CONSTELLATION DISCOVERY FROM OLTP PARALLEL-RELATIONS

JAMEL FEKI
YASSER HACHAICHI

Laboratory MIRACL, Faculté des Sciences Economiques et de Gestion de Sfax
Route de l'Aéroport km 4, B.P. 1088 Sfax, Tunisia
Jamel.Feki@fsegs.rnu.tn
Yasserhfr@yahoo.fr

ABSTRACT

Meaningful OLAP analyses often need to examine correlated data, i.e., data coming from multiple facts. Constellation schemes provide for this kind of need. In order to help the decisional designer construct efficient data marts, we propose a method that discovers and constructs constellation schemes directly from the relational database of the enterprise information system. For this, we define the concept of parallel-relations on which we base this construction and, a set of construction rules. Parallel-relations materialize interrelated business activities and therefore are good candidate for constellation construction. The construction rules extract multidimensional concepts; they build facts and measures on parallel-relations and, dimensions and their attributes on relations connected to parallel-relations. Our rules have the merit to produce dimensional attributes organized in hierarchies and, they keep track of the origin of each multidimensional component in the generated data mart schema. This trace is fundamental for the ETL processes.

Keywords: *Decisional support system, Parallel-relations, Fact constellation construction, relational data source.*

1. INTRODUCTION AND MOTIVATIONS

Decisional support systems (DSS) gather operational business information for decision support functionalities and data analyses. They still motivate several research issues such as design methodologies [16] [11] [12] [19], [20], complex data storage and manipulation [4], evolution [1], etc. They organize data in data warehouses (DW) and data marts (DM) [21]. A DM is modeled in a multidimensional way to facilitate the manipulation of data to decision-makers [18]. The commonly used multidimensional data models for DM are star [16] and constellation schemes [17]. In our previous works, we addressed the problem of constructing DM schemes directly from a relational database [10]. We proposed a set of rules [13] that allow the construction of star schemes. We demonstrated experimentally that this set of heuristics generates stars for all plausible facts over the given relational data sources. However, the generated schemes suffer from two lacks: i) all are limited to stars (no constellations are envisaged), and ii) sometimes the obtained stars contain common elements; in particular, some measures may appear in different facts having common dimensions. Moreover, we consider that the star schema is insufficient because it models one single fact at a time. However, in many situations, decisional users look for combined analyses, more sophisticated than those offered by a star schema.

Considering these aspects, the constellation schema, which is a generalization of the star, is preferable because it gathers two or several facts sharing common dimensions. That is, constellation schema allows cross-facts analyses that examine one fact with respect to others. Hence, constellations intensify analyses giving decision-makers the opportunity to better explore/understand their business performances. For instance, given a constellation composed of two facts, called *Sale* and *Purchase,* having the common dimensions Date, Product, Retail then the decision-maker who detects a free fall of sales in a given retail outlet can probably have the idea to throw a glance on the purchases (provisioning of stock) of that retail outlet during the same period. This significant decrease in sales may be due to a decrease in stock provisioning! To perform such an analysis, we need to build data marts with constellated facts.

We can design constellations in several ways: (i) on explicit request of decision-makers, or (ii) by examining/merging the decisional system star schemes,

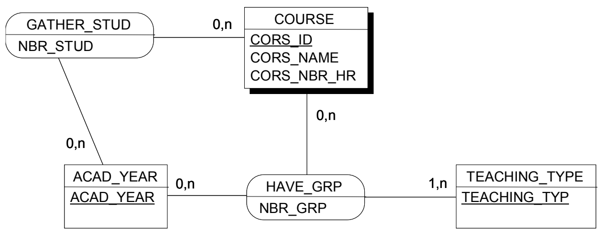
or, yet again, (iii) by identifying constellations automatically during the DSS design process.
To build constellations on explicit requirements of decisional users is not the best way because this assumes that users are skilled enough to forecast and clearly specify their needs, relatively complex, at the early stage of the decisional system design process. In practice, this is not easy since; in general, decision-makers are not only inexperienced for this task, but they are also unqualified for it. In addition, even if decision-makers can specify their needs in advance, the constellation built on the specified requirements may not be loadable from the target operational system. This problem is specific to top-down design methods.

