An Unnormalized Relational Data Model Based On User Views

G. Pernul
Department of Statistics & Computer Science
University of Vienna, Austria

0. Abstract

A description of a database design method according to user views is introduced. The database schema is decomposed by applying the operators: horizontal decomposition, vertical decomposition and derived horizontal decomposition. Resulting fragments (= effects of the decomposition) are controlled by a central Data Dictionary, consisting of two subschemata, the Application Subschema and the Data Subschema.

1. Introduction

In the past few years there has been a rapid increase in the use of database applications. The early hierarchical model and the more general network model have turned out widely applicable for computing areas with a large amount of data. These two models were exceeded, however, during the last ten years by Codd's relational data model, which was introduced already in 1970. Database practitioners and researchers have taken widespread interest in this model because of its conceptual simplicity, the meanwhile exhaustively investigated theoretical background and the well known mathematical theory of sets and relations. The heart of the relational model is the concept of normal forms. In Codd's traditional relational database approach, relations have to be in first normal form, i. e. they must have atomic attributes.

However, the use of structured records in the presentation of scientific and technical applications, in integrated office information systems and in geographical databases (in so called "non standard applications) can no longer be neglected. The constraint of first normal form does not apply to some new extended relational data models, which have been investigated by various database researchers in the last years [Kam83], [Jae84], [Fis83], [Abi84]. In [Sch86] a precise formal description of a so called NF^2 (Non First Normal Form) relational data structure and of a related NF^2 relational algebra was given. However, none of the new models (inc. NF^2 model) takes account of the different views of the users. The model proposed in this paper takes into account the different views of the universe of discourse of the different users.

2. Decomposition Operators

The universe of discourse is represented by an initial start schema which consists of non necessarily normalized relations (representations of the objects of the real world) and attributes of the relations (the characteristics of the objects). There may exist some dependencies in the form of functional (FDs) or multivalued dependencies (MVDs) between these attributes. In order to ensure data integrity every relation has a unique identifier, called surrogate [MeiLo], thereby ensuring that duplicate tuples do not occur within one relation. Other attributes may be used as one or more foreign keys, if values of the foreign key are matched by equal values within another table's surrogate. By using foreign keys it is possible to construct a hierarchy of relations, and to represent easily a hierarchical universe of discourse, i. e. we have a mapping from a hierarchical real world to a "hierarchical" relational data model with all the properties of a relational model. In most cases a large number of users share an integrated database. In such a multiuser environment different users will have different views of the database. These views may overlap or not. A prodecure will be shown, which constructs a decomposition of a non normalized start schema according to user views by horizontal or vertical cutting.

2. 1. Horizontal Fragmentation

A horizontal fragmentation (//) is the subdivision of the tuples of a relation R into disjoint subsets called fragments. Each horizontal fragment has the same attributes as the original relation. A way of defining a horizontal fragmentation is through a set X_j of fragmentation predicates defined on an attribute of R.

$$X_j = \{x_1, \ldots, x_n\}$$

Let S be the set of attributes of R, $A_j \in S$ be an attribute of S with domain Dom_j; then a predicate is expressed by

$$x_i: \text{val}(A_j) \in d_{ji} \text{ or } \text{val}(A_j) = \text{val}(A_k)$$

with $\text{val}(A_j)$ as the value of the attribute A_j to which x_i is applied and $d_{ji} \subseteq Dom_j$ or $\text{val}(A_k)$ respectively in the case of a comparison between two attributes. A_j is called selection attribute and X_j is the set of all predicates defined on A_j. Therefore a horizontal fragmentation according to only one selection attribute is defined by the

following expression:

attribute-name θ attribute-name or
attribute-name θ constant,

where θ is one of the following comparison operators: = (equals),< (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to), <> (different from).

