The GUHA Virtual Machine - Frameworks and Key Concept. Research Report COST 274

Feglar, Tomáš
2001

Institute of Computer Science
Academy of Sciences of the Czech Republic

The GUHA Virtual Machine – Frameworks and Key Concept Research Report COST 274

Tomáš Feglar

Technical report No. 858

November 2001

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: (+4202) 6605 1111, fax: (+4202) 858 57 89,
e-mail:feglar@czn.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

The GUHA Virtual Machine – Frameworks and Key Concept Research Report COST 274

Tomáš Feglar

Technical report No. 858

November 2001

Abstract:

The report describes and developes the notion of the GUHA Virtual Machine and its general, analytical, structuring and decision support modelling frameworks. It is a contribution to the Czech part of the COST Action 274 - TARSKI.

Keywords:
GUHA method, knowledge discovery in databases

SUMMARY

Introduction

This document summarizes research works made by the author in this year under the research project COST 274.

The first part concerns of the knowledge discovery in databases (KDD) process in a relation to the most important Modules of the Modern Database Systems. It stresses the fact that there is very important to have clear understanding of database complexity to be able to develop and design new KDD methods.

The second part introduces the GUHA Frameworks which are understand by the author as the backbone of a research for the hole time period of the COST 274 duration.
There are four frameworks concern of the GUHA Virtual Machine (GVM) design:

- The General Framework,
- The Analytical Processing Framework,
- The Structuring Framework,
- The Decision Support Modeling Framework.

The main longtime goal of the second part is understanding of a workflows, typical for analytical and decision support areas.

The GVM Concept in the third part focuses two views.

The Logical View

This part of the GVM Concept describes functional aspects of the GUHA virtual machine. It captures IT realizable parts of workflows and converts them into functional and logical diagrams. The Logical View reflects the Architect's thinking and understanding.

The Physical View

This part of the GVM Concept describes implementation aspects of the GUHA virtual machine. It captures components and component's structures needed to cover functional aspects. The Physical View reflects the Designer's thinking and understanding.

The main research items proposed for the next year are included in the part 4.

1. The GUHA method and knowledge discovery in databases

1.1. The GUHA method and data mining

The GUHA method was originally developed as exploratory data analysis tool in mid-sixties /1/. Analysts at those time dealt with only small set of analytically interesting data, which were collected by a proprietary methods. Rapid changes in the area of analytical processing and data mining sufficiently change current situation. Compared with modern data mining methods GUHA lacks a tightly coupling to the database technology. At the same time GUHA has several features typical for data mining /2/:

- search for relationships hidden in the data,
- limiting the search to relationships interesting according to some predefined criteria,
- focus on relationships that can not be found in a trivial way,
- automating the search as far as possible,
- optimization to avoid blind search whenever possible.

1.2. The four Modules of the modern Database System

The approach is to break down the entire database system into functional modules that serve different needs. The different modules are /3/:

a) OLTP – The OLTP (Online Transaction Processing) database stores current data – that's to say data which the database needs to run its applications; it's only necessary to keep a small amount of history.
b) ODS – Operational Data Store. Consolidated data used for day to day reporting. Such data is frequently consolidated from several disparate sources, with some degree of pre-aggregation performed, in order to save query time.
c) Data Warehouse – Grand data store for holding nearly all organization data and its history.
d) Data Mart – Specialized data store optimized for aggregations, used for specific situations, and held as a subset of data warehouse. Data marts are generally processed using a technology known as Online Analytical Processing (OLAP).

OLTP

The OLTP database contains the data used in everyday transactions in the process of conducting business.

The structure of the OLTP data store is built using normalization /5,6/. Normalization reduces the amount of redundant data, helping to prevent modification anomalies. A primary goal of the OLTP database is integrity of current corporate data. This is achieved by following two principles:

- Storing each current piece of data in a single place where it can be edited, so that any change is reflected everywhere else that it is used.
- Providing transactional support, so that multiple database alterations all have to take place together. If one of the alterations in the transaction fails none of the others should be allowed to occur.

There is not suitable to develop interface between OLTP module and GUHA at least for the following reasons:

- *high frequency of a data tables updates*
- *high cost of operations thanks to multiple joins, required to synthesize analytically interesting data set.*

ODS

This module is designed to try and address some of the problems associated with the OLTP concept which can be summarized as follows:

- OLTP databases generally have a complicated structure with many tables. The data structures can be quite complex to understand, and querying information may require creative use of the SQL language.

- Many OLTP have a large number of detailed records. Day-to day operations probably don't require access to every daily transaction, but will likely need to be able to obtain summations of the data.

ODS is responsible for limited "de-normalization", when particular timely limited set of data are synthesized and stored together.

There is not suitable to develop interface between ODS module and GUHA because ODS data sets include only operational information, not suitable for tactical and strategic analyses and decisions.

Data Warehouse
The primary use of the data warehouse is to support decision making, by storing as much historical information from the organization as necessary. **Decision support** is a generic term that refers to being able to answer the difficult questions about how an organization is doing.

A development of a interface between Data Warehouse module and GUHA is reasonable; it will open a possibility to use automatically generated hypotheses for decision support. Any appropriate reduction and transformation tasks have to be developed to decrease large amount of hypotheses.

Data Marts
A data mart is a distinctive segment of a data warehouse and usually pertains to either the specific needs of a division or department within an organization. It is built using special database structures known as star or snowflake schemas. Star schemas are actually simple databases with a single fact table connected to a set of **dimension tables** that categorize the facts in the fact tables. It should be noted that data in the fact table is primarily numeric. Snowflake schemas are simply an extension of star schemas where the fact tables may also be dimension tables.
The generic building unit of Data marts is a **Cube**. It is the Conceptual container of detailed values from a single fact table, along with all possible aggregations for one or more dimension hierarchies /7/.
There exist an SQL-like language to access the data in a data mart – **MDX – Multidimensional Expressions.**

A development of a interface between Data Mart module and GUHA is reasonable; it will open a possibility to use automatically generated hypotheses for analysis and data mining. There are various possibilities how to develop this interface.

1.3. Discovery outside databases

There is also discovery in science, which has been automated simultaneously with discovery in databases. Research on scientific discovery can be split into discovery of empirical lows and discovery of hidden structure / 9 /.
The very important for this direction is a Domain Expert, who understand a particular scientific field.
Discovery in science produces specific databases. Some of them may be investigated by the same way, as ordinary databases.

GUHA support for discovery in science is principally possible.

1.4. Knowledge discovery steps

Some difficulties have to be overcome on the border between GUHA terminology and terminology, used for information systems. They can be overcome taking into consideration very similar understanding of the term "information system" and the term "observational model" /3/. This unified approach gives us possibility to drill down into knowledge discovery process step by step /10/:

S01: Developing an understanding of the application domain, the relevant prior knowledge, and the goal of the end user.
S02: Creating or selecting a target data set.
S03: data cleaning and preprocessing: this step includes tasks like removing noise and imputation of missing values.

S04: Data reduction: Finding useful features to represent the data depending on the goal of the task. This may include dimensionality reduction or transformation.
S05: Matching the goals to a particular data mining method such as classification, decision support tree, regression, clustering etc.
S06: Model and hypothesis selection, choosing the data mining algorithms and methods to be used for searching for data patterns.
S07: Data mining.
S08: Interpreting mined patterns.
S09: Acting on discovered knowledge.

The GUHA method relates to the steps S05 – S09. A lot of questions are open:

- *the relationships of the GUHA to the other data mining algorithms and methods,*
- *the criteria for a choice of the optimal combination of method in dependency on application domain or a particular target data set*

2. The GVM Frameworks

As we sow in the previous chapter, GUHA method will be spread across few steps of the knowledge discovery. This is the main reason to think about a GUHA as a set of specialized components. All such components must cooperate together as well as with their environment. For the sake of simplicity we shall call this set of GUHA components the GVM – GUHA Virtual Machine.
The degree of a GUHA efficiency in relation to the KDD will be strongly dependent on many factors. Most of these factors will be well understood only in the context of an appropriate framework.
There are four frameworks concern of the GVM design:

- The General Framework,
- The Analytical Processing Framework,
- The Structuring Framework,
- The Decision Support Modeling Framework.

2.1. The General Framework

This framework consists of two loops (Fig. 2.1). The first loop (hypotheses driven loop) includes three processing nodes:

- Node 1: Information Gathering, Classification and Cleaning
- Node 2: Structuring Information in accordance with RDBM rules (E.F.Codd)
- Node 3: Analytical Processing

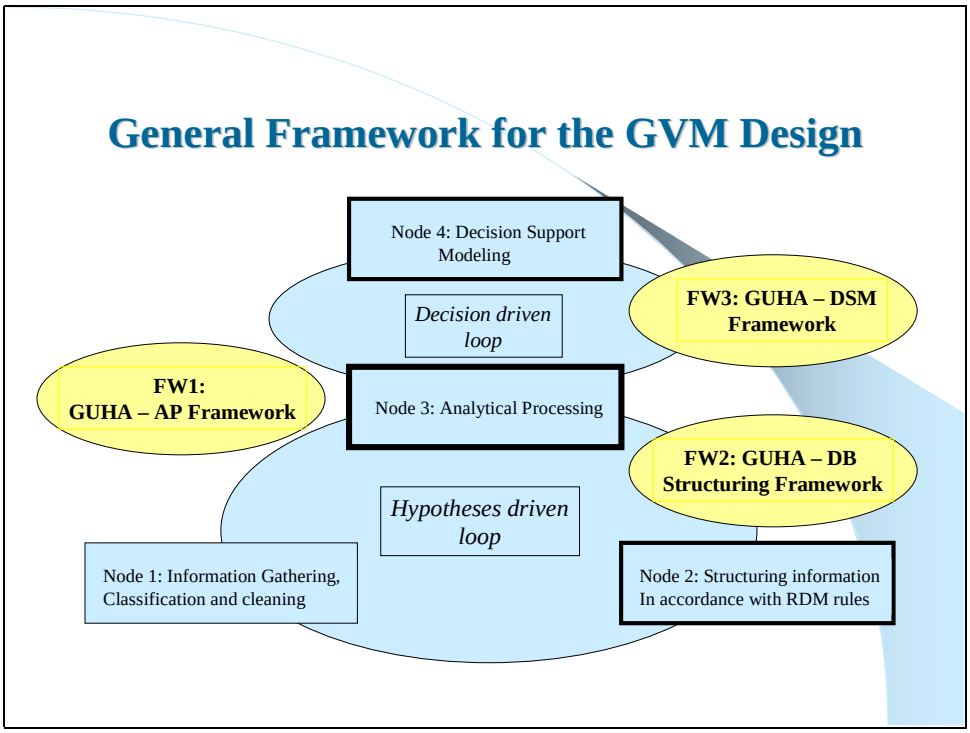


Fig. 2.1.

The second loop (decision driven loop) includes two processing nodes:

- Node 3: Analytical Processing
- Node 4: Decision Support Modeling

There are three particular frameworks within the General Framework which allow us to elaborate with GUHA components design:

- FW1: GUHA – Analytical Processing (AP) Framework,

- FW2: GUHA – Database (DB) Structuring Framework,
- FW3: GUHA – Decision Support Modeling (DSM) Framework.