A second way to build constellations can be by examining star schemes issued from the DSS design process, and then by merging those schemes with common dimensions. Unfortunately, this method raises several semantic problems: homonyms, synonyms, etc. For instance, two dimensions common to two different stars may have the same name but different meanings. Such semantic problems may also occur with fact names and attributes of dimensions
Consequently, we consider these two methods not very interesting because of their disadvantages listed above. Therefore, we seek a simple and effective one; the remainder of this paper addresses the following method.
A third way to build constellations is from scratch, i.e., directly on the OLTP (On Line Transaction Processing) system by identifying which objects (e.g., entities, relationships) of the data source model (e.g., E/R diagram, relational schema) could derive correlated facts. Among the goals of this work, we show that constellation can be automatically built by applying a set of identification/construction rules on the data model describing the operational information system.
This paper proposes our method to build constellations over a relational data source. It is organized in four sections. Section 2 introduces our basic idea to construct constellation based on the definition of parallel-relations. Section 3 details our proposed method of constellation construction; it defines a set of rules to build constellations, dimensions and their hierarchies. Section 4 summarizes the paper and outlines some perspectives.

2. CONSTELLATION CONSTRUCT BASICS

As mentioned in the introduction, we want to identify and build constellation schemes by examining the database data model of the operational information system. We base this work on the two following observations: *(i)* In data warehouse design approaches it has been unanimously accepted that a business activity (e.g. Sales Bill) is generally modeled as an association linking several entities; and *(ii)* the connected entities to this association (e.g. Items, Client…) represent the context of that activity. These two observations incite to limit the construction of:
– facts mainly on n-ary relationships [15], [12], [5], [23] and rarely on entities [19], [3], [20], [8], and
– dimensions on entities [12], [5], [19], [23] and sometimes on temporal attributes.
For instance, in this context, Moody [19] bases the DM construction on a classification of the concepts of an E/R diagram into transaction entities, component entities and classification entities. This classification helps the designer in DW construction.

On the other hand, we can find in operational systems two relationships that have a common subset of their linked entities; these relationships are said parallel. Figure 1-a shows that relationship GATHER_STUD is parallel to relationship HAVE_GRP because they share two entities: ACAD_YEAR and COURSE.
Parallel relationships represent two activities of the business and, they share a common sub-context; so they are partially dependent. This inspires us to build constellations on the basis of parallel relationships. We borrow this concept of parallel-relationships from the E/R model [22]. In an E/R diagram, a relationship *R1* related to *m* entities is said to be parallel to a relationship *R2* related to *n* (*m≤n*) entities if all entities connected to *R1* are also connected to *R2*. Because we assume that our data source is a relational database, we adapt parallel relationships to the context of the relational model.



(a)

GATHER_STUD (<u>Acad_Year</u>, <u>Cors_Id</u> # : **Course**, Nbr_Stud)
HAVE_GRP (<u>Acad_Year</u>, <u>Cors_Idt</u># : **Course**, <u>Teaching_Typ</u>, Nbr_Grp)
COURSE (<u>Cors_Id</u>, Cors_Name, Cors_Nbr_Hr,…)

(b)

FIGURE 1: EXAMPLE OF PARALLEL RELATIONSHIPS (a) AND THEIR RELATIONAL SCHEMA (b)

Thus, in case of relation *R1* parallel to relation *R2* and according to the above considerations, we propose to build a two-fact constellation schema whose facts are derived from *R1* and *R2*, and the set of shared dimensions is derived from entities simultaneously connected to *R1* and *R2*.

As this work extends our running approach of data mart schema construction [10], it continues to extract dimensional concepts (facts, measures, dimensions and attributes) directly from the relational database of the operational system. We have argued in [10] the motivations behind our orientation of working on relational databases; the main benefits are the availability, in the DBMS, of the most recent version of the operational system data model on the one hand (Mazón recently confirms this in [18]) and, the automation of the design process on the other hand.

As we assume that our data source is a relational database, we have to adapt parallel relationships to the context of the relational model. The matter is that the relational model [7] uses a single concept to model both entities and relationships; therefore we lost, in a relational database, the conceptual trace to distinguish which relation represents an entity and which relation is a relationship. So, how to identify parallel relationships in a relational database? To answer this question, we perform a reverse engineering task by accurately examining the structure of relations; first to determine the conceptual class of a relation, i.e. decide whether a relation implements an entity or a relationship and, secondly, to look for parallel relationships within the subset of relations classified as relationships.

