parameters: RE, 1D, 2D, 3D and CH, Their specifications are provided in Table10 .

Group	Query	Specification	Restriction	Ordering
Reporting	Q01	Min, Max, Sum, Avg of f_quantity and f_totalamount		
	Q02	f_quantity for each p_partkey	p_retailprice ≤ 1000	p_retailprice
	Q03	Sum of f_totalamount	n_name = ”FRANCE”	
1D cube	Q04	Sum of f_quantity per p partkey	p_retailprice > 1500	$p\_retailprice^{-1}$
	Q05	Sum of f_quantity and f_total-amount per m_monthname	Quarter(m_month key) = 1	m_monthnam e
	Q06	Sum of f_quantity and f_total-amount per d_dayname	Quarter(m_month key) = 1	d dayname
	Q07	Avg of f_quantity and f_total-amount per r_name	r_name = ”AMERICA”	
2D cube	Q08	Sum of f_quantity and f_total-amount per c_name and p_name	p_brand = ”Brand#25”	c_name, p_name,
	Q09	Sum of f_quantity and f_total-amount per n_name and p_name	p_brand = ”Brand#25”	n_name, p_name,
	Q10	Sum of f_quantity and f_total-amount per r_name and p_name	p_brand = ”Brand#25”	r_name, p_name,
	Q11	Max of f_quantity and f_total-amount per s_name and p_name	s_acctbal < 0	s name, p_name,
3D cube	Q12	Sum of f_quantity and f_total-amount per c_name, p_name and y_yearkey		c_name, p_name, y_yearkey
	Q13	Sum of f_quantity and f_total-amount per c_name, p_name and y_yearkey	y_yearkey > 2000 and c_acctbal > 5000	c_name, p_name, y_yearkey
	Q14	Sum of f_quantity and f_total-amount per c_name, p_name and y_yearkey	c_mktsegment = ”AUTO-MOBILE” and y_yearkey = 2002	c_name, p_name, y_yearkey
Complex hierarchy	Q15	Avg of f_quantity and f_total-amount per t_name		t_name
	Q16	Avg of f_quantity and f_total-amount per t_name	t_name = ”BRUSHED”	t_name
	Q17	Avg of f_quantity and f_total-amount per p_name	t_name = ”BRUSHED”	p_name
	Q18	Sum of f_quantity and f_total-amount per p_name	p_size > 40	p_name
	Q19	Sum of f_quantity and f_total-amount per t_name	p_size > 40	t_name
	Q20	Sum of f_quantity and f_total-amount per t_name	p_size > 40	t_name

Table10 Specification of the workload XWeB

4.3 EXPERIENCES

4.3.1 Experimental conditions

4.3.1.1 Hardware and Software Requirements

We have implemented experiments on a desktop computer with the following characteristics:

- **CPU type**: Intel (R) Pentium (R) CPU G620 @ 2.620 GHz;
- **RAM**: 4 GB
- **Operating System**: 32-bit Windows XP Professional, Service Pack 3
- **Database system**: Native XML databases: BaseX

4.3.1.2 Data used in the experiments

In order to evaluate our approach of fragmentation optimally, we generated many data warehouses of Benchmark XWeB, with the help of the platform program allocated for the purpose of generating files and warehouse documents, as far as the dimensions are concerned we generated it just one-time, unlike the facts document. This process produced a data warehouse with the following characteristics (Table 11):

Dimension	Documents	Number of instances	Size (KB)
Customer	dimension_Customers.xml	1000	432
Supplier	dimension_Suppliers.xml	1000	486
Date	dimension_Date.xml	500	105
Part	dimension_Part.xml	1000	389

Table 11 characteristics of the test data warehouse

The used copies of Facts documents are shown in the Table12

Fact Table	Documents	Number of facts	Size (KB)
Sale	Fact_Sale_500.xml	500	138
	Fact_Sale_1000.xml	1000	293
	Fact_Sale_2000.xml	2000	567
	Fact_Sale_3000.xml	3000	768
	Fact_Sale_4000.xml	4000	1114
	Fact_Sale_5000.xml	5000	1370
	Fact_Sale_6000.xml	6000	1606
	Fact_Sale_7000.xml	7000	1876

Table12 the used copies of Facts documents

4.3.1.3 The performance of the data warehouse before Fragmentation

To do experiments, I used BaseX - It is a Native XML database system, characterized by flexibility and good performance in the management of XML databases and data warehouses, It also provides the ability to inquire about this data in XQuery language, and shows results and statistics about them, and gives the time taken for each query, We loaded XWeB warehouse and then we executed queries workload, each query individually, one by one, and we saved the time taken (execution time + results loading time)in each query. This process was repeated with each copy of facts documents. The results of executing full-workload with different data warehouses are shown in the Table 13, and in the chart in Figure 62 :

	500	1000	2000	3000	4000	5000	6000	7000
Q1	53	116	191	281	404	536	763	941
Q2	54	115	176	257	369	497	721	882
Q3	75	133	206	308	434	575	838	995
RE	182	364	573	846	1207	1608	2322	2818
Q4	51	125	196	273	381	506	639	896
Q5	48	101	181	232	295	371	493	605
Q6	49	96	165	219	282	340	460	568
Q7	53	117	176	241	307	385	504	625
1D	201	439	718	965	1265	1602	2096	2694
Q8	85	116	197	258	341	412	562	677
Q9	43	93	180	249	360	505	703	868
Q10	39	66	149	206	297	392	559	694
Q11	27	44	140	176	241	314	453	526
2D	194	319	666	889	1239	1623	2277	2765
Q12	88	148	271	422	637	977	1509	1879
Q13	82	136	258	376	567	882	1362	1685
Q14	60	109	223	331	511	764	1213	1489
3D	230	393	752	1129	1715	2623	4084	5053
Q15	141	228	402	720	1258	2202	3432	4121
Q16	98	168	304	507	897	1561	2541	3035
Q17	101	180	311	521	935	1598	2948	3532
Q18	110	186	335	520	853	1379	2197	2601
Q19	118	195	366	548	901	1432	2259	2700
Q20	121	196	378	598	935	1453	2330	2781
CH	689	1153	2096	3414	5779	9625	15707	18770
Somme	1496	2668	4975	7243	11205	16230	26486	32100

Table 13 The results of the execution of all queries with all copies of the warehouse XWeB before fragmentation