2.2. The Analytical Processing Framework

2.2.1. A brief vocabulary

The most important terms using in the analytical processing are briefly explained.

M1: Dimension is a list of labels that can be used to cross-tabulate values from other dimensions
M2: Measure is a summarizable numerical value used to monitor business activity
M3: Member is a single item within a dimension
M4: Hierarchy means levels of aggregation within a single dimension
M5: Level means a layer of aggregation within a dimension hierarchy
M6: Aggregation means summarized values of a measure
M7: Fact table means the relational database table that contains values for one or more measures at the lowest level of detail for one or more dimensions
M8: Cube means the conceptual container of detail values from a single fact table, along with all possible aggregations for one or more dimension hierarchies.

2.2.2. Dimensions in Data Analysis

In the world of data warehousing, a summarizable numerical value that we use to monitor our business is called a *measure*. When looking for numerical information, our first question is which measure we want to see. We could look at, say, Sales Dollars, at Shipment Units, at Defects Per Hour, or at Ad Campaign Responses. Suppose that we ask to see a report of the company's Units Sold.
Here's what we get:

113

Looking at the one value doesn't tell us much. We want to break it out into something more informative. For example, how has the company done over time? We ask for a monthly analysis, and here's the new report:

January	February	March	April
14	41	33	25

The company has been operating for four months, so across the topy of the report we find four labels for the months. Rather than the one value we had before, we now find four values. The months subdivide the original value. The new number of values equals the number of months. This is analogous to calculating linear distances in the physical world: the length of a line is simply the length.
Because the company sells more products, we enhance our table by rows:

	January	February	March	April
Colony Blueberry Muffins			6	17
Colony Cranberry Muffins	6	16	6	8
Sphinx Bagels	8	25	21	

If the company sells three products, so down the left side of the report are the three product names. Each product subdivides the monthly values. Mean while, the four labels for the months are still across the top of the report. We now have 12 values to consider. The number of values equals the number of products times the number of months. This is analogous to calculating the area of a rectangle in the physical world: area equals the rectangle's length times its width. The report even looks like a rectangle.

The comparison to a rectangle, however, applies only to the arithmetic involved, not to the shape of the report. Our report could be organized differently- it could just as easily look like this:

Colony Blueberry Muffins	January	
Colony Blueberry Muffins	February	
Colony Blueberry Muffins	March	6
Colony Blueberry Muffins	April	17
Colony Cranberry Muffins	January	6
Colony Cranberry Muffins	February	16
Colony Cranberry Muffins	March	6
Colony Cranberry Muffins	April	8
Sphinx Bagels	January	8
Sphinx Bagels	February	25
Sphinx Bagels	March	21
Sphinx Bagels	April	

Whether we display the values in a list like the one above (where the numerical values form a line) or display them in a grid (where they form a rectangle): we still have the potential for 12 values if we have four monthly values for each of three products. Our report has 12 potential values because the products and the months are independent. Each product gets its own sales value- even if that value is zero-for each month.

Suppose that our company sells in two different states and we'd like to know how each product is doing each month in each state. Adding another set of labels indicating the states our company uses, and we get a new report, one that looks like this:

		January	**February**	**March**	**April**
WA	Colony Blueberry Muffins			30	10
	Colony Cranberry Muffins	3	16	6	
	Sphinx Bagels	4	16	6	
OR	Colony Blueberry Muffins			3	7
	Colony Cranberry Muffins	3			8
	Sphinx Bagels	4	9	I5	

The report now has two labels for the states, three labels for products (each shown twice), and four labels for months. It has the potential for showing 24 values, even if some of those value cells are blank. The number of potential values equals the number of states times the number of products times the number of months. This is analogous to calculating the volume of a cube in the physical world: volume equals the length of the cube times its width times its height.
Our report doesn't really look like a cube-it looks more like a rectangle. Again, we could rearrange it to look like a list, and the beginning of the list would look like this:

WA	Colony Blueberry Muffins	January	
WA	Colony Cranberry Muffins	January	3
WA	Sphinx Bagels	January	4
OR	Colony Blueberry Muffins	January	

OR	Colony Cranberry Muffins	January	3
OR	Sphinx Bagels	January	4
WA	Colony Blueberry Muffins	February	
WA	Colony Cranberry Muffins	February	16
WA	Sphinx Bagels	February	16

Whichever way we lay out our report, it has three independent lists of labels, and the total number of potential values in the report equals the number of unique items in the first independent list of labels (for example, two states) times the number of unique items in the second independent list of labels (three products) times the number of unique items in the third independent list of labels (four months). Because the phrase independent list of labels is wordy, and because the arithmetic used to calculate the number of potential values in the report is identical to the arithmetic used to calculate length, area, and volume-measurements of spatial extension-in place of independent list of labels, data warehouse designers borrow the term dimension from mathematics. Remember that this is a borrowed term.
A data analysis dimension is very different from a physical dimension. Thus, our report has three dimensions-State, Product, and Time-and the report's number of values equals the number of items in the first dimension times the number of items in the second dimension, and so forth. Using the term dimension doesn't say anything about how the labels and values are displayed in a report or even about how they should be stored in a database.

Each time we've created a new dimension, the items in that dimension have conceptually related to one another-for example, they are all products, or they are all dates. Accordingly, items in a dimension are called members of that dimension.

Now complicate the report even more. Perhaps we want to see dollars as well as units. We get a new report that looks like this:

		January		**February**		**March**		**April**	
		U	**$**	**U**	**$**	**U**	**$**	**U**	**$**
WA	Colony Blueberry Muffins					3	7 .44	10	24.80
	Colony Cranberry Muffins	3	7.95	16	42.40	6	15.90		
	Sphinx Bagels	4	7.32	16	29.28	6	10.98		
OR	Colony Blueberry Muffins					3	7 .44	7	17.36
	Colony Cranberry Muffins	3	7.95					8	21.20
	Sphinx Bagels	4	7.32	9	16.47	15	27 .45		

U=Units; $=Dollars

Because units and dollars are independent of the State, Product, and Time dimensions, they form what we can think of as a new, fourth dimension, which we could call a Measures dimension.
The number of values in the report still equals the product of the number of members in each dimension:
2 times 3 times 4 times 2, which equals 48. But there is not-and there does not need to be-any kind of physical world analogue. Remember that the word dimension is simply a convenient way of saying independent list of labels, and having four (or twenty or sixty) independent lists is just as easy as having three. It just makes the report bigger.

In the physical world, the object we're measuring changes depending on how many dimensions there are. For example, a one-dimensional inch is a linear inch, but a two-dimensional inch is a square inch, and a three-dimensional inch is a cubic inch. A cubic inch is a completely different object from a square inch or a linear inch. In our report, however, the object that our measure as our add dimensions is always the same: a numerical value. There is no difference between a numerical value in a "four-dimensional" report and a numerical value in a "one-dimensional" report. In the reporting world, an additional dimension simply creates a new, independent way to subdivide a measure.

Although adding a fourth or fifth dimension to a report is not a meta-physical act, that's not to say that adding a new dimension is trivial. Suppose that we start with a report with two dimensions: 30 products and 12 months, or 360 possible values. Adding three new members to the product dimension increases the number of values in the report to 396, a 10 percent increase. Adding a third dimension with three new members, however, increases the number of values in the report to 1080, a 300 percent increase.

2.2.3 Hierarchies in Data Analysis

When an organization is small or hasn't been around for a long time, it's easy to understand the business simply by looking at the detailed numbers. For example, if there is a company that has been in business for only four months and it has only three products, we can comfortably analyse data at the lowest level of detail. Psychological studies indicate that most people can easily comprehend about seven items-or seven groups of items [17]. Grouping-aggregating is the way that humans deal with numerous items. Once the company has sold products for more than six months, we'll undoubtedly want to start looking at the values by quarter as well as by month. Likewise, once the company has more than a dozen products, we'll probably want to group the products into product lines or product groups. But how do aggregations such as quarters and product lines fit into dimensions?

Generally, we think of members in a dimension as "belonging together." January and February naturally seem to belong together and clearly should reside in the same dimension. January and Colony Blueberry Muffins don't naturally belong together and clearly should not reside in the same dimension. But what about the members January and Qtr1? Do they belong together?

Remember that a dimension is really an independent list of labels for the report. To decide whether new members belong in a new dimension or in an existing dimension, imagine the new members as the column headings of a report, with the members of the existing dimension forming the row headings. If the new members are independent, we should-at least potentially-have a value each time they intersect. But look at a report that shows months on the rows and quarters on the columns:

	Qtr1	**Qtr2**
January	14	
February	41	
March	33	
April		25
May		29
June		39

Half the cells are empty, and it's not coincidental. There is no such thing as a January in Qtr2, just as there's no April in Qtr1. The report looks silly. Putting the two quarters in a dimension other than the Time dimension multiplies the total number of values by two but also guarantees that half of the values will always be empty. So the number of true potential values has not changed. In fact, the report with quarters on the columns never shows us what we want to see, which is the total for each quarter.

The report we want looks more like this:

Qtr1	88
January	14
February	41
March	33
Qtr2	93
April	25
May	29

June	39

Months and quarters are not completely independent members and should not appear in separate dimensions. The quarter totals are simply aggregations of the month totals, and they belong in the same dimension. There is, however, something different between Qtr1 and January. For one thing, we probably want the *January* label to be indented more than the Qtr1 label. And there's something similar about the labels Qtr1 and Qtr2.

Even though the words Month and Quarter don't appear in the report, we naturally refer to the labels January, February, and so forth as months, and we refer to the labels Qtr1, Qtr2, and so forth as quarters. Months and quarters form a hierarchy within the Time dimension, and each degree of summarization is referred to as a level. For example, in this Time dimension, January and February are members of the Month level, and Qtr1 and Qtr2 are members of the Quarter level. As time goes on and we add more months and quarters, we'll eventually add a Year level to the dimension's hierarchy. A dimension containing more than a few members almost always breaks into a hierarchy, and a hierarchy, by definition, contains levels.

When we have a hierarchy in a dimension, sometimes we'll want to see the entire hierarchy sometimes we'll want to see only the top one or two levels, and sometimes we'll want to see only the lowest level. We can use the term - members to describe either aIl the members in the entire dimension or only the members of a specific level within the dimension. For example, the members of the Time dimension include years and quarters as well as months, and the members of the Months level within the Time dimension do not include any years or quarters. The members at the lowest level of detail are called leaf members. A dimension cannot exist without leaf members, but it is possible to have a dimension with nothing but leaf members-that is, with only one level. For example, in the Measures dimension, it doesn't make sense to sum the total of Units and Dollars.

Some hierarchies, such as Time, are balanced: if there are months under Qtr1, there will also be months under Qtr2, Qtr3, and Qtr4. In a balanced hierarchy, it's easy to give names to levels. For example, the levels in a typical Time hierarchy might have the names Year, Quarter, and Month.