2.1. IDENTIFICATION OF THE CONCEPTUAL CLASS OF A RELATION

Let *S* be a relational database schema for a given operational information system (IS), we split relations of *S* into two subsets:
– *Sr*: the subset of relations from *S* modeling relationships; we suggest designate them *relation-relationships* (*R-r* for short). Generally, each *R-r* is known by its primary key composed of one or several foreign keys.
– *Se*: the subset of relations from *S* modeling entities; we call them *relation-entity* (*R-e* for short). Generally, each *R-e* is known by its primary key not containing a foreign key.

We have pointed out in [10] that this classification should well form the two subsets *Sr* and *Se*; to do so, it should satisfy the following three properties:

i) Disjoint: $Sr \cap Se = \varnothing$, *ii) Completeness:* $Sr \cup Se = S$, *and iii) Correctness:* $\forall sr \in Sr$, *sr* is not an entity and, $\forall se \in Se$, *se* is not a relationship.

The first two properties are trivial; however, the correctness property is not satisfied in the two following situations: i) when the primary key of a relationship is not the concatenation of all its foreign keys: this primary key can be an artificial attribute such as a sequential number; or ii) when the primary key of a relationship is the concatenation of attributes coming from empty entities: such attributes are never foreign keys since an empty entity never transforms into a relation. The reader, if interested in empty entities, is referred to [13] where a detailed example is given.

2.2. A SAMPLE RELATIONAL DATA SOURCE

In order to illustrate our method of constellation construction, we consider the relational database of figure 2; it models university professors' loads (teaching courses and supervising students' projects). In this schema, primary keys are underlined and, foreign keys are followed by the sharp sign (#) and the name of referenced relation.

3. CONSTELLATION CONSTRUCTION METHOD

Our constellation construction method starts by identifying the conceptual class of each relation in the source database. Afterward, to design one constellation based on two relations *R1* parallel to *R2* with *m* relations simultaneously referred to by *R1* and *R2*, we create two facts conventionally called *F-R1* and *F-R2* (built on *R1* and *R2* respectively). To complete these facts we extract their measures from parallel-relations themselves. After that, we determine the common dimensions issued from the *m* relations and then, construct hierarchies of dimensions. In a next step, we look for specific dimensions and their attributes. Following the foreign key links, we obtain dimensional attributes organized in hierarchies. Finally, the result is a two-fact constellation.

Applying our method to the sample relational database of figure 2, we obtain the two subsets: *Se* and *Sr* depicted in figure 3-(a) and figure 3-(b) respectively.

Note that by analogy with parallel-relationship in an E/R diagram, we introduce in the relational context the term *parallel-relation* to denote a relation representing a parallel-relationship; therefore, a parallel-relation is necessarily of class *Sr*.

CURRICULUM (Cur_Id, Cur_Name,…)
AUDIENCE (Aud_Id, Aud_Name, Level_of_Study, Cur_Id# : **Curriculum**)
STUDENT (Stud_Id, Stud_Fname, Stud_Lname,…)
COURSE (Cors_Id, Aud_Id # : **Audience**, Cors_Name, Cors_Nbr_Hr,…)
PROFESSOR (Prof_Id, Prof_Fname, Prof_Lname, Telphone, Mobile, E_Mail, Prof_Typ, Prof_ Grad)
ENROLLED_IN (Aud_Id # : **Audience**, Stud_Id# : **Student**, Acad_Year)
CAN_TEACH (Prof_Id # : **Professor**, Teaching_Typ, Cors_Id# : **Course**)
SUPERVISE (Prof_Id # : **Professor**, GroupNo, Acad_Year)
SUB_GROUP (Stud_Id # : **Student**, Acad_Year, GroupNo, SubGroupNo)
HAVE_GRP (Acad_Year, Cors_Id # : **Course**, Teaching_Typ, Nbr_Grp)
CONTAIN_HOUR (Cors_Id # : **Course**, Teaching_Typ, Nb_Hr)
GATHER_STUD (An_Univ, Cors_Id # : **Course**, Nbr_Stud)
PROF_REALIZED _TEACHING (Cors_Id # : **Course**, Prof_Id # : **Professor**, Week_No, Teaching_Typ, Acad_Year, Nbr_Grp_Taught)
PROF _REQUIRED_LOAD (Teaching_Typ, Prof_Qualif, Hr_Load_Req)
PROF_REALIZED_SUPERVISING (Cur_Id # : **Curriculum**, Prof_Id # : **Professor**, SemesterNo, Acad_Year, Superv_Real)
SCHEDULE (Sch_Date, SemesterNo, Cors_Id # : **Course**)