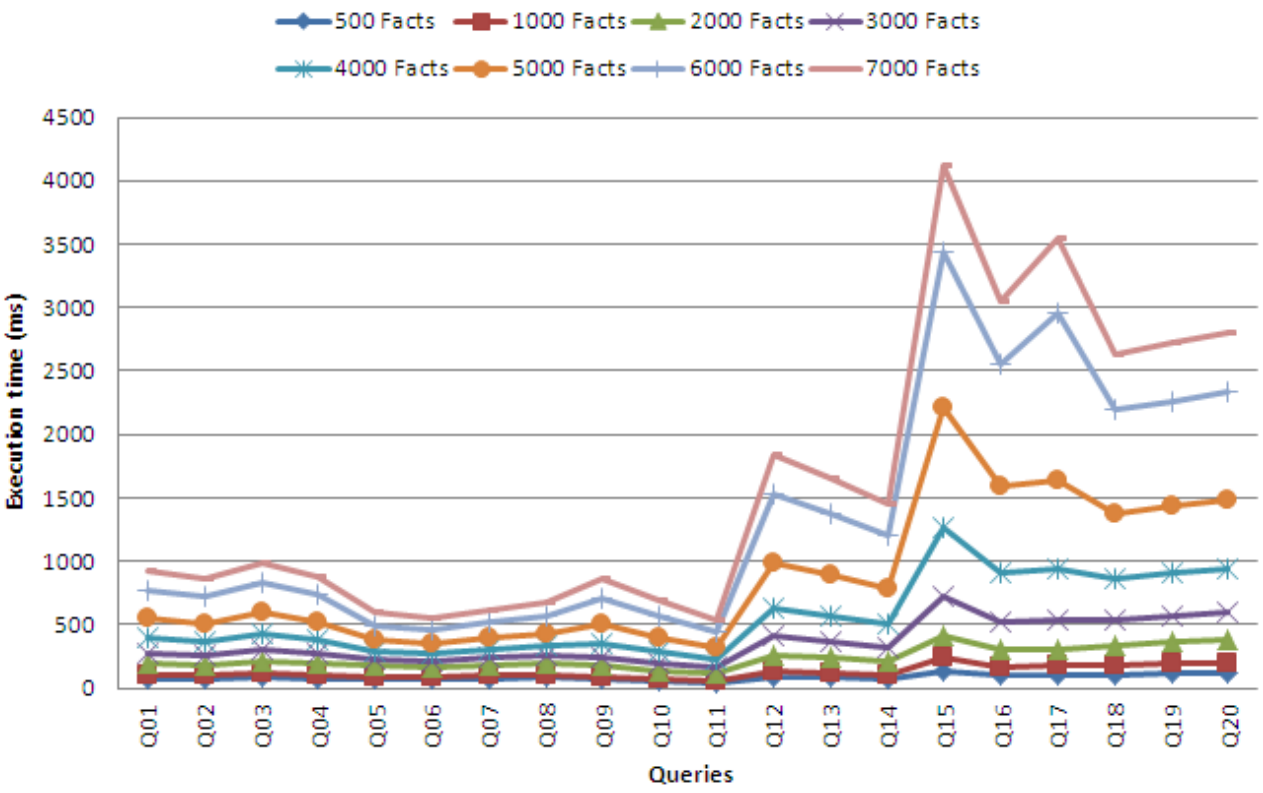

Figure 62 The results of the execution of all queries with all copies of the warehouse XWeB before fragmentation

We have noted that the execution time rises nearly with each query. For example, we found a clear difference between the queries Q1, Q6, Q15, Q20, and this is caused by increased complexity in queries - as we have pointed out previously - and also we noticed that the Q12 time was greater than Q13 and Q14 time. and this is because the results loading time was less because it was confined in a smaller number, the same thing happened with Q15 with Q16 and Q17.

To clarify the effect of the complexity of the query on time, we collected the results of each of the five categories - RE, 1D, 2D, 3D and CH – the result was a growing line from the least to most complex, as shown in the chart in Figure 63:

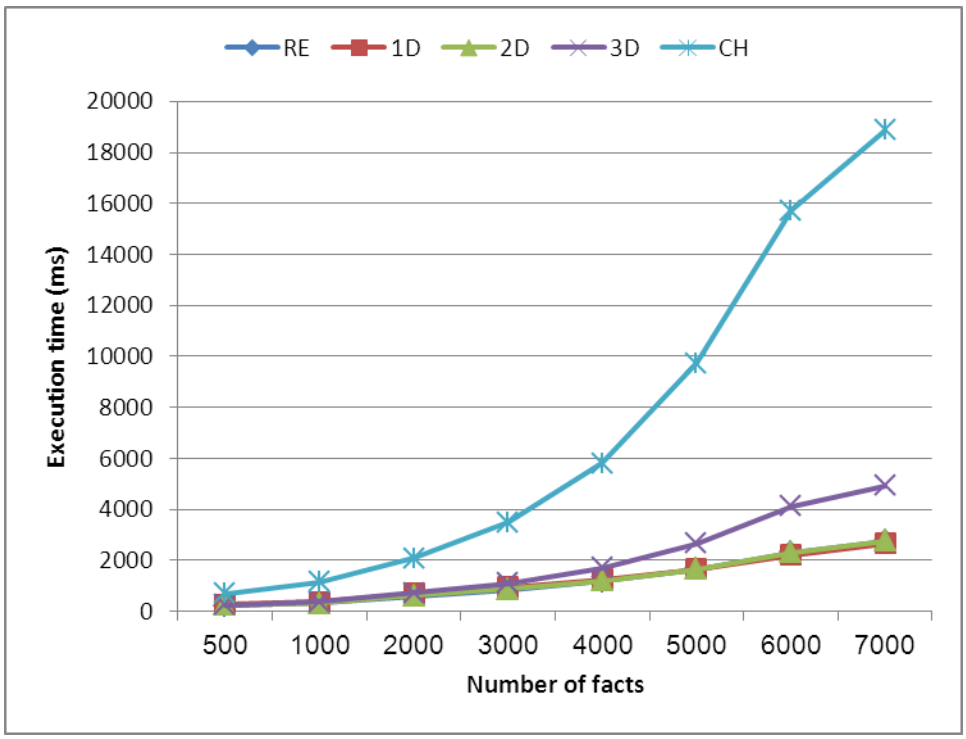

Figure 63 The results of each category queries with all copies of the warehouse XWeB before fragmentation

I have described in the previous paragraphs, the effect of query complexity on the time taken for its execution. The previous results and charts also show the impact of the number of facts on time, and to further clarify we calculated the total time taken to execute all the twentieth queries on the facts documents, one by one, the result is shown in the chart in Figure 64:

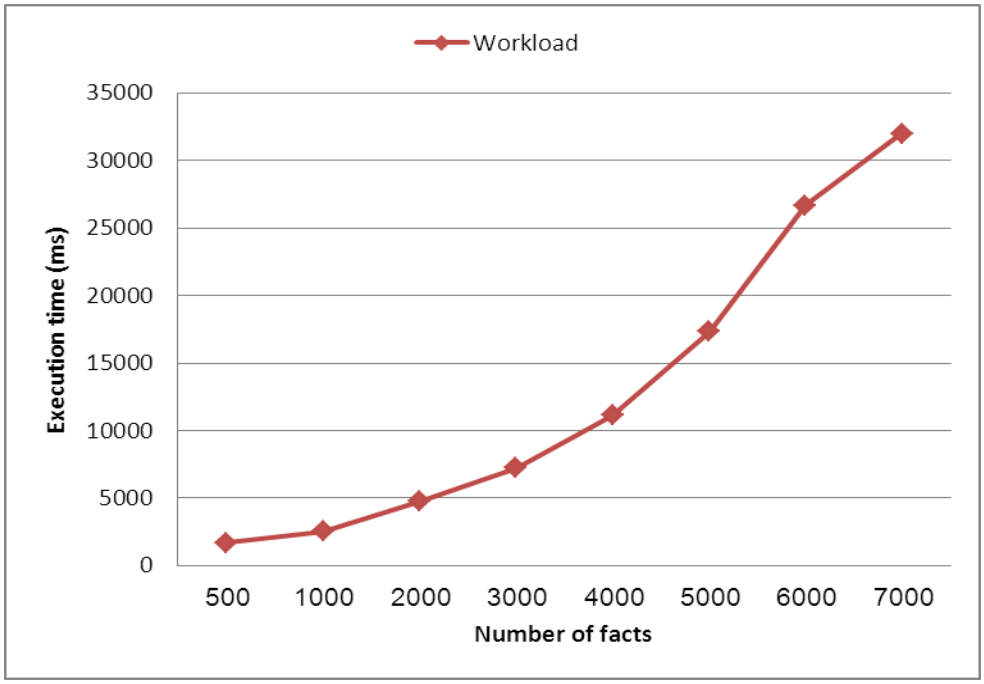

Figure 64 Total of execution time of all queries before fragmentation

4.3.1.4 The performance of the data warehouse after Fragmentation

After the application of the proposed approach XHaFrag to segment the data warehouse on XWeB, several parts of the data warehouse resulted, We loaded the fragmented warehouse with all its parts, and we repeated the same experiences that we've made to the warehouse before fragmentation, after execution of all queries on all copies of the warehouse, we got the results shown in the Table 14,and in the chart in Figure 65 :

	500	1000	2000	3000	4000	5000	6000	7000
Q1	50	77	145	208	283	396	550	691
Q2	45	72	141	198	262	369	527	627
Q3	52	86	161	233	320	448	630	758
RE	147	235	447	639	865	1213	1707	2076
Q4	50	78	140	205	277	390	541	661
Q5	42	68	120	162	217	275	354	428
Q6	46	53	118	154	205	269	340	412
Q7	52	70	130	171	227	285	372	455
1D	190	269	508	692	926	1219	1607	1956
Q8	53	80	138	180	235	304	399	476
Q9	43	69	119	170	259	353	502	610
Q10	38	52	98	140	207	295	413	509
Q11	26	41	85	109	153	238	312	385
2D	160	242	440	599	854	1190	1626	1980
Q12	51	77	160	250	372	597	914	1132
Q13	45	79	153	238	356	562	859	1067
Q14	46	68	131	206	309	504	774	942
3D	142	224	444	694	1037	1663	2547	3141
Q15	88	159	270	480	841	1485	2299	2761
Q16	63	108	191	311	552	974	1557	1874
Q17	73	122	218	369	637	1118	2009	2411
Q18	83	124	241	364	590	958	1519	1714
Q19	76	116	204	324	507	823	1290	1574
Q20	84	122	220	351	563	890	1399	1692
CH	467	751	1344	2199	3690	6248	10073	12026
Somme	1106	1721	4012	4823	7372	12573	17560	23478

Table 14 The results of the execution of all queries with all copies of the warehouse XWeB After XHaFrag fragmentation

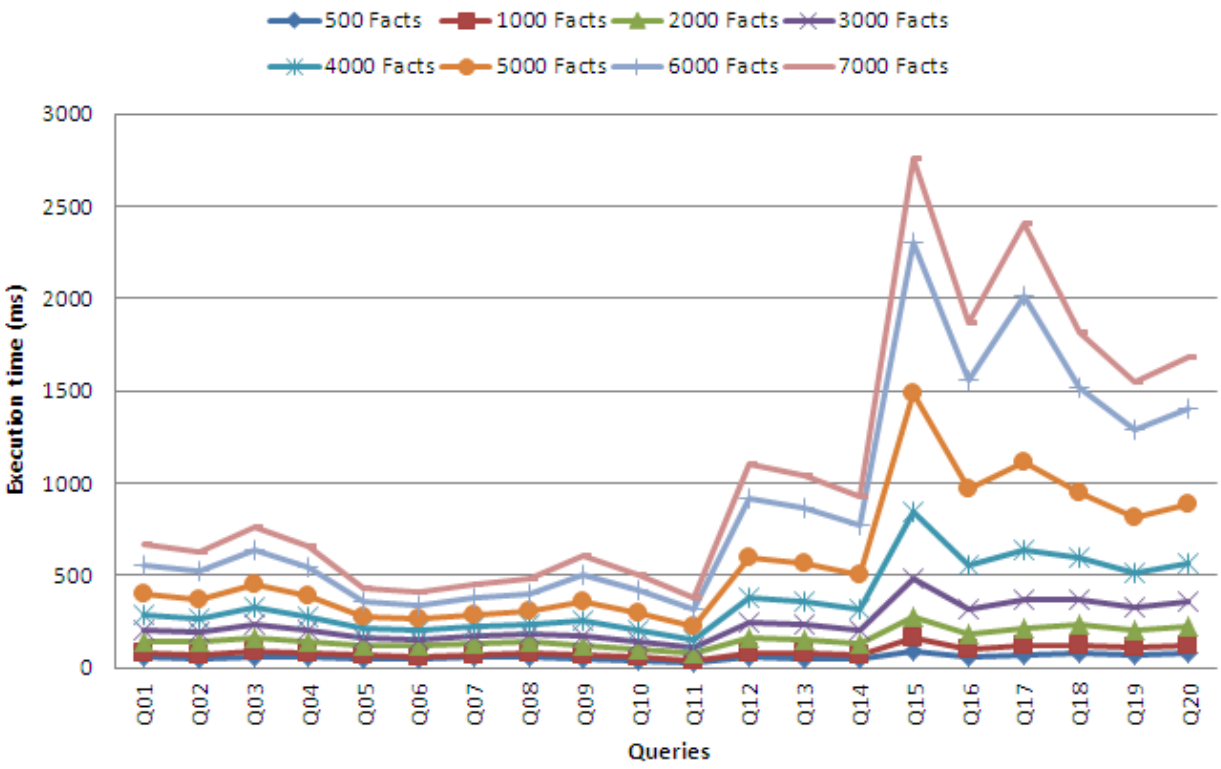

Figure 65 The results of the execution of all queries with all copies of the warehouse XWeB After XHaFrag fragmentation