Example: $S = \{A_1, A_2, \ldots, A_n\}$
$X_1 = \{val\ (A_1) <= 10,$
$11 <= val\ (A_1) >= 20,$
$val\ (A_1) > 60$
$val\ (A_1) = val\ (A_7)\}$

The example shows a (//) of S according to the value of selection attribute A_i into 8 fragments possibly.

Horizontal fragmentation of an object type is not restricted to only one selection attribute, i. e. implications between predicates defined on different selection attributes are possible in the form of ∧ (conjunction) and ∨ (disjunction).

Let P be the set of predicates defined on all selection attributes of R then $P = \{X_1, \ldots, X_k\}$ splits R into a set of horizontal fragments. It is assumed that fragments are disjoint and therefore P defines disjoint groups of tuples of the relation. If there are overlapping fragments, it is always possible to build a finer fragmentation in which fragments are disjoint. A horizontal fragmentation is correct if each tuple of R is mapped exactly in one of the resulting fragments.

2.2 Vertical Fragmentation

A vertical fragmentation (*) is defined by projecting a relation R over subsets of its attributes. To make the projection lossless, a primary key attribute or a surrogate as a unique tuple idenfifier is included within each fragment. Vertical fragmentation is non-overlapping, i. e. every non-prime attribute belongs to one and only one fragment. For $R = (*)\ (F_1, \ldots, F_n)$ (where (*) is the vertical fragmentation operator, R is the start schema and F_i are the fragments obtained by vertical fragmentation) a vertical fragmentation is correct if for every tuple u in R, u is the natural join of $v_1, v_2, \ldots, v_n$ with $v_1, v_2, \ldots, v_n$ in $F_1, F_2, \ldots, F_n$, respectively (lossless join dependency).

2.3. Derived horizontal fragmentation

A derived horizontal fragmentation (d//) is the concept of partitioning a relation R by applying to it the same partitioning criterion as applied to another relation or user view. Owner and member relations are concatenated via a link (foreign key). With a given partitioning of an owner object and an 1:n association between owner object and member object it is always possible to define a corresponding derived horizontal fragmentation.

3. Structured Decomposition of an Integrated DB Schema

The start schema R consists of a set of relations $\{R_1, \ldots R_n\}$ and a set of user views $\{V_1, \ldots V_m\}$ defined on them. Every relation R_i is uniquely identified by a surrogate K_i.

Def. 3.1: A user view V_x consists of the whole or of a part of the attributes of the relations constituing the start schema as it is seen by a group of users. In this sense, a view can be considered as being the same as a sub-schema of R. V_x consists of a set of attributs ATTR (V_x).

Def. 3.2: Two different user views V_x and V_y are said to be overlapping if:

$$\exists\ (A_i) \mid (A_i \in V_x \wedge A_i \in V_y) \wedge (SD(V_x)[A_i] \cap (SD(V_y[A_i] \neq \phi$$

with $SD(V_x)[A_i]$ as the subdomain of attribute A_i in view V_x and $SD(V_y)[A_i]$ in view V_y respectively. $SD(V_j)[A_i] \subseteq DOM(A_i)$.

Def. 3.3: A user view V_x is called isolated user view, if for every user view V_y the following expression is valid:

$$\nexists\ (A_i) \mid (A_i \in V_x \wedge A_i \in V_y) \text{ or if}$$
$$\exists\ (A_i) \mid (A_i \in V_x \wedge A_i \in V_y) \text{ than}$$
$$(SD(V_x)[A_i] \cap (SD(V_y[A_i] = \phi$$

with $SD(V_x)[A_i]$ as the subdomain of attribute A_i in view V_x and $SD(V_y)[A_i]$ in view V_y respectively. $SD(V_j)[A_i] \subseteq DOM(A_i)$.