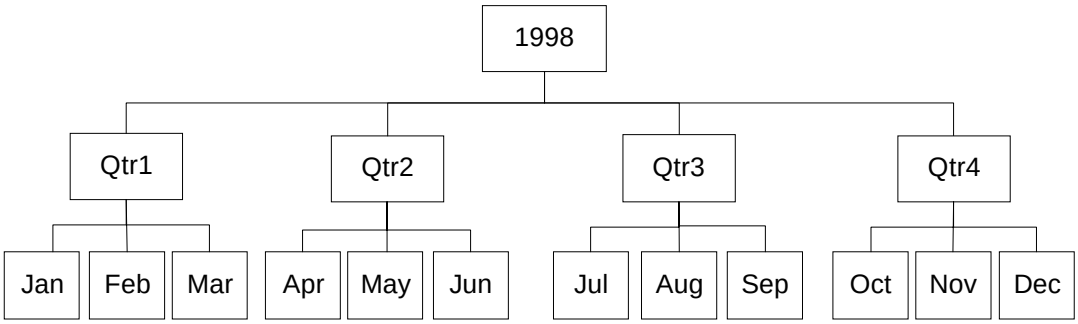


Fig.2.2

Some hierarchies unbalanced. An organization chart is often unbalanced. For example, in many companies, there might be many more people-and thus many more levels of management-in the Manufacturing organization than in Human Resources. In an unbalanced hierarchy, it's often difficult to give names to specific levels, but leaf members are always the ones that have no children below them.

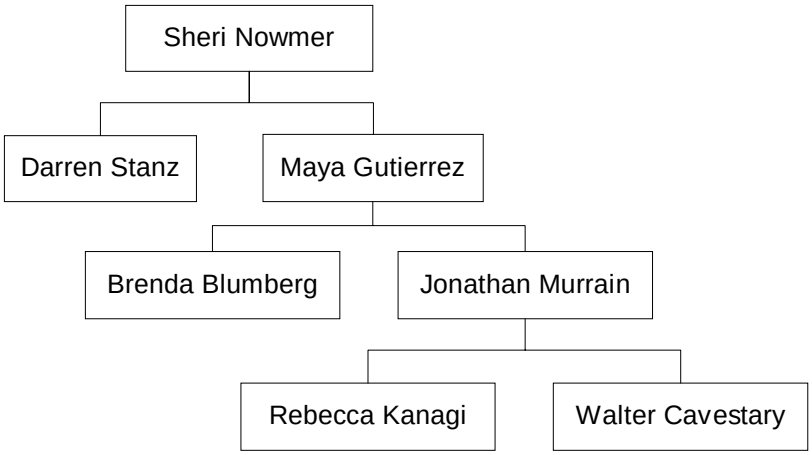


Fig. 2.3

Some hierarchies appear to blur the distinction between unbalanced and 4 balanced. For example, in a geographic hierarchy, you might have easily named levels-Country, Region, and State-but skip the Region level for certain States. J This really is a balanced hierarchy (because there are easily named levels), but the parents of some of the members are missing or invisible. A hierarchy that hides some of the parent members is called a ragged hierarchy.

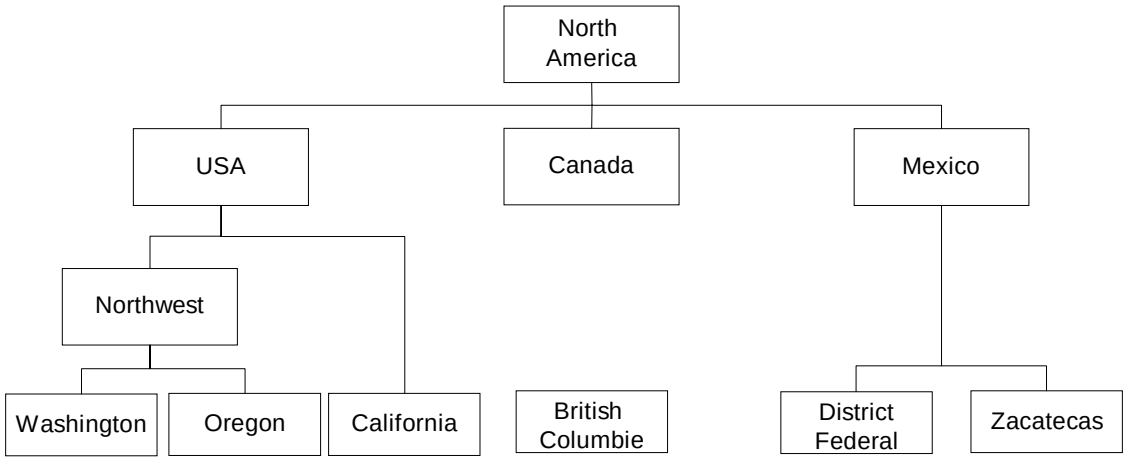


Fig. 2.4.

Analysis Services gives us a great deal of flexibility in defining balanced or unbalanced hierarchies, whether ragged or not. A dimension always has leaf members. The hierarchy simply defines how (and whether) the values for leaf members are summarized.

2.2.4. The Structure of a Data Warehouse

Analysis Services makes it easy for a client application to create reports that use multiple dimensions, but the values displayed in the report ultimately come from a relational data warehouse. Analysis Services assumes that we already have a relational data warehouse.
Analysis Services requires a data warehouse with a very specific form-a form characterized by a fact table.

A fact table is a table in the relational data warehouse that stores the detailed In a detailed for measures, or facts. A fact table that stores Dollars and Units by State, by Product, and by Month has five columns, conceptually similar to those in the following sample:

State	Product	Month	Units	Dollars
WA	Colony Cranberry Muffins	January	3	7.95
WA	Sphinx Bagels	January	4	7.32
OR	Colony Cranberry Muffins	January	3	7.95
OR	Sphinx Bagels	January	4	7.32
WA	Colony Cranberry Muffins	February	16	42.40

In these sample rows from a fact table, the first three columns-State, Product, and Month-are key columns. The remaining two columns-Units and Dollars-contain measure values. Each column in a fact table should be either a key or a measure.

To be usable by Analysis Services, the fact table must contain a column for each measure. A Sales warehouse might contain two measure columns-one for Dollars and one for Units. A shop-floor warehouse might contain three measure columns-one for Units, one for Minutes, and one for Defects. In a report, we can think of the measures as forming a separate dimension. That is, we can put Units and Crones/Dollars side by side as column headings, or we can put Units and Crones/Dollars as row headings. In the fact table, however, each measure appears as a separate column.

To be usable by Analysis Services, the fact table must contain rows at the lowest level of detail we might want to retrieve for a measure. In other words, the fact table contains rows only for leaf members of each dimension. Analysis Services cannot use a fact table that stores aggregates, such as quarter and year totals. For example, if a State dimension includes a hierarchy, consisting of State, Region, and Country, only the members from the State level appear in the fact table. Analysis Services will create all the summarized values. In the fact table, specifying a single leaf member for each dimension should identify a single row.

The sample rows in the preceding table illustrate the conceptual layout of a fact table. Actually, a fact table almost always uses an integer key for each member, rather than a descriptive name. Because a fact table tends to include an incredible number of rows in a reasonably large warehouse, the fact table might easily have millions of rows-using an integer key can substantially reduce the size of the fact table. The key column for a date dimension might be either an integer key or a Date value. The actual layout of a fact table might look more like that of the following sample rows:

STATE-ID	PROD-ID	Month	Sales-Units	Sales-Dollars
1	589	1/1/1998	3	7.95
1	1218	1/1/1998	4	7.32
2	589	1/1/1998	3	7.95
2	1218	1/1/1998	4	7.32
1	589	2/1/1998	16	42.40

2.3. The Structuring Framework

This framework relates to the data models. We shall investigate three data models, starting from the most general up to the most specialized.

2.3.1. The Data model

We assume that Data model has the following parts) see Fig. 2.5) /10/:

a) A domain D of interest
b) A system E, which consists of a body of data and relations among data, called an *empirical system*, and a mapping e: D -> E, called operationalisation
c) A representation system M (called a *numerical* or *graphical system*), and a mapping m: E -> M, called scaling which maps the data and the relations among the data to a numerical or graphical scale.
d) The researcher (the Domain Expert)

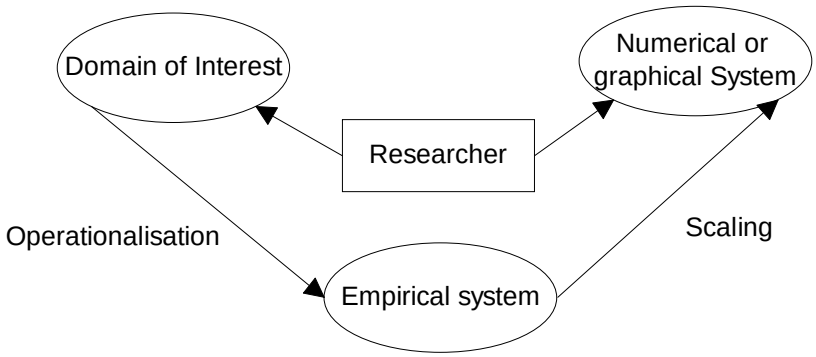


Fig. 2.5.

The Data model may be used for more deeper investigation based on the application of a Rough set data analysis approach.

2.3.2. The OLAP model

This model takes into consideration multidimensional nature of the OLAP storage unit – the cube - and its relationships to the flat tables, derived from OLTP tables. Derivation process relates to the KDD 3 (see paragraph 1. ..).
Regardless of the source tables, what's important is that the derivation process guarantees that the integrity of the data is maintained and that nonsensical data never gets into any of the target tables.
The required integrity must be achieved on the RDBMS level before transition data for analytical processing. (Node 2 in the hypotheses driven loop is responsible for it (see Fig. 2.1.)).
The normalization process helps us to create data structures that don't require the same data to be entered multiple times. An example of the well structured tables is in the Fig. 2.6. This part of database was built to support customer ordering . It includes Customers table, an Orders table, an Order Details table, an Products table and an Categories table. Updates of Customer relevant information don't touch Products and vice versa. The intelligent use of references reduces redundant data entry and makes the model efficient. Figure 2.6 shows how the efficiency of this model increases its complexity, making it somewhat more difficult for users to understand.

There is also another site effect of a normalized database – they are complex queries. *It seems to be specially interesting in a GUHA context, because queries are the necessary mechanism how to retrieve data for hypotheses.*

Let 's say that we want to use a SQL query to retrieve all the names of customers who purchased seedlings in April, 1998, along with the quantity and date of each purchase.
The query would look something like this:

```
SELECT
        cst.CompanyName,
        ord.ShippedDate,
        prd.ProductName,
        COUNT(1) AS qty
FROM customers cst
        JOIN [orders] ord
                ON cst.CustomersId = ord.CustomerID
        JOIN [order details] det
                ON det.ordered = ord.ordered
        JOIN [products] prd
                ON det.productid = prd.productid
WHERE ord.ShippedDate BETWEEN '4/1/1998' AND '5/1/1998'
AND prd.productname = 'seedlings'
GROUP BY
        cst.CompanyName,
        ord.ShippedDate,
        prd.ProductName
```

Query like this seems to be complicated, but there is no other way how to retrieve information from normalized database.
The alternative approach is to create new structure, including all required information in one table. Process like this is known as denormalizing.
When the database is denormalized, the query shown in the earlier example looks like this.

```
SELECT
        CompanyName,
        ShippedDate,
        ProductName,
        Count(1) AS qty
FROM ProductPurchaces
        WHERE ShippedDate BETWEEN 4/1/1998' AND '5/1/1998'
AND prd.productname = 'seedlings'
GROUP BY
        cst.CompanyName,
        ord.ShippedDate,
        prd.ProductName
```

By denormalizing the tables, we eliminate the complexities, associated with joins. The downsize of this process is that it increases data redundancy.