It appears clearly from the chart above, that the longest period of execution time is less than than 3,000 milliseconds, whereas it exceeded 4000 milliseconds before fragmentation, and precisely, the execution time of query Q15 on a document with 7000 fact was: 4121 ms before fragmentation application, But after it, the time has been reduced to 2761 ms, the difference was 1360 ms (33%), to illustrate more, we calculated the results of each category of the five categories - RE, 1D, 2D, 3D and CH -, we got the results shown in the

Figure 66:

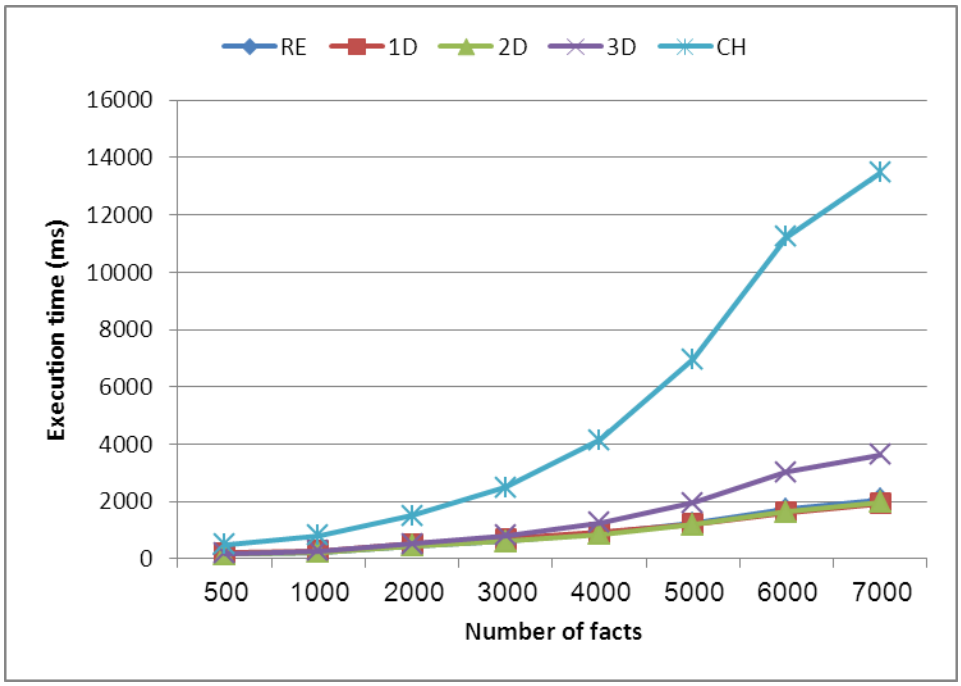

Figure 66 The results of each category queries with all copies of the arehouse XWeB After XHaFrag fragmentation

The result of Category 2D with 7000 facts was 1980 milliseconds, but it was 2765 ms before fragmentation, the decrease of 29%, the same thing, in the case of CH Category 18770 ms before fragmentation to 12026 ms later, a decrease of 28%, When we collected all the results for each queries on all parts, we got the chart in

Figure 67.

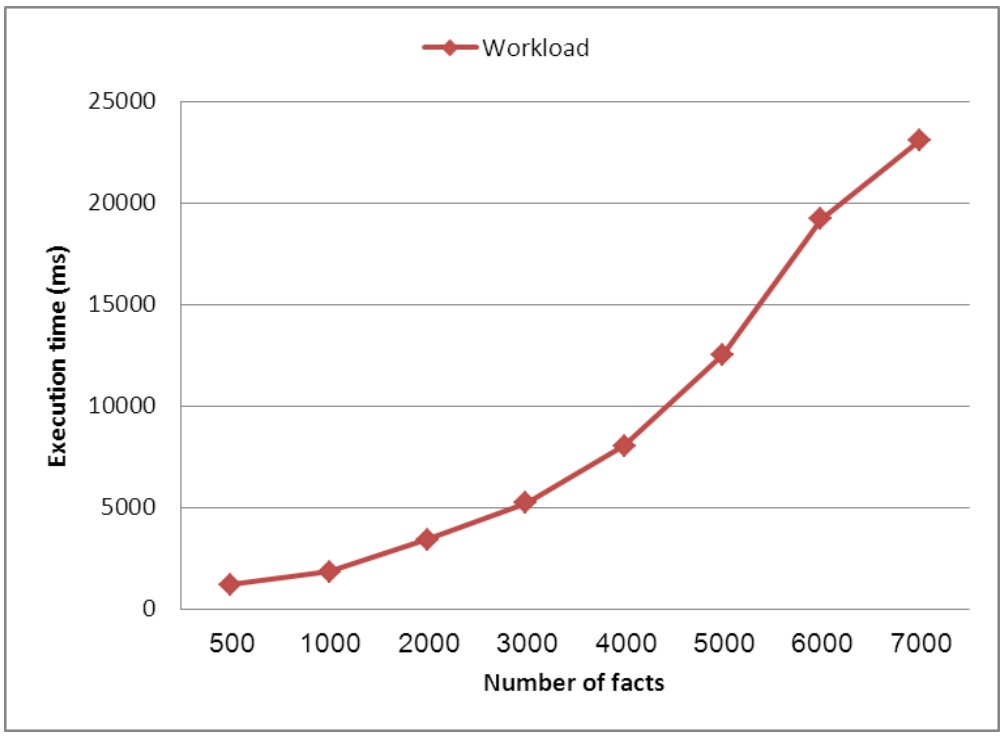

Figure 67 Total of execution time of all queries After XHaFrag fragmentation

Comparison between the previous diagram in

Figure 67 and diagram in Figure 64, shows a clear difference between the execution time of queries, especially with the increasing of number of facts. In a data warehouse with the facts document of 7000 fact, the execution time before fragmentation was 32100 but after the implementation of fragmentation *XHaFrag* it became 23 478 milliseconds, (a decrease of 27%), but with the 5000 facts, the execution time before fragmentation was 16230 but after the implementation of fragmentation *XHaFrag* it became 12573 milliseconds, (a decrease of 23%). And with the 2000 facts document the time taken before fragmentation was 4975 but after the implementation of *XHaFrag* it became 4012 milliseconds, (a decrease of 19%),we noticed that the performance of fragmentation became better with the increase in the number of facts, this proves that the horizontal fragmentation in general and *XHaFrag* especially play a big role in improving the performance of queries, and reduce execution time, and to explain more we put the following charts, which show the difference between the execution time before and after

fragmentation, and this with the facts documents of 7000 and 500 fact, with five categories - RE, 1D, 2D, 3D and CH – (Figure 68,Figure 69), and with a full work load (Figure 70,Figure 71),