FIGURE 2: *S* RELATIONAL DATABASE SCHEMA OF UNIVERSITY PROFESSORS' LOADS

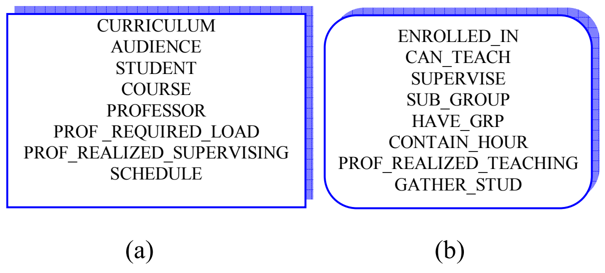


FIGURE 3: RELATIONS FROM SOURCE *S* CLASSIFIED INTO ENTITIES (a) AND RELATIONSHIPS (b)

Moreover, in order to accurately formalize our extraction rules for multidimensional concepts, the remainder of this paper uses the notations below:

– S: a relational database schema checked for third normal form,
– R: a relation from S,
– Ω_R: the set of all attributes of R,
– $\Omega_{R/NUM:}$ the subset of numeric attributes from Ω_R,
– $\Omega_{R/BOL:}$ the subset of Boolean attributes from Ω_R,
– $\Omega_{R/TEM:}$ the subset of temporal (date or time) attributes from Ω_R,
– Pk_R: the set of attributes constituting the primary of R ($Pk_R \subseteq \Omega_R$) and,
– Fk_R: the set of attributes constituting all foreign keys of R ($Fk_R \subseteq \Omega_R$); Fk_R may be empty.

3.1. IDENTIFICATION OF PARALLEL RELATIONS

Remember that parallel-relations correspond to parallel relationships in an E/R diagram. Therefore, to optimize their retrieval, we search them among relations of class *Sr* by using the following rule.

Rule 1. Parallel relation: Given *R1* and *R2* two relations of the same class *Sr* such as *R1* and *R2* connect *m* and *n* (*m*≤ *n*) relations respectively, *R1* is said to be parallel to *R2* (noted R1//R2) if and only if the primary key of *R1* is included in or equal to the primary key of *R2*.

Example: In figure 1-b, the two relations GATHER_STUD and HAVE_GRP model two relationships since they belong to *Sr* (cf. figure 3-b); also, and in accordance with our *Rule 1*, the relation GATHER_STUD is parallel to the relation HAVE_GRP since the primary key of the first one is included in the primary key of the second.

Formalization: According to our notation above, we formalize rule 1 as follows:

$$Given\ \ R1 \in Sr \wedge R2 \in Sr \wedge Pk_{R1} \subseteq Pk_{R2} \Rightarrow R1 // R2$$

For our running example of figure 2, this rule identifies within the set *Sr* (figure 3-b) all pairs of parallel relations. Figure 4 depicts the graph of these parallel-relations.

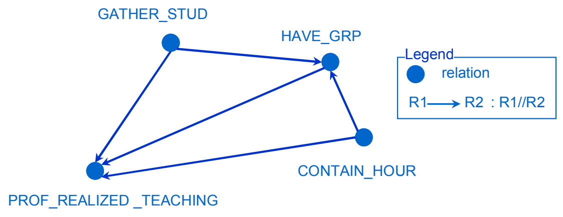


FIGURE 4: GRAPH OF PARALLEL RELATIONS RETRIEVED FROM SOURCE *S*

3.2. IDENTIFICATION OF CONSTELLATION FACTS

As shown in figure 4, parallel-relations probably form a graph with n ($n \geq 2$) nodes. This raises a matter; which nodes (i.e., parallel-relations) are most interesting for constellation construction. We estimate that defining a metric on the number of common dimensions may help the designer to decide which parallel-relations are most significant for their business. In the remainder, we focus on how to build constellation on a single pair of parallel-relations. For this, we define the rule below.