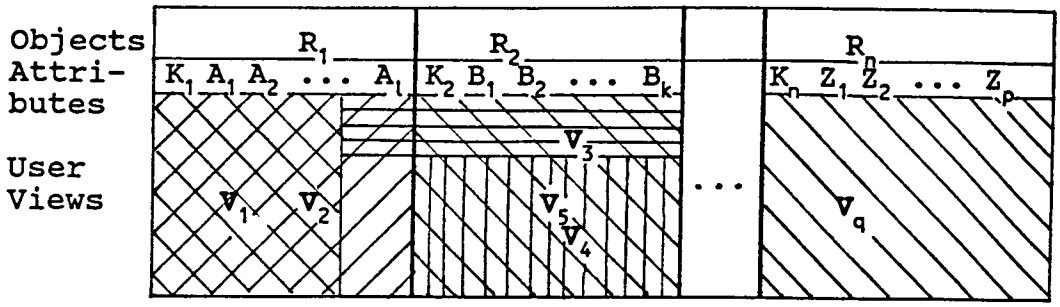

Figure 1: Start Schema R

Figure 1 shows an integrated database schema used as a start schema for a structured decomposition. The schema consists of n relations, each uniquely identified by a surrogate K_i and consisting of a set of attributes. This example illustrates the different user views in a multiuser environment. View V_1 consists of only a part of the attributes of R_1 (vertical fragmentation) while V_2 consists of all attributes of R_1. V_3 reflects a derived horizontal fragmentation consisting of all attributes of R_2 which are satisfying the selection criterion and of some attributes of R_1. For example, if R_1 represents courses held at the University of Vienna and R_2 represents students, which attend courses than a possible explanation could be: V_3 consists of the whole information about students, who attend the courses Compilers I and Database Systems and of the title and the location of the courses. V_4 is a horizontal fragmentation of R_2, it may reflect a view, consisting only of students born in Vienna. V_5 consists of all attributes of R_2. V_q consists of all attributes of R_n.

The construction of the decomposition is done by the following procedure:

1. Find all user views which are overlapping.

2. For each pair of overlapping user views perform a structured decomposition on them by applying (//), (d//), and (*) into the smallest non overlapping rectangular strips.

3. Perform a vertical, horizontal or derived horizontal fragmentation for each isolated user view.

The structure of the decomposition can be represented by a decomposition tree. Figure 2a shows the decomposition tree, where CF_i stands for the i-th candidate fragment, candidate for demonstrating that this fragmentation need not be the final fragmentation. Figure 2b shows the resulting fragmentation.

Def. 3.4: A candidate fragment CF is the result of the decomposition of two overlapping user views or an isolated user view by using the three decomposition operators

1. vertical fragmentation
2. horizontal fragmentation
3. derived horizontal fragmentation.

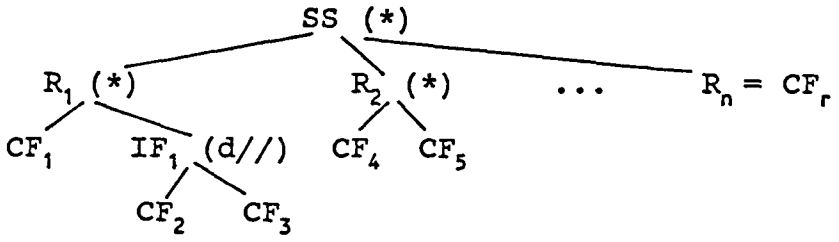

Figure 2a

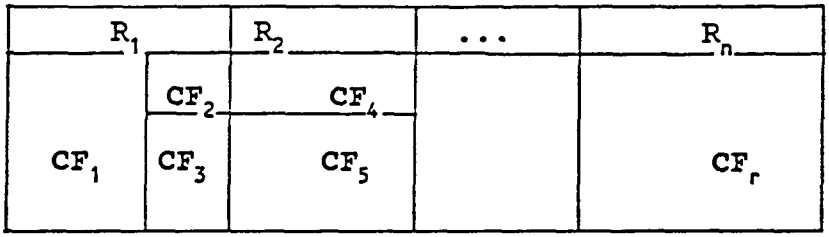

Figure 2b

SS : Start schema R
IF_i: i-th intermediate Fragment

The user view V_3 now consists of 2 candidate fragments CF_2 and CF_3. V_5 for example consists of CF_4 and CF_5.