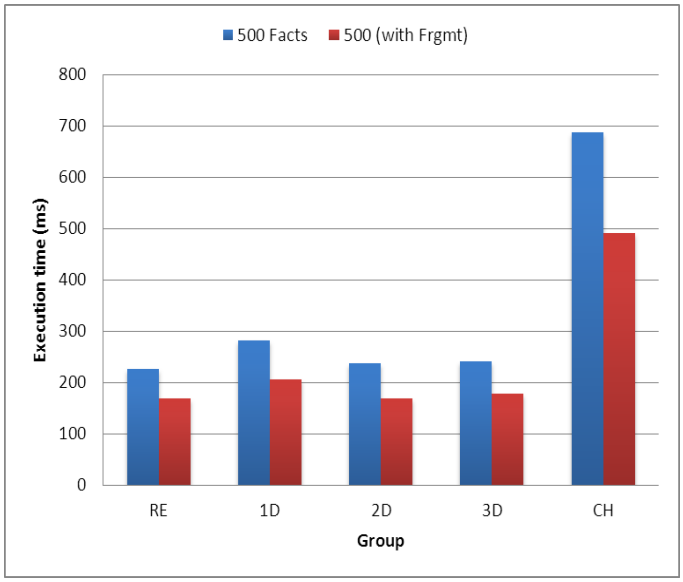

Figure 69 Execution time of the five categories with 500 facts before and after fragmentation

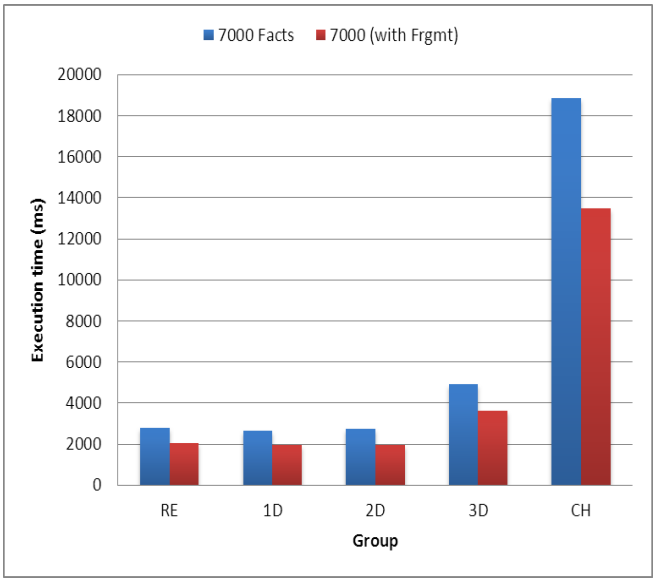

Figure 68 Execution time of the five categories with 7000 facts before and after fragmentation

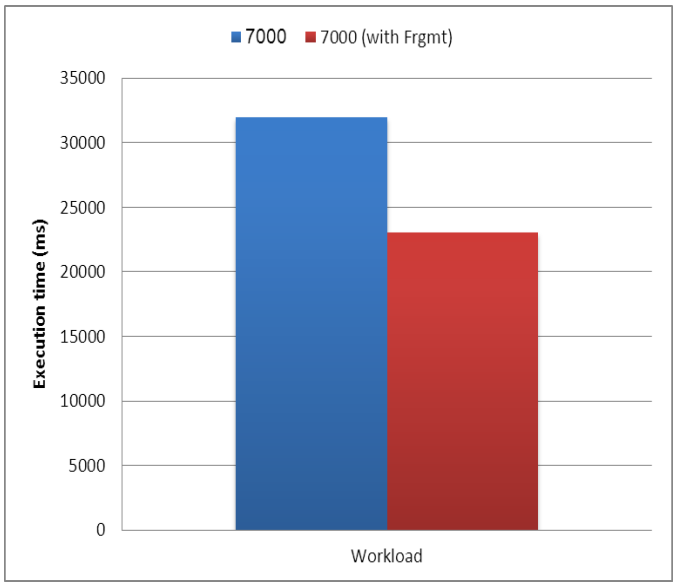

Figure 70 Execution time of the workload with 7000 facts before and after fragmentation

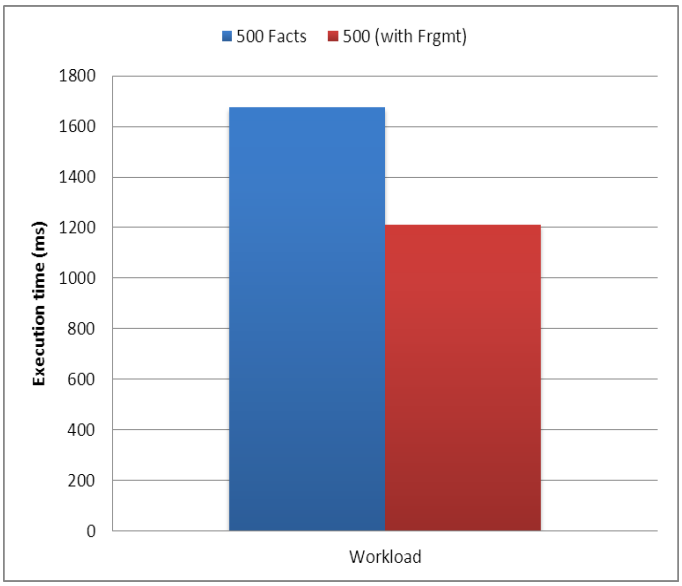

Figure 71 Execution time of the workload with 500 facts before and after fragmentation

4.4 Conclusion:

In this chapter we proposed an approach to the fragmentation of complex data warehouse XML by hash, which we called XHaFrag short for "XML Hash Fragmentation", it is based on the fragmentation of the facts document only, because it is the largest document, and the most updated among the rest of the documents (dimensions). This approach is characterized by the simplicity and clarity of its principle, ease of application and implementation. It is also characterized by diversity of function, and non-compliance with one hash function, and the possibility of working with various fields or properties of facts document.

This approach was subject to experiment on the benchmark XWeB, which proved its efficiency and its usefulness and the experiment also proved that it is an idea that can be developed and applied.

CONCLUSION

Conclusion

The work presented in this thesis is part of the physical design of the complex data warehouse XML. We have proposed a query optimization approach based on the concepts of Hash Fragmentation.

Many models and optimization techniques have been proposed in the literature. We have presented in our research the main techniques to improve the performance of data warehouses, such as indexing, materialized views, and fragmentation with its types. These techniques are based on different models of data warehouses: ROLAP and MOLAP model. We have also provided technologies related to XML in terms of basic concepts and the foundations of XML databases, and the complex data warehouses as well. In addition to that we have mentioned the different techniques to improve the performance of database and data warehouse, we also focused on indexing and fragmentation, and we cited techniques for XML data warehouses.

Finally, we proposed *XHaFrag* approach for fragmentation of complex XML data warehouses. This approach is based on the hash function and the primary horizontal fragmentation, According to this approach, only the facts document is fragmented, because the data warehouse is based mainly on the facts table, which is the larger table and the most updated, compared to dimension tables, For this we have proposed in our approach, to fragment the facts document in particular, we have not determined in our proposal a particular hash function or particular field or property, but we have left the door open for the use of any possible hash function, and the use of any field of facts table fields, that must be simple numerically or textual.