Rule 2. Constellation facts: Given *R1* parallel to *R2*, each containing a non key, numeric attribute, we build a constellation schema composed of two facts conventionally named *F-R1* and *F-R2*.
Note that the condition "*non key numeric attribute"* improves rule 2 in that it excludes facts without measures; considered as infrequent facts.

Formalization: According to our notation, every pair *R1*, *R2* of relations verifying the following conditions form a two-fact constellation noted *Const(F-R1, F-R2)*:

$$R1 \,//\, R2 \wedge \Omega_{R1/NUM} - \left(Pk_{R1} \cup Fk_{R1}\right) \neq \varphi \wedge$$
$$\Omega_{R2/NUM} - \left(Pk_{R2} \cup Fk_{R2}\right) \neq \varphi \Rightarrow$$
$$Const\left(F\text{-}R1, F\text{-}R2\right)$$

For our running example (cf. figure 4), we can build up to five two-fact constellations; naturally, not all of them have the same importance for OLAP analyses.

3.3. IDENTIFICATION OF MEASURES OF CONSTELLATION

A fact contains a finite set of measures, generally numerical [9]. These measures are extracted from the fact-relation (*i.e.*, relation on which the fact is built). The following rule extracts measures for a fact.

Rule Hm1: Constellation measures. *Non key numerical attributes* belonging to a relation R building a fact *F-R* and, *not belonging to other relations* are candidate measures for *F-R*.

This rule excludes key-attributes from the set of candidate measures because keys are generally artificial and redundant data; moreover, keys do not trace/record the enterprise business activity. Also, we exclude from *R* its *'non key attributes belonging to other relations*' because these attributes really represent keys issued from *empty entities*.

Formalization: According to our notation, the set of candidate measures for a given fact *F-R,* built on relation *R*, is defined by:

$$\Omega_{R/NUM} - \left(Pk_R \cup Fk_R \cup \bigcup_{\substack{Rj \in S \wedge \\ Rj \neq R}} \Omega_{Rj} \right)$$

Figure 5 shows measures extracted using our rule Hm1 for all facts of figure 4. The first two rows are measures for the facts of the constellation under construction *Const(F-GATHER_STUD, F-HAVE_GRP).*

Fact name	Extracted measure
F-GATHER_STUD	NBR_STUD
F-HAVE_GRP	NBR_GRP
F-CONTAIN_HOUR	NB_HR
F-PROF_REALIZED_SUPERVISING	NBR_GRP_TAUGHT

FIGURE 5: CANDIDATE MEASURES FOR THE FACTS OF FIGURE 4

3.4. IDENTIFICATION OF CONSTELLATION DIMENSIONS

In general, a dimension can be established on entity directly linked to a relationship. Also, we can build dimensions on attributes or even on empty entities. We have precisely studied these issues in [13] for star schema dimension construction. The rules we have established remain applicable for constellation dimension construction.

a) DIMENSION BUILT ON AN ENTITY

For a constellation *Const(F-R1, F-R2)*, we find a non empty set of dimensions common to facts *F-R1* and *F-R2* and then, for each of these facts we determine a set of specific dimensions. The first set is mainly built on the common relations connected simultaneously to *R1* and *R2*; whereas the second set is built on relations linked to *R2* but not to *R*1. The following rule finds dimensions for a constellation schema. It is useful to determine the set of dimensions common to both facts and also, to determine the dimensions specific for the fact established on relation *R2*.

Rule Hd1: Every relation-entity *R* directly referred to by a fact-relation *F* is a candidate dimension for *F*. Conventionally, the name of this dimension is that of *R* and its identifier is the primary key of *R*.

Example: According to rule Hd1 relation COURSE is directly referred to by the fact-relation HAVE_GRP; therefore, it is a dimension for the fact F-HAVE_GRP with CORS_ID as its identifier.

Formalization:
$Hd1:\{R \in Se : Pk_R \cap Fk_{R1} \neq \phi\}$; where Pk_R is the identifier of dimension built on *R*.

In the next sections, we call *dimension-relation* a relation identified as a dimension.

b) DIMENSION BUILT ON AN ATTRIBUTE

In addition to dimensions built on relations, we can define a dimension on an attribute of a special data type (Boolean, temporal) or, even on an attribute issued from an empty entity. We first give the corresponding rules and then, we apply our rules to the relational data source of our example. Naturally, a Boolean column splits data-rows of its table into two subsets; thus such an attribute can be an axis of analysis. As an example, a *Gender* column in a *Student* database table can build a dimension.