This is sufficient place to introduce OLAP based approach. Example of the one particular OLAP model is in the Figure 2.7. It was design as Sales cube. Keeping all source tables in normalized view, we enhance our model with special table called **fact table** (see sales_fact_1997).

Structure like this are the basic platform for OLAP models. Working with this structure we need new language – Multidimensional Expression (MDX).

2.3.3. The OLAP model with Data Mining (DM) support

This paragraph describes Data Mining (DM) in a context of the currently used database systems. **We suppose further, that serious comparison and testing of the existing DM methods is very important for GVM design at least for the following reasons:**

a) **GUHA method helps us to find new relationships between data in comparison with currently used DM algorithms. Is it true ?**

b) What is the difference between GUHA and existing DM algorithms and how is it dependent on the data patterns ?
c) Is it possible (and reasonable) to try to combine GUHA and other DM algorithms ?

There are at least two reasons why DM algorithms can be useful for organization's stores of data: to make sense of their past and to make predictions about their future. The kind of information an organization wants determines the kind of model it builds and the algorithms applied to the structured data.

Directed Data Mining

In order to perform a directed data-mining operation, all the known factors, or input variables, need to be given so that the data-mining engine can find the best correlations of those attributes to a rule within the data-mining model. The prediction is based on unknown values or target variables, meaning that data-mining engine will find the most likely value for those missing values based on the known values provided with the input variables.
Directed data mining uses the most popular data-mining techniques and algorithms, such as decision trees. It classifies data for use in making predictions or estimate with the goal of deriving target values - in fact, it's the request for target values that gives directed data mining its "direction".

Undirected Data Mining

Because undirected data mining isn't used to make predictions, target values aren't required. Instead, the data is placed in a format that makes it easier for us to make sense of. For example, an online bookseller wanted to organize his book's list in accordance with reader's groups. The data could reveal that readers of science fiction are men with technical education, living in large and medium cities.
Clustering is the algorithm commonly used for mining historical data. Data are classified in accordance with it's properties and each group of similar objects are represented by a profile.
There is also possible to use clustering in combination with decision trees algorithm. In this case clustering is the only first step in the process used to define broad groups. Once groups are established, directed data mining is used on groups that are of particular interest.

GUHA and Directed / Undirected Data Mining

It is open question how the GUHA can be applied from this point of view. As we know, there is One serious problem with the GUHA database applications relates to the complexity of the task in dependency on number of attributes (literals) included in the antecedent and succedent
One possible way how to decrease this complexity is a combination of two methods – clustering as the first method and then the GUHA.

The OLAP model with Data Mining (DM) support takes into consideration multidimensional nature of the OLAP storage unit – the cube - and Data mining algorithm (Fig. 2.8). Decision trees algorithm was applied for a cube specified with the set of dimensions on the left site of the figure 2.8. The result of the union of Data mining algorithm and set of dimensions is a new OLAP model with DM support called "Customer Pattern Discovery". This model may be used for investigation of relationships between customer's cards and other customer's properties. Let say, we are interesting in customers who have the gold card (Fig. 2.9). We can see that the gold card owners dominant feature is the yearly income equal or higher to 150 000 $. Next step in this direction let us to know, that almost all "golden" customers are married (Fig. 2.10).

2.4. The Decision Support Modeling Framework

A typical problems with the GUHA outputs are:
- a lot of verified hypotheses,
- semantic interpretation of them (excluding trivial assertions)

To many analytical results are not acceptable by decision makers. They usually need a small set of comprehensive alternatives, with well understandable criteria, relevant to a decision process.

The GUHA – Decision Support Modeling framework relates to the General Framework for the GVM design in accordance with the Figure 2.11. Node 3 was described in previous paragraphs. We focus attention on the node 4, which is new.

Decision making in a complex environment requires knowledge to be continuously organized. Such organizing combines a deductive approach and a system (inductive) approach. The essence of organization relates to analyzing and structuring hierarchies. These hierarchies are divided into two types: structural and functional

In contrast, functional hierarchies decompose complex systems into their constituent parts in accordance to their essential relationships. Such functional hierarchies help people to steer a system toward a desired goal – like conflict resolution or efficient performance.

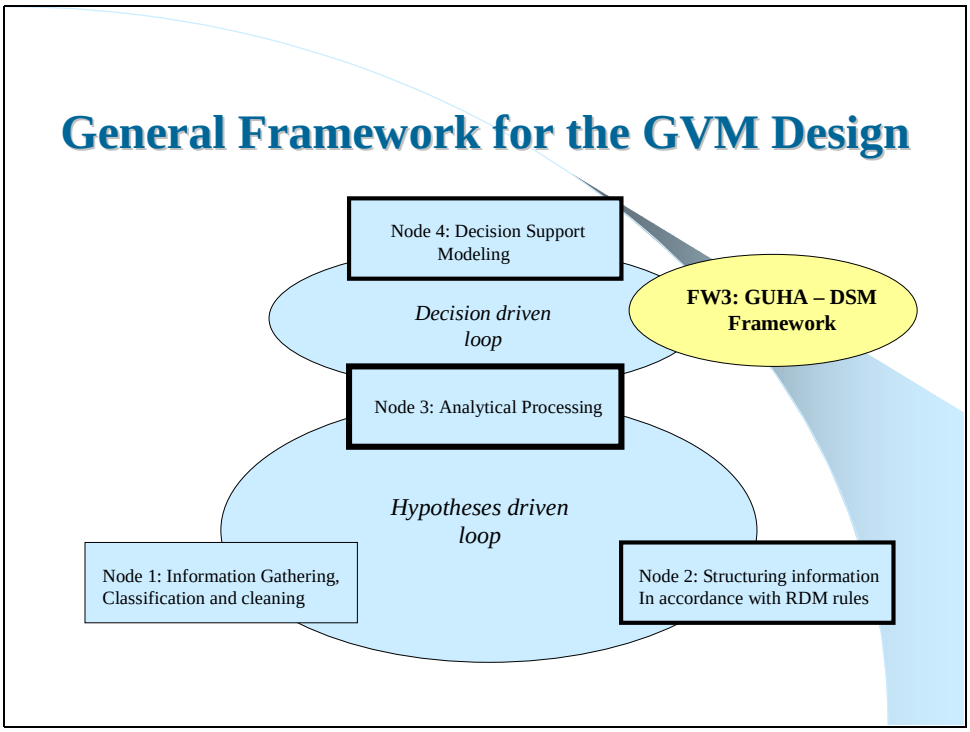


Fig. 2.11.

Typical simplified hierarchy structure using for decisions is in the Fig. 2.12. This structure includes three types of nodes. Objective node, a set of Criteria nodes (which may include a hierarchy of sub-criteria) and Alternative nodes.

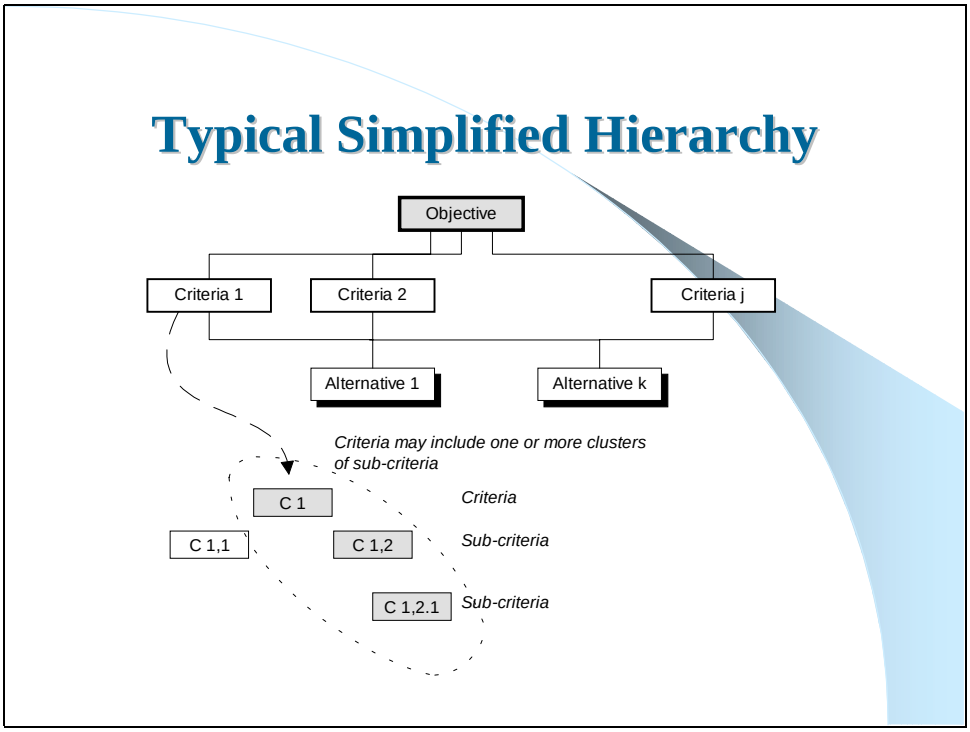


Fig. 2.12.

There are some important rules, which must be applied during a design of a Hierarchy structure:

Rule 1: Hierarchical Levels Templates: These templates include recommended levels like "uncontrollable environmental constrains", "risk scenarios", "controllable systematic constrains", "overall objectives of the system", "actors", "actors objectives", "exploratory scenarios", and "composite scenarios".
Rule 2: A set of recommended steps for a design of a hierarchy: This rule includes steps like "identify the overall goal", "identify the sub-goals", "identify criteria", "identify sub-criteria under each criterion", "identify the actors involved", and "identify the actors' goals". We can see that both rules include "hidden interface" to GUHA. These "hidden interfaces" are the "exploratory scenarios" in Rule 1 and the "identify sub-criteria under each criterion" in Rule 2.

There are two reasons why we are interesting in a usage of hypotheses(the GUHA) for a decision support:

- ***to understand what criteria are important for a decision process***
- ***to understand hierarchy and preferences..***

We explain the most important ideas of the decision process using samples. These samples relate to the research presented this year on the ISAHP symposium in Switzerland /11/.

Sample 1. Decision Support Tree
Decision Support Tree used for the choice of the optimum variant of management of radwaste from NPP's includes the Goal node and the first hierarchical level of criteria (C1, .. C4) (Fig. 2.13a). Criteria "Global Factors" include three lower level sub-criteria (Fig. 2.13b).