In our research, experiments were conducted by using XWeB benchmark, and native XML databases system BaseX, and we used the hash function: [xF (x) = x MOD 7], and we have chosen the field: *f_quantity* of facts document *Fact.xml*.

The results were mentioned in the previous chapter, clearly demonstrate the role of horizontal fragmentation, using the proposed hash function from us, then the results show that the rate of performance improvement to be about 27%.

We consider this approach the proposed an addition and a building block can be built upon it in subsequent research. And we know for sure that it is not without flaws, we hope that we make right her or others do in their research. We did not determine a clear and accurate algorithm for use, XHaFrag is the beginning, may built upon it, and a name which we may find who seeks to develop it, and with this we finish our mission to leave the field open to research.

Bibliographie

Abdelhédi, F., 2014. Conception assistée d'entrepôts de données et de documents XML pour l'analyse OLAP. Toulouse 1.

Abiteboul, S., Buneman, P., Suciu, D., 2000. Data on the Web: from relations to semistructured data and XML. Morgan Kaufmann.

Afanasiev, L., Manolescu, I., Michiels, P., 2005. MemBeR: a micro-benchmark repository for XQuery, in: Database and XML Technologies. Springer, pp. 144–161.

Andrade, A., Ruberg, G., Baião, F., Braganholo, V.P., Mattoso, M., 2006. Efficiently processing XML queries over fragmented repositories with PartiX, in: Current Trends in Database Technology–EDBT 2006. Springer, pp. 150–163.

Anthony, Robert Newton. "Planning and Control Systems: A Framework for Analysis." (1965): n. pag.

Aouiche, K., 2005. Techniques de fouille de données pour l'optimisation automatique des performances des entrepôts de données. Lyon 2.

Barg, M., Wong, R.K., 2003. A fast and versatile path index for querying semi-structured data, in: Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on. IEEE, pp. 249–256.

Bellatreche, L., Missaoui, R., Necir, H., Drias, H., 2007. Selection and pruning algorithms for bitmap index selection problem using data mining, in: Data Warehousing and Knowledge Discovery. Springer, pp. 221–230.

Bischoff, J., Alexander, T., 1997. Data warehouse: practical advice from the experts. Prentice-Hall, Inc.

Bizarro, P., Madeira, H., 2001. The Dimension-Join: A New Index for Data Warehouses., in: SBBD. pp. 259–273.

Böhme, T., Rahm, E., 2001. XMach-1: A benchmark for XML data management, in: Datenbanksysteme in Büro, Technik Und Wissenschaft. Springer, pp. 264–273.

Bonifati, A., Cuzzocrea, A., 2007. Efficient fragmentation of large XML documents, in: Database and Expert Systems Applications. Springer, pp. 539–550.

Bouchakri, L.B.R., Boukhalfa, K., 2009. Administration et Tuning des Entrepots de Donnees. Optim. Par Index Jointure Binaires Fragm. Horiz. HAVE BE FOUND HAVE BE FOUND.

Boukhalfa, K., 2009. De la conception physique aux outils d'administration et de tuning des entrepôts de données. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers.

Boukraâ, D., BenMessaoud, R., Boussaıd, O., 2006a. Proposition d'un Modele physique pour les entrepôts XML, in: Atelier Systemes Décisionnels (ASD 06), 9th Maghrebian Conference on Information Technologies (MCSEAI 06), Agadir, Morocco.

Boukraâ, D., BenMessaoud, R., Boussaıd, O., 2006b. Proposition d'un Modele physique pour les entrepôts XML, in: Atelier Systemes Décisionnels (ASD 06), 9th Maghrebian Conference on Information Technologies (MCSEAI 06), Agadir, Morocco.

Boukraâ, D., Boussaïd, O., Bentayeb, F., 2011. Vertical fragmentation of XML data warehouses using frequent path sets, in: Data Warehousing and Knowledge Discovery. Springer, pp. 196–207.

Boukraâ, D., Boussaïd, O., Bentayeb, F., Zegour, D.-E., 2013. A layered multidimensional model of complex objects, in: Advanced Information Systems Engineering. Springer, pp. 498–513.

Boussaid, O., Messaoud, R.B., Choquet, R., Anthoard, S., 2006a. Conception et construction d'entrepôts XML., in: EDA. pp. 3–22.

Boussaid, O., Messaoud, R.B., Choquet, R., Anthoard, S., 2006b. X-warehousing: an XML-based approach for warehousing complex data, in: Advances in Databases and Information Systems. Springer, pp. 39–54.

Braganholo, V., Mattoso, M., 2014. A Survey on XML Fragmentation. ACM SIGMOD Rec. 43, 24–35.

Bremer, J.-M., Gertz, M., 2003. On Distributing XML Repositories., in: WebDB. pp. 73–78.

Bressan, S., Lee, M.L., Li, Y.G., Lacroix, Z., Nambiar, U., 2003. The XOO7 benchmark, in: Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web. Springer, pp. 146–147.

Bush, V., others, 1945. As we may think. Atl. Mon. 176, 101–108.

Carmè, A., Mazón, J.-N., Rizzi, S., 2010. A model-driven heuristic approach for detecting multidimensional facts in relational data sources, in: Data Warehousing and Knowledge Discovery. Springer, pp. 13–24.

Chen, Q., Lim, A., Ong, K.W., 2003. D (k)-index: An adaptive structural summary for graph-structured data, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. ACM, pp. 134–144.

Chung, C.-W., Min, J.-K., Shim, K., 2002. APEX: An adaptive path index for XML data, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. ACM, pp. 121–132.

Codd, E.F., Codd, S.B., Salley, C.T., 1993. Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. Codd Date 32.

Cooper, B.F., Sample, N., Franklin, M.J., Hjaltason, G.R., Shadmon, M., 2001. A fast index for semistructured data, in: VLDB. pp. 341–350.

Corey, M.J., Abbey, M., 1996. Oracle data warehousing. Osborne/McGraw-Hill.

Darmont, J., 2006. Optimisation et évaluation de performance pour l'aide à la conception et à l'administration des entrepôts de données complexes. Université Lumière-Lyon II.

Darmont, J., Boussaid, O., Ralaivao, J.-C., Aouiche, K., 2007. An architecture framework for complex data warehouses. ArXiv Prepr. ArXiv07071534.

Davis, G.B., Olson, M.H., 1984. Management information systems: conceptual foundations, structure, and development. McGraw-Hill, Inc.

Garcia-Molina, H., 2008. Database systems: the complete book. Pearson Education India.

Goldman, R., Widom, J., 1997. Dataguides: Enabling query formulation and optimization in semistructured databases.

Golfarelli, M., Rizzi, S., Vrdoljak, B., 2001. Data warehouse design from XML sources, in: Proceedings of the 4th ACM International Workshop on Data Warehousing and OLAP. ACM, pp. 40–47.