Rule Hd2: A Boolean attribute *b* belonging to a fact-relation *F* generates a candidate dimension for *F.* The identifier of that dimension is *b.*

On the other hand, the data warehouse community assumed a data warehouse as a chronological collection of data [17]. Consequently, the *Date* dimension appears in all data warehouses. For this reason, we propose to build dimensions on temporal attributes. As an example, the *Booking date* attribute in an aircraft reservation database may interest decision-makers as an axis of analyses for the *Booking* activity of the company [2]. Considering this, we define the following rule.

Rule Hd3: A temporal attribute *(date or time)* belonging to a fact-relation *F* timestamps the fact built on *F;* it generates a candidate dimension for *F* where it is the identifier.
In addition, the following two rules establish dimensions on the basis of attributes denoting empty entities.

Rule Hd4: Given *F* a fact-relation of class *R-r*; if the primary key of *F* contains a non foreign key attribute *a,* then attribute *a* generates one dimension identified by *a.*

Example: The attribute TEACHING_TYP that represents the empty entity TEACHING_TYPE (cf. figure 1) is identified as a dimension for fact F-HAVE_GRP.
Rule Hd5: Every non key (primary of foreign) attribute *a* belonging to a fact-relation *F* and belonging to other relations becomes a dimension for fact *F.* The dimension identifier is *a.*

Formalization: Given *R1* a fact-relation belonging to a data source *S,* the set of all its candidate dimensions is the union of the following sets obtained by the above rules.
Note that when we build a dimension on attribute *a* (e.g. *Gender*) we assign the name *D_a* to that dimension (e.g. *D_Gender*). The identifier of dimension *D_a* is attribute *a* (i.e., *Gender*).

$Hd1:\{R \in Se : Pk_R \cap Fk_{R1} \neq \phi\}$; where Pk_R, *is the* identifier of dimension built on *R*.

$$Hd2: \bigcup_{a \in \Omega_{R1/BOL}} \{D\_a\}; \qquad Hd3: \bigcup_{a \in \Omega_{R1/TEM}} \{D\_a\};$$

$$Hd4: \bigcup_{a \in (Pk_{R1} - Fk_{R1})} \{D\_a\};$$

$$Hd5: \bigcup_{a \in \Omega_{R1} - (Pk_{R1} \cup Fk_{R1})} \{D\_a : \exists R \in S - \{R1\} \wedge \Omega_{R1} - (Pk_{R1} \cup Fk_{R1}) \subseteq R\};$$

Example: For the relational database of our example, these rules produce dimensions shown in figure 6.

3.5. IDENTIFICATION OF HIERARCHES

In multidimensional modelling, the attributes of a dimension are ordered from the lowest to the highest granularity to form candidate hierarchies [21]; such ordered attributes are said parameters. In addition, any hierarchy of a dimension *d* has the identifier of *d* as its finest parameter (*i.e.,* of rank 1) already extracted with *d.* This identifier can functionally determine some attributes within its correspondent relation; these attributes describe the identifier and, therefore, can play the role of descriptive (or non dimensional) attributes. Among these, the textual attributes are more significant than the numerical ones. Due to space limitation, the rules for hierarchy construction are not detailed here (cf. [10]); we limit ourselves to give the result of their application on our example.
Figure 7 depicts two constellations represented according to DFM [12] notation; they are built on two pairs of parallel-relations issued from the relational database of figure 2. For instance, the hierarchy of the dimension COURSE has three parameters (shadowed

circles) and four descriptive attributes. All of them are extracted from relations COURSE, AUDIENCE and CURRICULUM by exploring the foreign key constraints.
Finally, note that all the elements of an obtained multidimensional schema are extracted and carried forward with their corresponding attachment to the data source ; this allows the designer to rename these elements more significantly without losing the origin in the source. This deals with implementation, an immediate step for this work.