5. The integration of FDs and MVDs into candidate fragments

A superficial consideration of the explanations given until now might suggest that the problem of partitioning a database schema according to user views has been treated exhaustively. However, the model proposed has to be extended to the effect that dependencies between attributes of different CFs must not be lost. In [Smi85] a method was shown in which dependencies between attributes are depicted as bubbles and double bubbles in a dependency diagram, from which a set of fully normalized tables is derived. The normalization theory does yield a database free from processing anomalies, but this theory is a very unending way to achieve a nonloss decomposition, i. e. to satisfy all 5 NF guidelines.

What are the consequences of horizontal or vertical fragmentation on dependencies? [deP82] showed, that in case of horizontal fragmentation there is no loss of any dependencies, on the contrary in the resulting fragments some new dependencies are valid which have not been valid in the start schema. But what about vertical fragmentation? A dependency within a CF has no effect on the decomposition. But let us consider dependencies within different CFs. Relation R_1 of figure 1 consists of l=8 attributes and is vertically fragmented in the candidate fragments CF_1, and IF_1 which is horizontally fragmented in CF_2 and CF_3 (see figure 3).

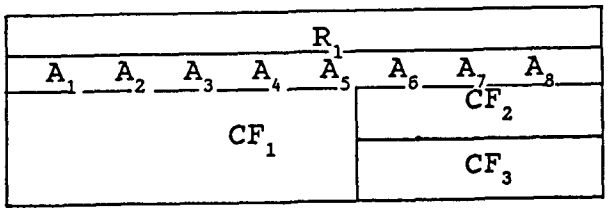

Figure 3: Relation R_1, consisting of 3 CFs

Let us assume that there exists a FD: $A_4 \rightarrow A_6$ and a MVD: $A_2 \twoheadrightarrow A_7$.

By applying the conventions of the dependency diagram given in [Smi85] a candidate fragment can be expressed as a bubble, consisting of all attributes of the candidate fragment and associated with a domain flag (Δ) indicating a horizontal fragmentation. A FD is expressed by a single-headed arrow while an MVD is expressed by a double-headed arrow. R_1 of figure 3 leads to the following dependency diagram in figure 4.

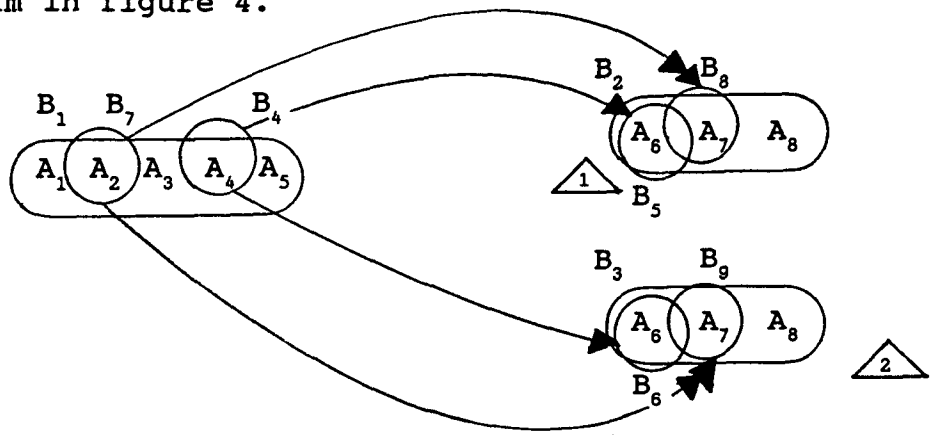

Figure 4: Dependency Diagram

B_1, B_2, B_3 are called candidate fragment bubbles. B_4 is called prime key bubble, because A_4 is the left side of a FD. B_5, B_6 are called target bubbles, because A_6 is the right side of a FD. B_7 is called uplink key bubble as a

left side of a MVD and B_8 and B_9 are called end key bubbles.