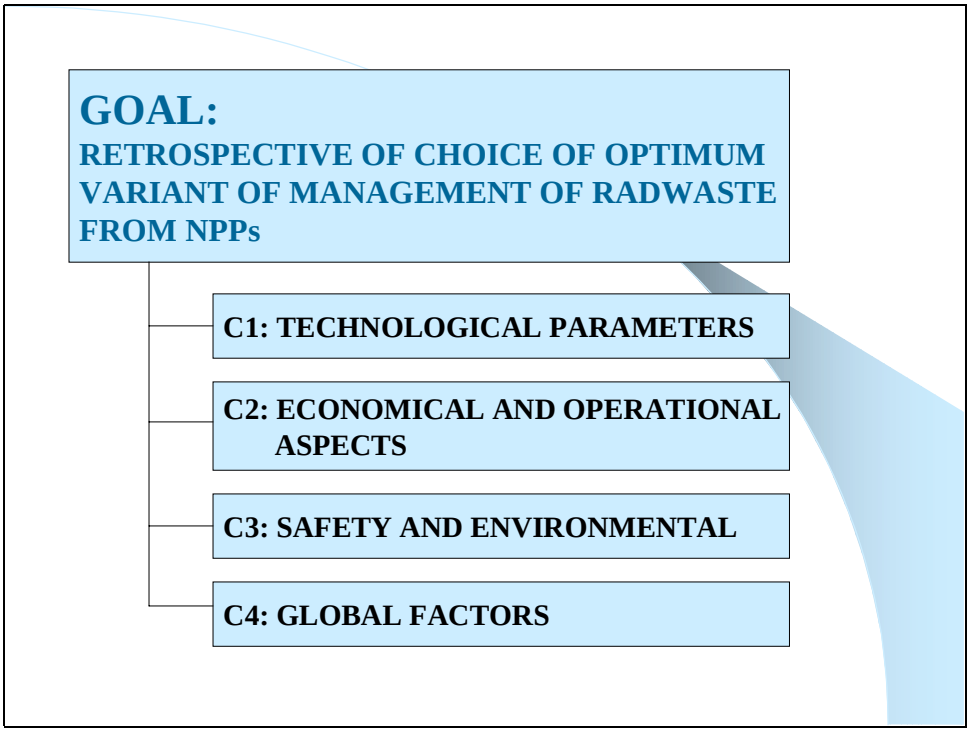


Fig. 2.13a.

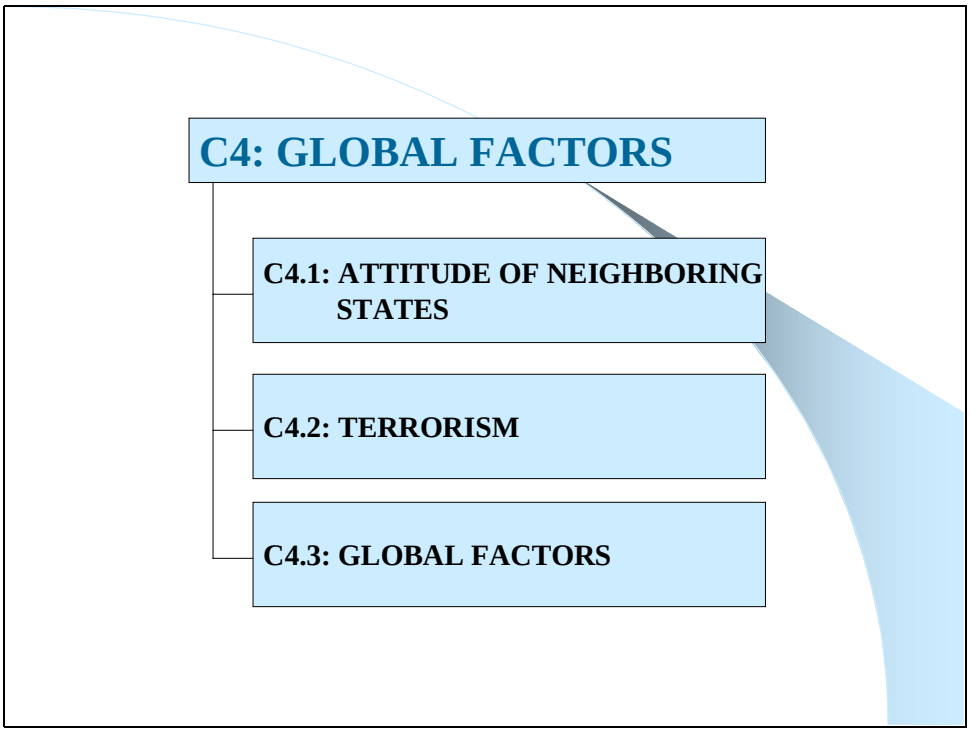


Fig. 2.13b.

Decision making process includes seven phases:

1. Problem identification and research
2. Eliminate Infeasible Alternatives
3. Build an AHP Model
4. Make Judgments
5. Synthesize
6. Examine and Verify the Decision
7. Present and Document the Decision

The first phase involves three components which seems to be interesting in a relation to the GUHA:

- Identify the problem
- Identify the objectives and alternatives
- Research the alternatives and criteria

The objectives and alternatives identification can be done by two ways:

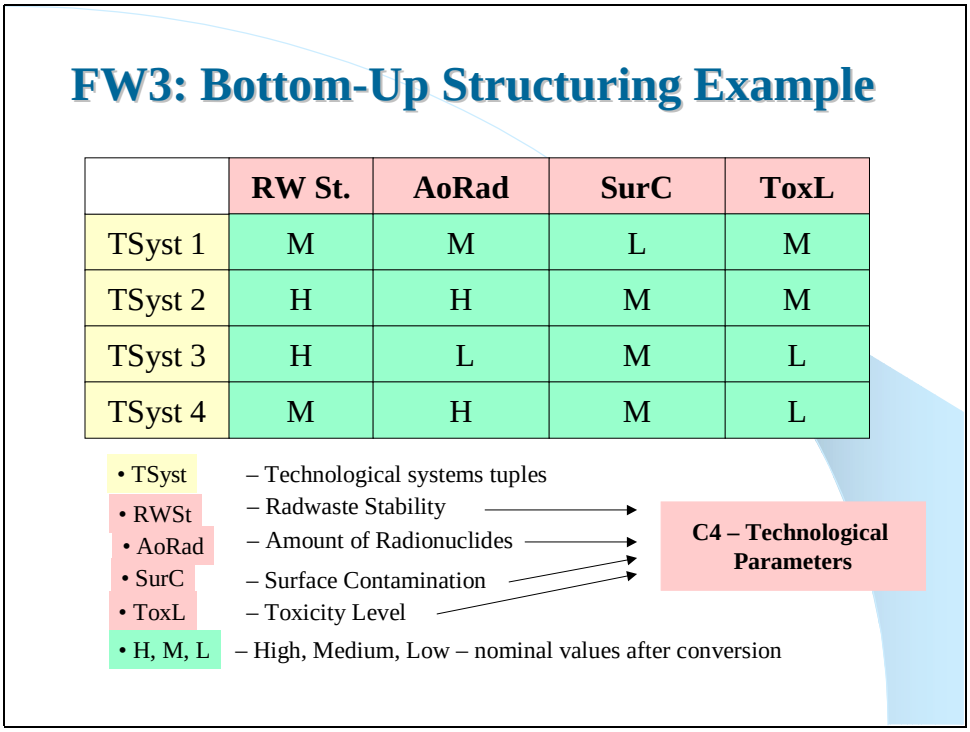
- using a top-down approach (objectives are identified firstly and then the alternatives)
- using a bottom-up approach (alternatives are identified firstly and then the objectives)

Starting with the Objectives Implies the Top-Down Structuring Feature
Top-down structuring is better suited for decisions of a strategic nature where the objectives are better understood or known than the alternatives. Top-level objectives are identified followed by the identification of sub-objectives.

Starting with the Alternatives Implies the Bottom-Up Structuring Feature
Bottom-up structuring is best suited for situations where the alternatives are better understood than the objectives. The pros and cons of the alternatives are used to help identify the objectives that can then be clustered into groups.

We should investigate an influence of each particular alternative to the appropriate criteria .This investigation should concern of alternative's properties.

Sample 2: Bottom-up Structuring
The technological parameters of alternatives (T Systems) are well understood (see tab. 2.1).



Tab. 2.1.

From a viewpoint of these parameters we are interesting to evaluate hypotheses about associations between parameters and a possibility, that radwaste could used by terrorists.

The GUHA method might be used for improvement of Top-down structuring when we investigate an influence of each particular T System to the C4.2: TERRORISM .

3. The GVM Concept

Almost 40 years of the GUHA method existence covers more than five generations of the computer machine's generations, staring with the classical SISD (Single Instruction Single Data) architectures typical for the majority of a mainframes in the sixties up to the current highly distributed component based architectures. Already in the 1980's during the development of the fifth generation computer systems /12/ there were developed three fundamental directions (inference machine, relational machine, abstract data type support machine) which covered knowledge architecture.
Many of ideas of those times strongly influenced a scientific research in the Czechoslovakia. Specially classification and relationship analysis /13,14,15/ were subject of a research in the analysis of a hidden information using various database architectures /16,17/. Even now, the techniques and algorithms developed at those time seems to be very effective to cover some analytical needs of a nowadays. Even more, they may be efficiently combined with hypotheses generations and decision support. The final result of a such "melting pot" strategy could be very progressive. But many depends on well understanding of a "thinking" pipelines in accordance with the Fig. 2.1. We try to divide the problem of the "Workflow - IT Components" mapping into two particular views – a Logical and a Physical.

The Logical View
This part of the GVM Concept describes functional aspects of the GUHA virtual machine. It captures IT realizable parts of workflows and converts them into functional and logical diagrams. The Logical View reflects the Architect's thinking and understanding.

The Physical View
This part of the GVM Concept describes implementation aspects of the GUHA virtual machine. It captures components and component's structures needed to cover functional aspects. The Physical View reflects the Designer's thinking and understanding.

In addition to these views there is the third view – Deployment – which covers all realizations of the particular pair of Concepts (Logical, Physical) in the particular IT environment. This part is beyond the scope of the scientific research and will be covered in dependency on the needs of the stages of the research project OC 274.001.

The combination of the first two views gives us necessary flexibility to overcome the gap between mathematical abstractions and the methods, which will be offered by the component infrastructure.

3.1. The GVM Logical View

Existing version of the GUHA method known as GUHA PC includes three modules (Fig.3.1).

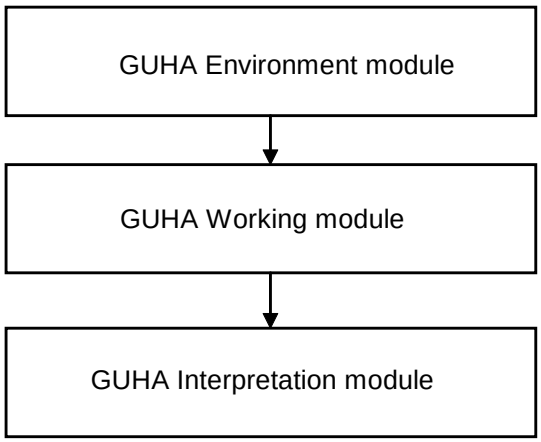


Fig. 3.1.

The first module (GUHA Environment) helps a user to prepare all necessary parameters which are needed for hypotheses processing. Source data for this module are taken from a flat relational tables or from a Excel spreadsheets. There is zero support of SQL language support for direct elaboration with

source data during their transformation from database (spreadsheet) format into GUHA matrix format. The second module (GUHA Working) supports generation of interesting hypotheses and their verification. Computational complexity of this GUHA module reflects the GUHA main principle, that there are generated ALL possible combinations of literals (expressions "<Attribute>, <Operator>, <Attribute's Value>) for Antecedent and Succedent. All relevant assertions is an output from this module. The third module (GUHA Interpretation) let a user to see all relevant assertions and to choose these assertions which are close enough to his decision needs.
The most important processing steps within GUHA Environment module are described in the Fig. 3.2. User interface is supported by the very simple GUHA control language. Parameters for the second GUHA module are generated as an output of the final phase of processing.