Fact name	Extracted Dimension	Dimension Identifier	Extraction Rule
F-HAVE_GRP	COURSE	Cors_Id	Hd1
	D_ACAD_YEAR	Acad_Year	Hd4
	D_TEACHING_TYP	Teaching_Typ	
F- CONTAIN_HOUR	COURSE	Cors_Id	Hd1
	D_TEACHING_TYP	Teaching_Typ	Hd4
F- GATHER_STUD	COURSE	Cors_Id	Hd1
	D_ACAD_YEAR	Acad_Year	Hd4
F- PROF_REALIZED _TEACHING	COURSE	Cors_Id	Hd1
	PROFESSOR	Prof_Id	
	D_SEMESTERNO	SemesterNo	Hd4

FIGURE 6: CANDIDATE DIMENSIONS FOR THE FACTS ISSUED FROM SOURCE *S*

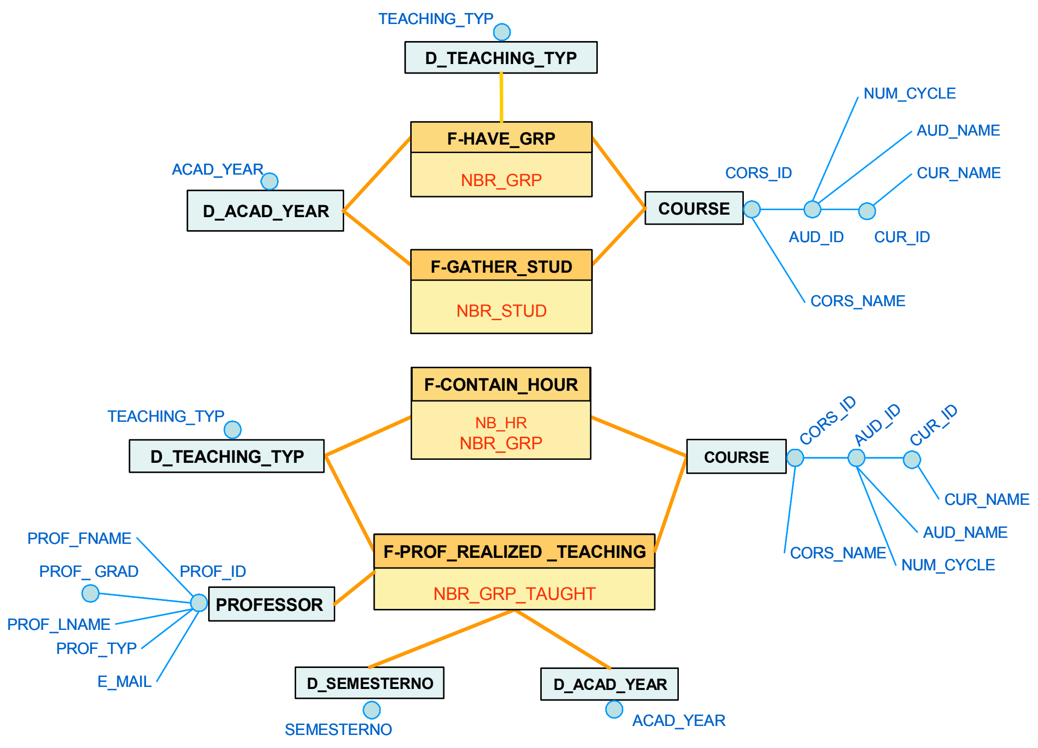


FIGURE 7: TWO CONSTELLATIONS BUILT ON TWO PAIRS OF PARALLEL RELATIONS

4. SUMMARY AND FUTURE EXTENSIONS

The context of this work is the design of data mart schemes. Our design method aims to assist the designer to define the structure of the data mart on the basis of the enterprise operational system. This system is modeled as a relational database. More precisely, our contribution addressed the problem of construction of constellation schemes. For this, we defined the concept of *parallel relationships* as relationships describing two dependent activities of the business and, sharing a common context. Also, we have argued that parallel relationships are significant to build consistent constellation schemes useful to enhance transversal, decisional analyses. These analyses allow understanding dependencies between the enterprise activities and, therefore, better explain business results (e.g., sales). For our constellation construction method we defined a set of rules and, we illustrated their use through a concrete example. Our rules extract facts and their measures; also, they determine dimensions and their attributes organized into hierarchies. So far, our method has built constellations limited to two facts; however, in practice, we may need multiple fact constellations; therefore we need to decide what facts are to be constellated? What is the 'best' combination? … For the short term, we plan to extend this work by defining a metric that helps the decisional designer during the constellation process. In addition, since constraints in multidimensional systems began to intensively interest the DW community, we envision detecting implicitly some constraints at the multidimensional schema construction phase; it is to enhance the quality of data to be loaded in the DM.