Grust, T., 2002. Accelerating XPath location steps, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. ACM, pp. 109–120.

Gupta, H., 1997. Selection of views to materialize in a data warehouse, in: Database Theory—ICDT'97. Springer, pp. 98–112.

Hachaichi, Y., Feki, J., Ben-Abdallah, H., 2010. Modélisation multidimensionnelle de documents XML centrés-données. J. Decis. Syst. 19, 313–345.

Haifeng, J., Lu, H., Wei, W., Ooi, B.C., 2003. XR-tree: Indexing XML data for efficient structural join. ICDE IEEE Comput. Soc.

Haisten, M., 2003. Real time data warehouse: The next stage in data warehouse evolution. DM Rev.

Halverson, A., Burger, J., Galanis, L., Kini, A., Krishnamurthy, R., Rao, A.N., Tian, F., Viglas, S.D., Wang, Y., Naughton, J.F., others, 2003. Mixed mode XML query processing, in: Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29. VLDB Endowment, pp. 225–236.

Hammerschmidt, B.C., 2006. KeyX: Selective Key-Oriented Indexing in Native XML-Databases. IOS Press.

Hümmer, W., Bauer, A., Harde, G., 2003. XCube: XML for data warehouses, in: Proceedings of the 6th ACM International Workshop on Data Warehousing and OLAP. ACM, pp. 33–40.

Inmon, W.H., 2005. Building the data warehouse. Wiley. com.

Inmon, W.H., 2000. WHAT IS AData WAREHOUSE?

Inmon, W.H., n.d. Building the Data Warehouse, 1992. QED Inf. Sci. Wellesley MA.

Ivancevich, J.M., Lorenzi, P., Skinner, S.J., Crosby, P.B., 1994. Management: Quality and competitiveness. Irwin Burr Ridge, IL.

Kalakota, R., Robinson, M., 2000. E-business 2.0. Looking over the new horizon. EAI J. 22–30.

Kalta, M., Lowe, T., Tyler, D., 1998. A Decision Support System for designing assembly cells in Apparel Industry, in: Group Technology and Cellular Manufacturing. Springer, pp. 255–272.

Kaushik, R., Bohannon, P., Naughton, J.F., Korth, H.F., 2002. Covering indexes for branching path queries, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. ACM, pp. 133–144.

Kaushik, R., Krishnamurthy, R., Naughton, J.F., Ramakrishnan, R., 2004. On the integration of structure indexes and inverted lists, in: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. ACM, pp. 779–790.

Kechar, M., Bahloul, S.N., 2014. Hybrid Fragmentation of XML Data Warehouse Using K-Means Algorithm, in: Advances in Databases and Information Systems. Springer, pp. 70–82.

Keen, P.G., Morton, M.S.S., 1978. Decision support systems: an organizational perspective. Addison-Wesley Reading, MA.

Kelly, S., 2007. Data warehousing in action. John Wiley & Sons.

Kerkad, A., 2013. L'interaction au service de l'optimisation à grande échelle des entrepôts de données relationnels. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique - Poitiers.

Kha, D.D., Yoshikawa, M., Uemura, S., 2001. An XML indexing structure with relative region coordinate, in: Data Engineering, 2001. Proceedings. 17th International Conference on. IEEE, pp. 313–320.

Kimball, R., 1998. The data warehouse lifecycle toolkit: expert methods for designing, developing, and deploying data warehouses. John Wiley & Sons.

Kimball, R., Ross, M., 2002. The data warehouse toolkit: the complete guide to dimensional modelling. Nachdr N. Y. Ua Wiley.

Kim, J., Kim, D., Song, I.-Y., Lee, S., Moon, Y.-S., Khare, R., An, Y., 2009. SAMSTARplus: An Automatic Tool for Generating Multi-Dimensional Schemas from an Entity-Relationship Diagram. Rev. Informática Teórica E Apl. 16, 79–82.

Kling, P., Özsu, M.T., Daudjee, K., 2011. Scaling XML query processing: distribution, localization and pruning. Distrib. Parallel Databases 29, 445–490.

Kling, P., Özsu, M.T., Daudjee, K., 2010. Generating efficient execution plans for vertically partitioned XML databases. Proc. VLDB Endow. 4, 1–11.

Koong, K.-L., Haw, S.-C., Soon, L.-K., Subramaniam, S., 2015. Prefix-based Labeling Annotation for Effective XML Fragmentation. ArXiv Prepr. ArXiv150503246.

Kurita, H., Hatano, K., Miyazaki, J., Uemura, S., 2007. Efficient query processing for large XML data in distributed environments, in: Advanced Information Networking and Applications, 2007. AINA'07. 21st International Conference on. IEEE, pp. 317–322.

Lalmas, M., 2009. XML Retrieval (Synthesis Lectures on Information Concepts, Retrieval, and Services). Morgan Claypool San Rafael CA.

Laudon, Kenneth C., and Jane Price Laudon. *Management Information Systems*. Vol. 6. Prentice Hall New Jersey, 2000.

List, B., Schiefer, J., Tjoa, A.M., 2000. Process-oriented requirement analysis supporting the data warehouse design process a use case driven approach, in: Database and Expert Systems Applications. Springer, pp. 593–603.

Li, Y., An, A., 2005. Representing UML snowflake diagram from integrating XML data using XML schema, in: Data Engineering Issues in E-Commerce, 2005. Proceedings. International Workshop on. IEEE, pp. 103–111.

Lonjon, A., Thomasson, J.-J., 2006. Modélisation XML. Editions Eyrolles.

Mahboubi, H., 2008. Optimisation de la performance des entrepôts de données XML par fragmentation et répartition. Université Lumière-Lyon II.

Mahboubi, H., Darmont, J., 2009a. Indices in XML databases. Encyclopedia of Database Technologies and Applications, Second Edition. Idea Group Publishing.

Mahboubi, H., Darmont, J., 2009b. Enhancing xml data warehouse query performance by fragmentation, in: Proceedings of the 2009 ACM Symposium on Applied Computing. ACM, pp. 1555–1562.

Mahboubi, H., Darmont, J., 2007. Fragmentation des entrepôts de données XML. Actes 3èmes Journ. Francoph. Sur Entrepôts Données Anal. En Ligne EDA 07 177–190.

Mahboubi, H., Darmont, J., others, 2006. Benchmarking XML data warehouses, in: Atelier Systèmes Décisionnels (ASD 06), 9th Maghrebian Conference on Information Technologies (MCSEAI 06). pp. 1–13.

Ma, H., Schewe, K.-D., 2003. Fragmentation of XML Documents., in: SBBD. pp. 200–214.

McHugh, J., Abiteboul, S., Goldman, R., Quass, D., Widom, J., 1997. Lore: A database management system for semistructured data. SIgMOD Rec. 26, 54–66.