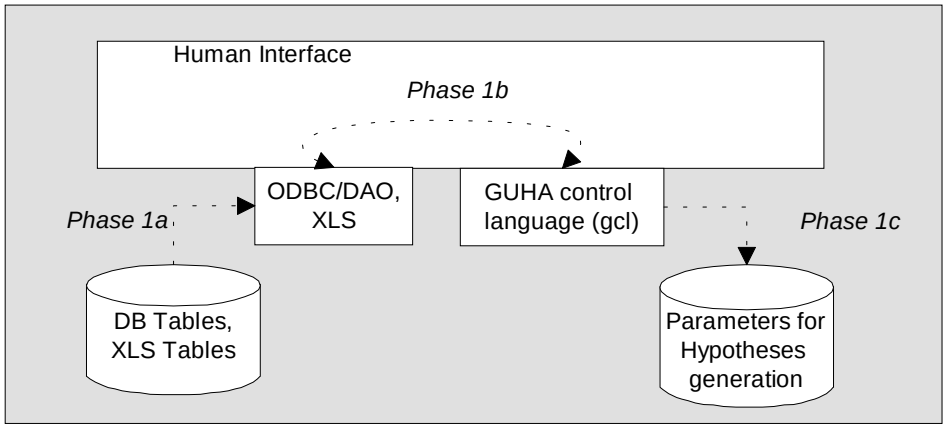


Fig. 3.2.

The GUHA working module (Fig.3.3) is responsible for hypotheses processing.

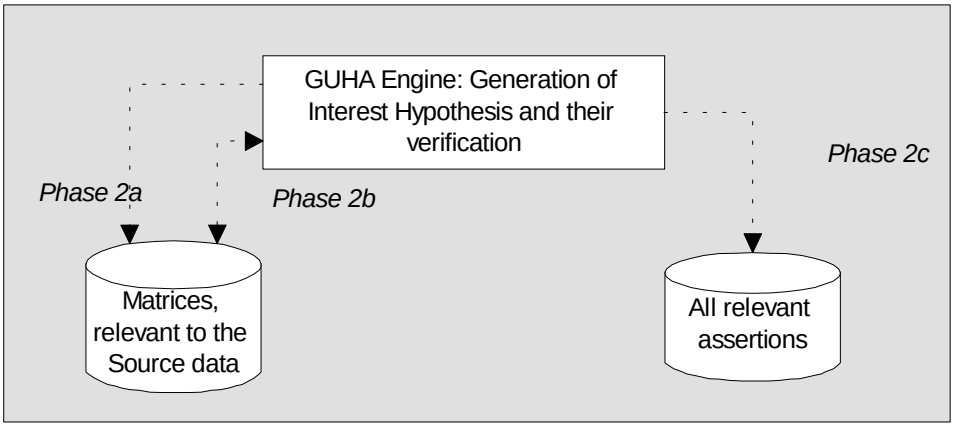


Fig. 3.3.

It is not simple to analyze a lot of assertions – for this purpose we use GUHA Interpretation module (Fig. 3.4). The problem is, that a form of the assertions is more matrix-oriented then SQL oriented. The impact of this is a cumbersome interface in comparison with interfaces used in a modern relational database systems.

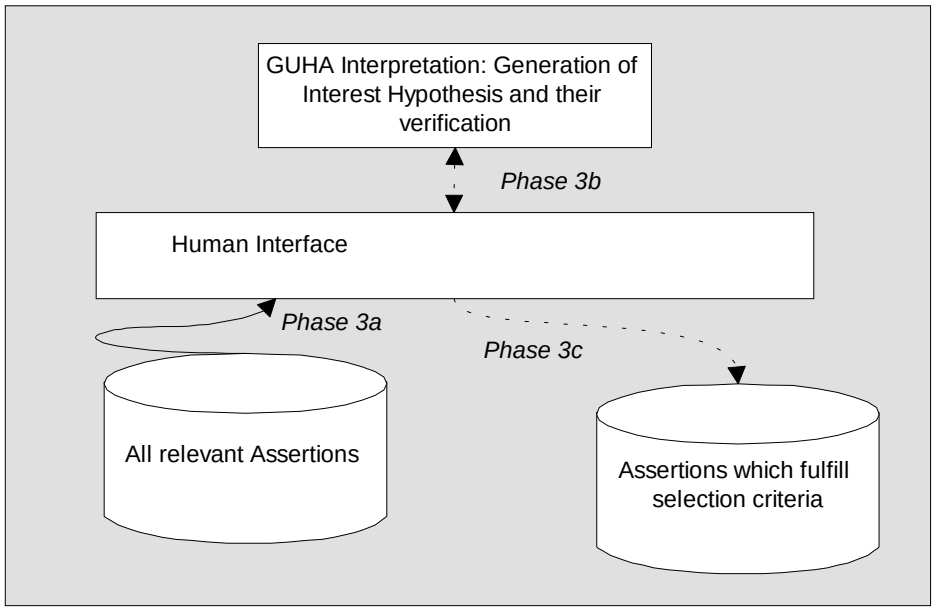


Fig. 3.4.

A vision of a GVM Concept unify two directions (Fig. 3.5). The first one is relevant to the GUHA theory and to the existing GUHA method implementations. The second directions would introduce new RDB mechanisms.

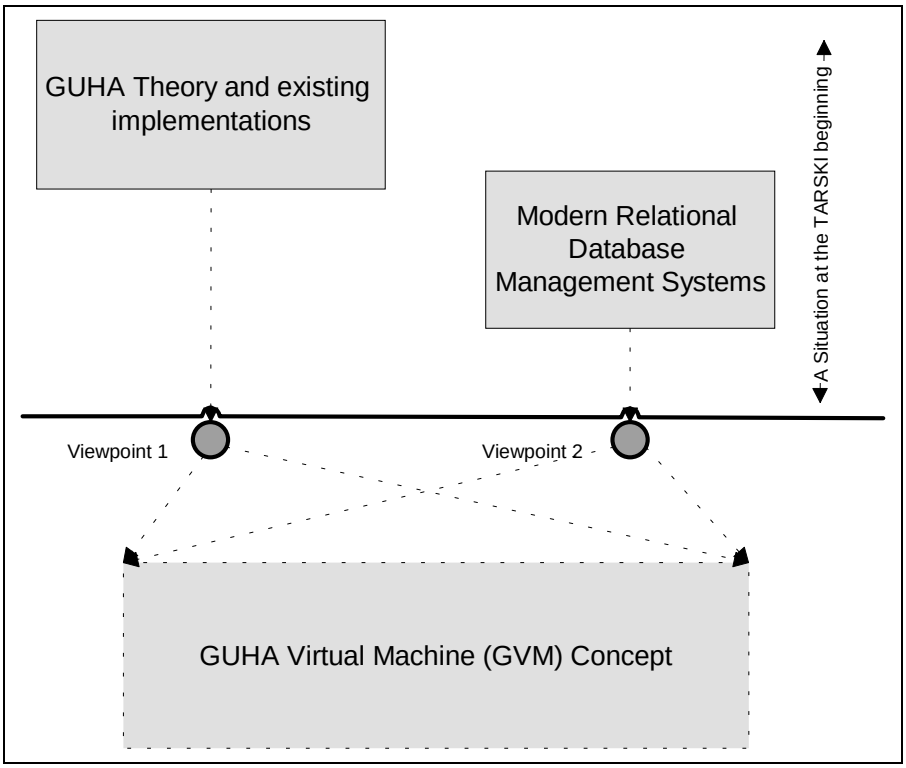


Fig. 3.5.

Fig. 3.6 fixes the current situation from a viewpoint of the most important areas, which will influence the GVM Concept and which are based on the GUHA original research, done in the Czech for almost forty years.

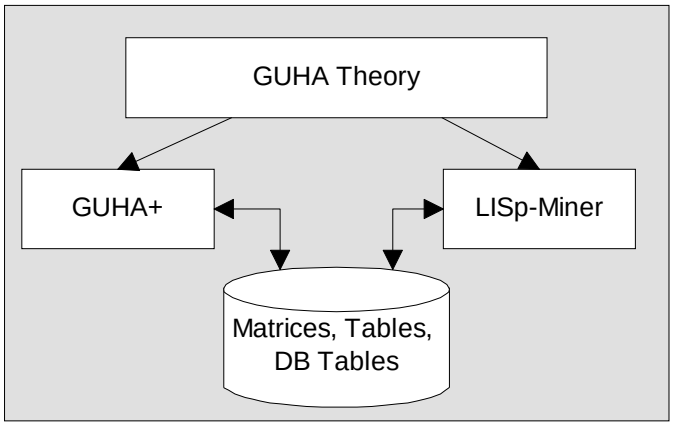


Fig. 3.6.

Fig. 3.7 fixes the starting situation from a viewpoint of the most important areas, which will influence the GVM Concept and which relate to the modern relational database systems. When we compare this figure with the figure 3.6 we can see that the most important difference captures two new engines – SQL and MDX.

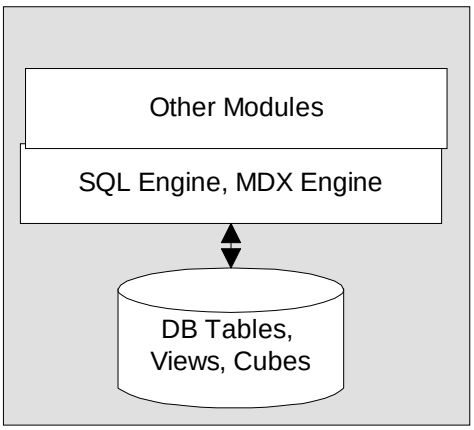


Fig. 3.7.

The first very general GVM Concept is in the Fig. 3.8. We distinguish four basic blocks. The GVM/HI will support a Human Interface. The GVM/DB will be responsible for all database oriented procedures and for exchange procedures between a database and other GVM blocks. The GVM/E block encapsulates all GUHA kernel procedures and internal data (matrices). The new block is a GVM/DSM. This block will be responsible for an interface to the decision support area outside a GVM Concept. This block will allow to control assertions choice in dependency on the needs of decision makers.
Each basic GVM block relates to the combination of working groups (WG). The particular working group number is the same as specified in the basic COST 274 document.