REFERENCES

[1] Blaschka M., "FIESTA: A Framework for Schema Evolution in Multidimensional Databases", Ph. D. thesis, Institut für Informatik der Technischen Universität München, December 2000.

[2] Böhnlein M., Ulbrich-vom Ende A., "Deriving Initial Data Warehouse Structures from the Conceptual Data Models of the Underlying Operational Information Systems", 1999.

[3] Bonifati A., Cattaneo F., Ceri S., Fuggetta A. and Paraboschi S., "Designing Data Marts for Data Warehouse", ACM Transaction on Software Engineering and Methodology, vol. 10, pp. 452-483, Octobre 2001.

[4] Boussaid, O., "Evolution de l'entreposage des données complexes", Memoire de HDR, université lumière Lyon2, 2006.

[5] Cabibbo, L., and Torlone, R., "A Logical Approach to Multidimensional Databases", Conference on Extended Database Technology, Valencia-Spain, pp. 187-197, 1998.

[6] Calvanese, D., Dragone, L., Nardi, D., Rosati R. and Trisolini, S., "Enterprise modeling and Data Warehousing in Telecom Italia", Information. Systems. 31(1), 2006.

[7] Codd, E.F. "A Relational Model of Data for Large Data Banks", ACM Communications, Vol. 13, No 6, pp 377-387, 1970.

[8] Ghozzi, F., "Conception et manipulation des bases de données dimensionnelles à contraintes", Thèse de Doctorat, Univ. Paul Sabatier, France, 2004.

[9] Feki, J. and Ben-Abdallah, H., "Multidimensional Pattern Construction and Logical Reuse for the Design of Data Marts", International Review on Computers and Software (IRECOS), Mars 2007.

[10] Feki, J. and Hachaichi Y., "Du relationnel au multidimensionnel : Conception de magasins de données", Revue RNTI vol B-3, p 5-19, 2007.

[11] Golfarelli M., Maio D., and Rizzi, S., "Conceptual Design of Data Warehouses from E/R Schemas", Conference on System Sciences, Kona-Hawaii, 1998, Vol. VII.

[12] Golfarelli, M., Maio, D. and Rizzi, S., "The dimensional fact model: a conceptual model for data warehouses", Int. Journal of Cooperative Information Systems, pp.215-247, 1998.

[13] Hachaichi, Y. and Feki J., "Heuristiques de construction de MD à partir d'une source OLTP relationnelle", Atelier des Systèmes Décisionnels (ASD'06), Agadir-Maroc, 2006.

[14] Hüsemann, B., Lechtenbörger, J. and Vossen, G., "Conceptual Data Warehouse Design", Proc. of the Int'l Workshop on Design and Management of Data Warehouses, Stockholm-Sweden, pp. 6.1-6.11, 2000.

[15] [Kimball, 1997] Kimball R., "A Dimensional Modeling Manifesto", DBMS Magazine, Juillet, 1997.

[16] Kimball R., "The Data Warehouse Toolkit", John Wiley and Sons Inc, 1997.

[17] Kimball R., Revues L., Ross M., Thornthwaite W. "Le data warehouse: Guide de conduite de projet", Eyrolles, 2005.

[18] Mazón J.-N. and Trujillo J., "An MDA approach for the development of data warehouses", Decisional Support System., 2007.

[19] Moody L.D., and Kortink M.A.R., "From Enterprise Models to Dimensional Models: A Methodology for Data Warehouses and Data Mart Design", Proc. of the Int'l Workshop on Design and Management of Data Warehouses, Stockholm-Sweden, 2000.

[20] Phipps C. and Davis K., "Automating data warehouse conceptual schema design and evaluation", DMDW'02, Canada, 2002.

[21] Ravat, F., Teste, O., Zurfluh, G., "Modélisation multidimensionnelle des systèmes décisionnels", Revue ECA, vol. 1, n° 1-2, 1999, p.201-212.

[22] Seba D., Merise - Concepts et mise en œuvre, Eni,edition 2003.

[23] Soussi A., Feki J., Gargouri F., "Approche semi-automatisée de conception de schémas multidimensionnels valides", Revue RNTI vol B-1, p71-90, 2005.