McLeod, R., Schell, G., 2001. Management Information Systems 8/e. Prentice-Hall, Inc.

Milo, T., Suciu, D., 1999. Index structures for path expressions, in: Database Theory—ICDT'99. Springer, pp. 277–295.

Nahmias, S., n.d. Production and operations analysis, 1997. Irwin Homewood IL.

Nassis, V., Rajugan, R., Dillon, T.S., Rahayu, W., 2004. Conceptual design of XML document warehouses, in: Data Warehousing and Knowledge Discovery. Springer, pp. 1–14.

Naughton, J.F., DeWitt, D.J., Maier, D., Aboulnaga, A., Chen, J., Galanis, L., Kang, J., Krishnamurthy, R., Luo, Q., Prakash, N., others, 2001. The Niagara internet query system. IEEE Data Eng Bull 24, 27–33.

Negash, S., 2004. Business intelligence. Commun. Assoc. Inf. Syst. 13, 54.

Niemi, T., Niinimäki, M., Nummenmaa, J., Thanisch, P., 2002. Constructing an OLAP cube from distributed XML data, in: Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP. ACM, pp. 22–27.

O'brien, J.A., 1998. Introduction to information systems. McGraw-Hill, Inc.

O'Neil, P., Graefe, G., 1995. Multi-table joins through bitmapped join indices. ACM SIGMOD Rec. 24, 8–11.

O'Neil, P., Quass, D., 1997. Improved query performance with variant indexes, in: ACM Sigmod Record. ACM, pp. 38–49.

Ouaret, Z., 2008. Modélisation conceptuelle et logique d'un entrepôt de données XML. L'École nationale Supérieure d'Informatique l'ESI.

Pagnamenta, F., 2005. Design and initial implementation of a distributed xml database.

Pendse, N., 2003. What is OLAP. Anal. What Increasingly Misused OLAP Termis Supposed Mean URL Httpwww Olapreport Comfasmi Htm.

Pinet, F., Schneider, M., 2009. A unified object constraint model for designing and implementing multidimensional systems, in: Journal on Data Semantics XIII. Springer, pp. 37–71.

Pokorny, J., 2002. XML data warehouse: Modelling and querying, in: Databases and Information Systems II. Springer, pp. 67–80.
Pokorny, J., 2001. Modelling stars using XML, in: Proceedings of the 4th ACM International Workshop on Data Warehousing and OLAP. ACM, pp. 24–31.
Pujolle, G., Ravat, F., Teste, O., Tournier, R., Zurfluh, G., 2011. Multidimensional database design from document-centric XML documents, in: Data Warehousing and Knowledge Discovery. Springer, pp. 51–65.
Rajugan, R., Chang, E., Dillon, T.S., 2005. Conceptual design of an XML FACT repository for dispersed XML document warehouses and XML marts, in: Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on. IEEE, pp. 141–149.
Ronald, B., 2003. XML et les bases de données [WWW Document]. URL http://peccatte.karefil.com/software/rbourret/xmlbd.htm#datavdocs (accessed 10.18.14).
Runapongsa, K., Patel, J.M., Jagadish, H.V., Chen, Y., Al-Khalifa, S., 2006. The Michigan benchmark: towards XML query performance diagnostics. Inf. Syst. 31, 73–97.
Russell, R.S., Taylor, B.W., 2003. Operations management. Prentice Hall ^ eNew Jersey New Jersey.
Rusu, L.I., Rahayu, J.W., Taniar, D., 2005. A methodology for building XML data warehouses. Int. J. Data Warehous. Min. IJDWM 1, 23–48.
Schenkel, R., Theobald, A., Weikum, G., 2004. HOPI: An efficient connection index for complex XML document collections, in: Advances in Database Technology-EDBT 2004. Springer, pp. 237–255.
Schmidt, A., Waas, F., Kersten, M., Carey, M.J., Manolescu, I., Busse, R., 2002. XMark: A benchmark for XML data management, in: Proceedings of the 28th International Conference on Very Large Data Bases. VLDB Endowment, pp. 974–985.
Scholer, F., Williams, H.E., Yiannis, J., Zobel, J., 2002. Compression of inverted indexes for fast query evaluation, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, pp. 222–229.
Schroeder, R., dos Santos Mello, R., Hara, C.S., 2012. Affinitybased XML Fragmentation., in: WebDB. pp. 61–66.
Simon, H.A., Dantzig, G.B., Hogarth, R., Plott, C.R., Raiffa, H., Schelling, T.C., Shepsle, K.A., Thaler, R., Tversky, A., Winter, S., 1987. Decision making and problem solving. Interfaces 17, 11–31.
Slimani, N., DARMONT, M.J., MAHBOUBI, M.H., 2007. Entrepôts de données XML répartis sur grille de données. Master's thesis, Université Lyon 2.
Song, I.-Y., Khare, R., An, Y., Lee, S., Kim, S.-P., Kim, J., Moon, Y.-S., 2008. SAMSTAR: An automatic tool for generating star schemas from an entity-relationship diagram, in: Conceptual Modeling-ER 2008. Springer, pp. 522–523.
Tseng, F.S., Chou, A.Y., 2006. The concept of document warehousing for multi-dimensional modeling of textual-based business intelligence. Decis. Support Syst. 42, 727–744.
Valduriez, P., 1987. Join indices. ACM Trans. Database Syst. TODS 12, 218–246.
Vrdoljak, B., Banek, M., Rizzi, S., 2003. Designing web warehouses from XML schemas, in: Data Warehousing and Knowledge Discovery. Springer, pp. 89–98.
Vrdoljak, B., Banek, M., Skočir, Z., 2006. Integrating XML sources into a data warehouse, in: Data Engineering Issues in E-Commerce and Services. Springer, pp. 133–142.
Wang, H., Park, S., Fan, W., Yu, P.S., 2003. ViST: a dynamic index method for querying XML data by tree structures, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. ACM, pp. 110–121.
Yao, B.B., Ozsu, M.T., Khandelwal, N., 2004. XBench benchmark and performance testing of XML DBMSs, in: Data Engineering, 2004. Proceedings. 20th International Conference on. IEEE, pp. 621–632.

Yeh, L., Gardarin, G., others, 2004. Indexing XML Objects with Ordered Schema Trees., in: BDA. pp. 21–45.

Yeoh, W., Koronios, A., 2010. Critical success factors for business intelligence systems. J. Comput. Inf. Syst. 50, 23–32.

Zezula, P., Amato, G., Rabitti, F., 2003. Processing XML queries with tree signatures, in: Intelligent Search on XML Data. Springer, pp. 247–258.

Zhang, J., Ling, T.W., Bruckner, R.M., Tjoa, A.M., 2003. Building XML data warehouse based on frequent patterns in user queries. Springer.

Zhang, J., Wang, W., Liu, H., Zhang, S., 2005. X-warehouse: building query pattern-driven data, in: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web. ACM, pp. 896–897.