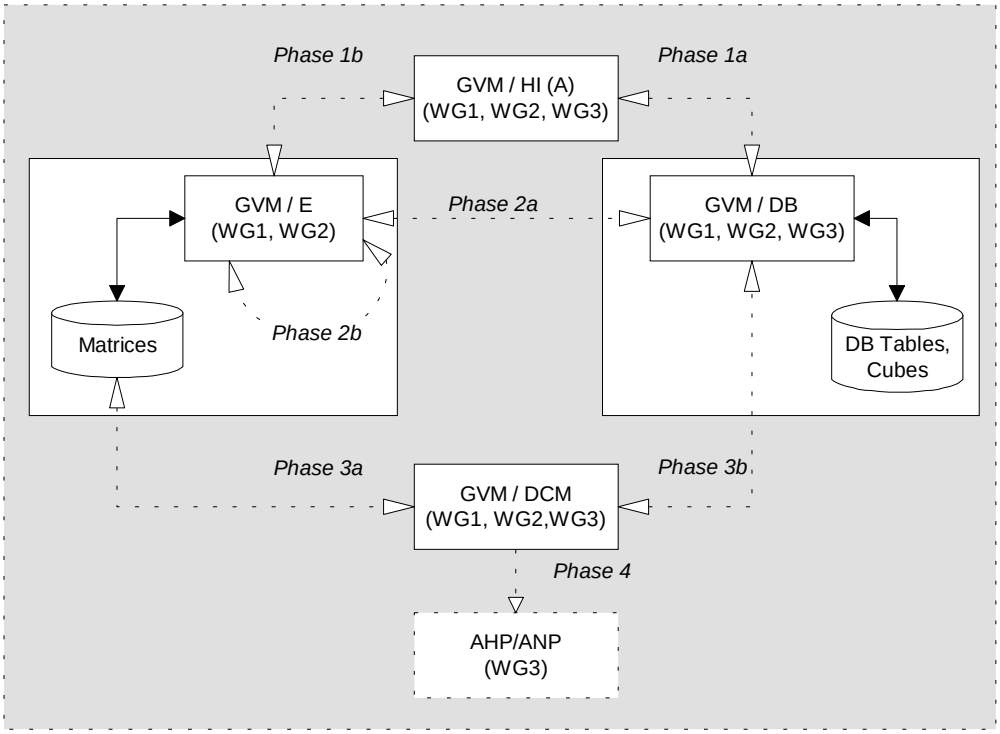


Fig. 3.8.

3.2. The GVM Physical View

The natural basement for a GVM implementation is a Architecture supporting analytical processing (Fig. 3.9). It helps to create new analytical space through OLAP cubes using Decision Support Objects (DSO).

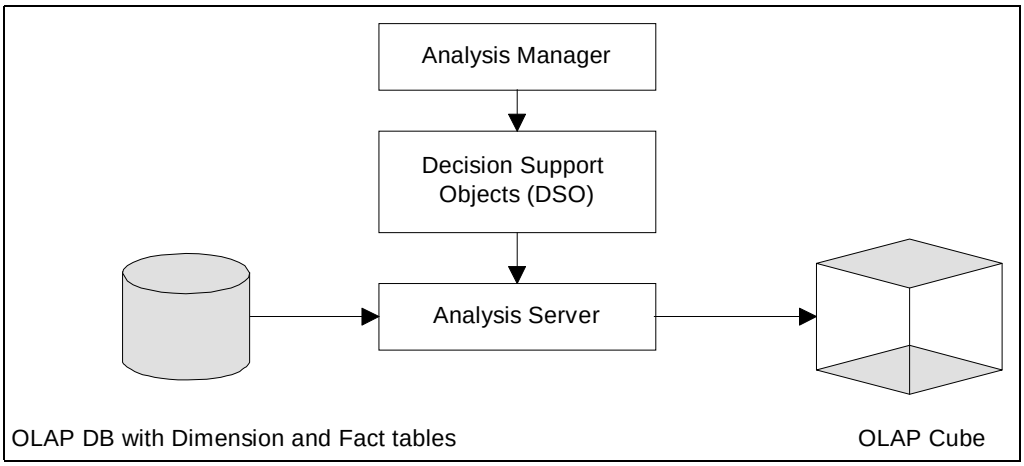


Fig. 3.9.

There are two phases of a Cube processing (Fig. 3.10). The first phase creates abstract structure consists of dimensions. The second phase (learning) fills cube structure with a data pattern. The result aggregates represents necessary analytical knowledge.

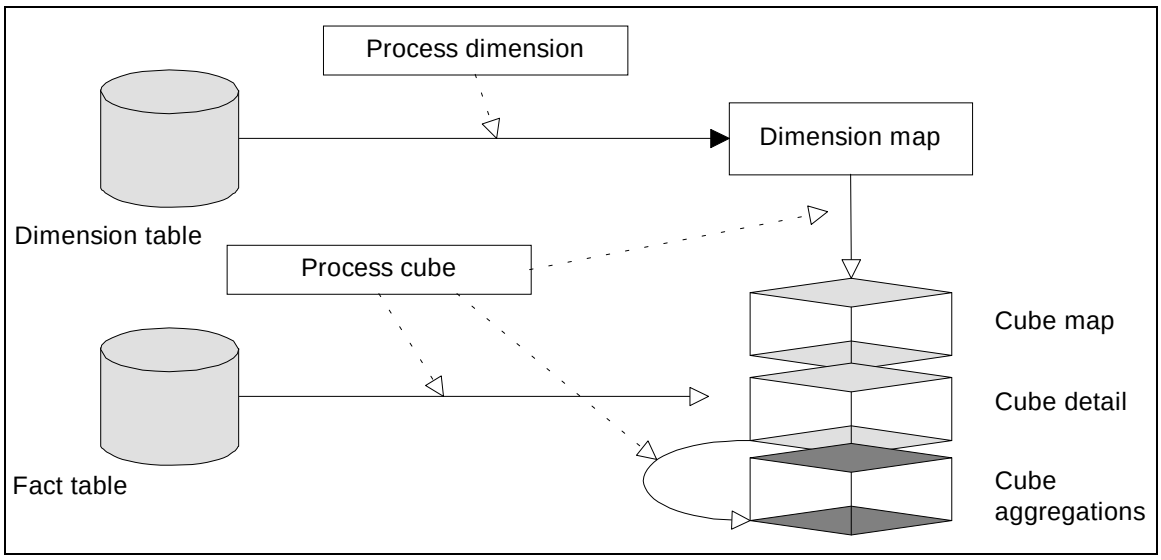


Fig. 3.10.

Till now the GUHA supports interface to the relational databases (RDB) missing analytical interface (Fig. 3.11). Even more, there is no special difference between OLTP and OLAP technologies, because the current GUHA doesn't take into account problems relating to the "dirty" data in the OLTP format. It is not too important for desktop databases. Quite different situation is in the server database.

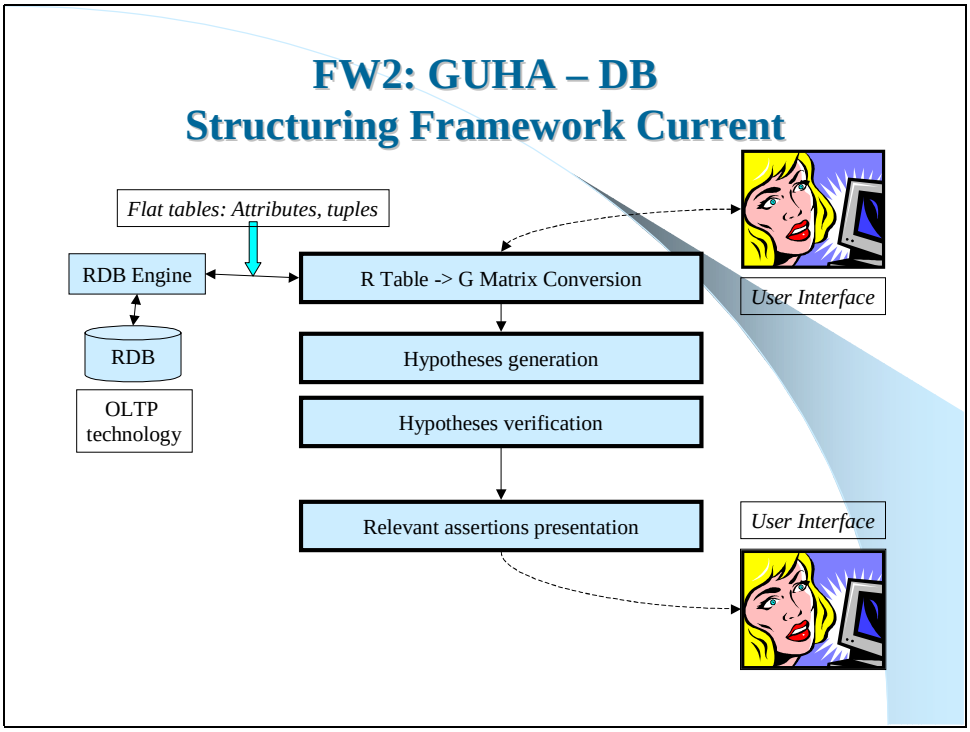


Fig. 3.11.

There are various ways how to replace "RDB" interface with a Analytical one. Two possibilities are demonstrate in the Fig. 3.12 and Fig. 3.13. The first possibility converts Cubes into GUHA matrix compatible format. Next processing phases are similar as in nowadays. The second possibility supposes a design of a client applications for the input part (with OLAP interface) and output part.

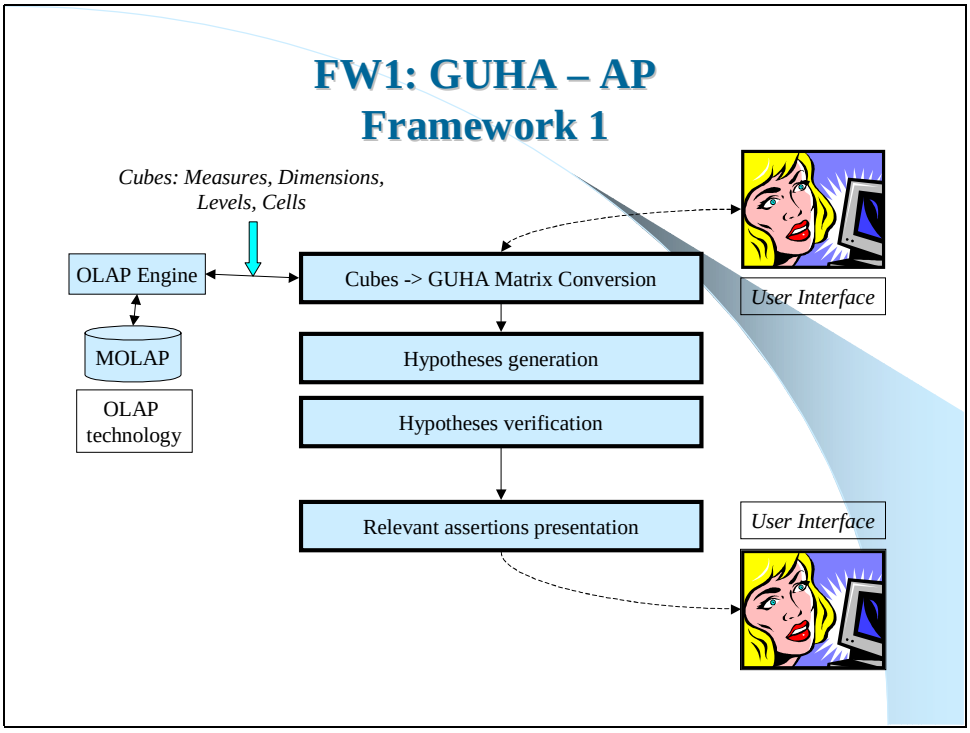


Fig. 3.12.

Still open is a distribution of GVM components between Client and Server site. This problems concern of the OLAP architecture (Fig. 3.14) and particularly the way how ADO MD will be used.

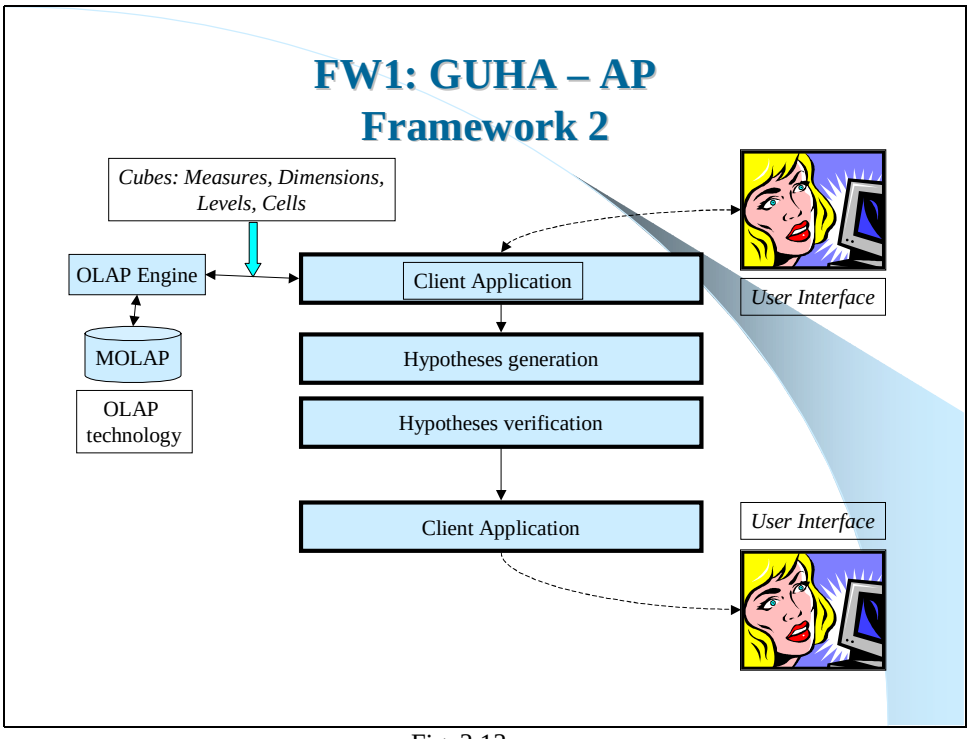


Fig. 3.13.

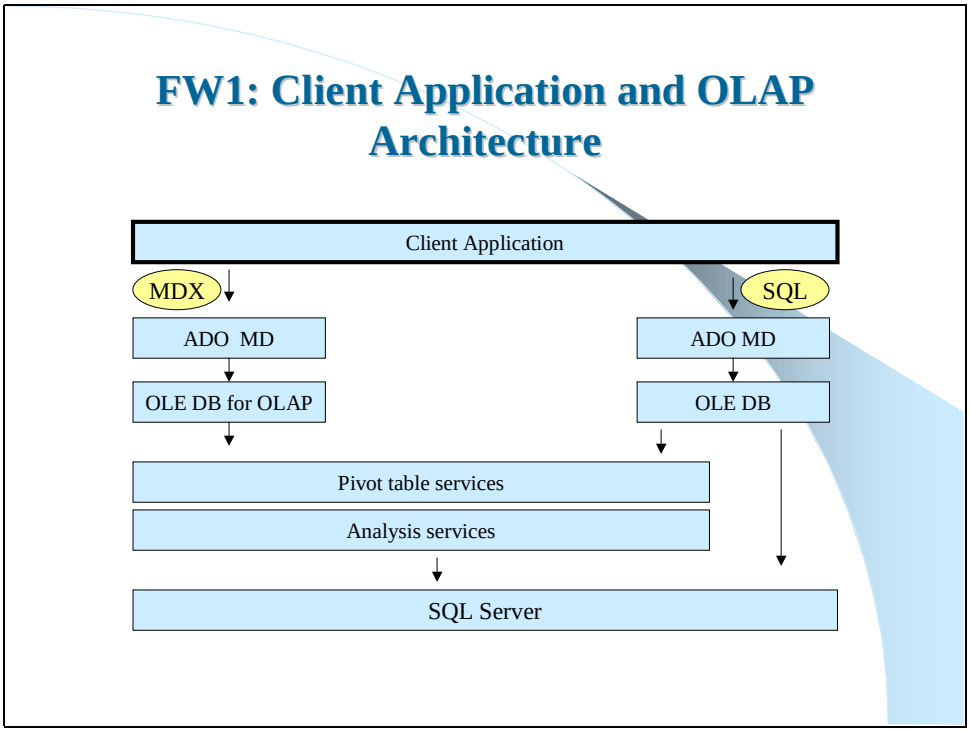


Fig. 3.14.

ADO MD includes set of very powerful objects for the Catalog maintenance, Cube and Dimensions definition, Hierarchy and Level modeling and for processing of aggregates (a Cellset, a Cell, a Axis, a Position) as it is in the Fig. 3.15.

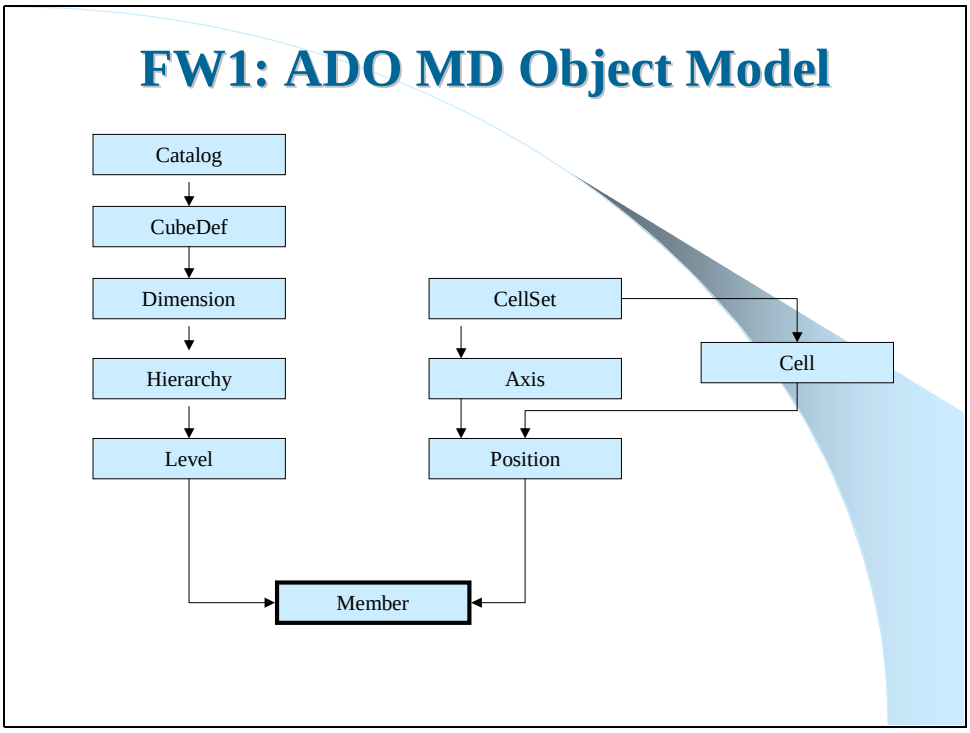


Fig. 3.15.

4. 2002 Research Objectives

The GVM Framework and the GVM Concept are loosely coupled research areas. The first one is necessary from a viewpoint of workflow understanding and an analytical tasks specification. The second is important from a viewpoint of architecture development and a component based design.
To cover both research areas we propose following list of research items (RI) for the 2002:

RI1: An improvement and enhancement of a GUHA kernel procedures (unknown values processing, new / modified quantifiers)

RI2: An improvement of a statistic's logs in accordance with a different data types processed by the GUHA kernel.

RI3: A research and development of a new interface, more suitable for analytical interpretation of hypotheses (the Logical View)

RI4: A research of an influence of database statistics received from various databases to the GUHA tasks specification.

RI5: A research of a possibility to use combination of Data mining techniques in dependency on database patterns.

References

/1/ Hajek,P,, Havel,I., Chytil,M. (1996): The GUHA method of automatic hypotheses determination. Computing, 1, 293-308

/2/ Hajek,P., Holena,M.: Formal logics of discovery and hypothesis formation by machine

/3/ Hajek,P. (2001): Relations in GUHA style data mining. Proceedings RalMICS'6 – TARSKI, Oisterwijkm the Niederlands, October 2001

/4/ Davidson,L. (2001): Professional SQL Server 2000 Database Design, Wrox Press Ltd., ISBN: 1-861004-7-61

/5/ Codd,E.F. (1985): Is your DBMS Really Relational ?, Computer World, October,14.

/6/ Codd,E.F. (1985): Does Your DBMS Run By the Rules?, Computer World, October,21.

/7/ Jacobson,R. (2001): Step by Step SQL Server 2000 Analysis Services, Microsoft Press, ISBN: 0-7356-0904-7.

/8/ Agrawal,R., Mammila,H., Srikant,R., Toivonnen,H., Verkamo,A.I. (1996): Fast Discovery of association rules. Advances in Knowledge Discovery and Data Mining, AAA Press/MIT, pp.307-328.

/9/ Zytkow,J.,M. (1999): The melting pot of Automated Discovery: Principles for a New Science. Proceedings of Discovery Science, Second International Conference, Tokyo, Japan, December 1999, pp. 1 – 12.

/10/ Duntsch,I., Gediga,G.: Rough set data analysis : A road to non-invasive knowledge discovery. ISBN: 1-903280-01-X.

/11/ Halova,J., Feglar,T. (2001): Systematic approach to the choice of optimum variant of radioactive waste management, ISAHP 2001m Berne, Switzerland, August 2-4.

/12/ T.Moto-Oka: The fifth generation computer systems - the Preliminary Report, Japan Information Processing Development Center, 1982, 108 pp.

/13/ Ashany,R.: Application of sparse matrix techniques to search, retrieval, classification and relationship analysis in large database systems – SPARCOM, Proc. VLDB-78, pp. 499 – 516.

/14/ Fushimi,S., Kitsuregawa,M., Tanaka,H., Moto-Oka,T.: Multidimensional Clustering Techniques for Large relational database machines, Proceedings of the International Conference on Foundations of Data Organization, May, Kyoto, Japan, pp.226 – 235

/15/ Kazumasa,O. (1985): A Stratificational Overlapping Cluster Schema, Pattern Recognition, Vol.18, No.3/4, pp. 279 – 286.

/16/ Feglar,T. (1986): The Usage of a Analysis of a Hidden Information for the query processing in a relational database system, PhD Dissertation, Institute of Technical Cybernetics of the Slovak Academy of Science, Bratislava, 112 p.

/17/ Feglar,T.,Zoc,I. (1989): Architectures of Database Processors, Information Systems, Research Institute of the Social-economic Information and automation in Management, Bratislava, V.18, No. 2/1989, pp. 143 – 159.

/18/ Saaty, T.L.: Fundamentals of Decision Making and Priority Theory", Vol. VI of the AHP Series, RWS Publications, Pittsburgh, USA, pp. 84-86.