To reach a nonloss decomposition of a start schema according to user views, every CF must be expanded to prime key bubbles, uplink bubbles and end key bubbles. After this expansion CFs are called fragments to indicate that the decomposition procedure is finished. CF_1 of figure 3 is now called fragment F_1 and consists of the attributes $\{A_1, A_2, \ldots A_5, A_7\}$

$$F_2 = \{A_2, A_4, A_6, A_7, A_8\}$$
$$F_3 = \{A_2, A_4, A_6, A_7, A_8\}.$$

5. Data Dictionary

The structure of the decomposition is stored in the Data Dictionary (DD) which is also modelled by using the NF^2 relational data structure. The DD consists of two subschemata, the Application Subschema which contains the information about the decomposition structure and the Data Subschema which contains information about the user views and the access rights of the different users. Figure 5 shows the DD for the database schema of figure 1.

D A T A D I C

A p p l i c a t i o n S u b s c h e m a

Fragment	Parent	Operator	Brother & Sisters*
F_1	R_1	(*)	$\{IF_1\}$
F_2	IF_1	(d//)	$\{F_3\}$
F_3	IF_1	(d//)	$\{F_2\}$
F_4	R_2	(*)	$\{F_5\}$
F_5	R_2	(*)	$\{F_4\}$
...	...	...	...
$F_r=R_n$	SS	(*)	$\{R_1, R_2, \ldots R_{n-1}\}$
IF_1	R_1	(*)	$\{F_1\}$
R_1	SS	(*)	$\{R_2, R_3, \ldots R_n\}$
R_2	SS	(*)	$\{R_1, R_3, \ldots R_n\}$
SS	-	-	-

T I O N A R Y

D a t a S u b s c h e m a

Access Rights			Attributes	
Fragment	User	Kind	Name	Range
F_1	$\{V_1$	R/W	$\{A_1$	integer
	$V_2\}$	R/W	...	...
			A_5	char
			$A_7\}$	[10 .. 200]
F_2	$\{V_2$	R/W	$\{A_2$	char
	$V_3\}$	R/W	A_4	[$\geqslant$ 101]
			A_6	char
			A_7	[10 .. 200]
			$A_8\}$	integer
F_3	$\{V_2\}$	R/W	$\{A_2$	char
			A_4	[0 .. 100]
			A_6	char
			A_7	[10 .. 200]
			$A_8\}$	integer
...				

Figure 5: Data Dictionary

F_2 and F_3 consist of the same attributes but the powersets of the domains of their attributes are not the same, because they are the results of a derived horizontal fragmentation. The selection attribute is A_4. In F_2 only tuples are stored with a value in A_4 greater or equal to 101 while in F_3 there are only tuples with a value in A_4 between 0 and 100.

6. Conclusion

The data model proposed has several advantages in a multiuser environment. The main advantage is based upon considerations of efficiency in database retrieval and storage because it is possible to process user queries by creating concurrent processes to work on the subrelations (= fragments) of a start schema. The second reason for structured decomposition according to user views is that it can lead to a more secure database. For example, consider a patient database, consisting of a relation patient with attributes Name and Adress as non confidential data and

other attributes as confidential data. Vertical fragmentation can lead to two fragments, one containing non confidential data and the other containing confidential data. Thus this technique is an attractive way to ensure database security.

The data model proposed may be expanded in several directions. It has advantages in a multiprocessor environment, where the differnt fragments are distributed over several processors and one master processor holds the DD and acts as an interface to the user or host. Another investigation would be to extend and implement the procedures shown and to use the implementations as CADD (Computer Aided Database Design).

REFERENCES

[Abi84]: S. Abiteboul, N. Bidoit. Non First Normal Form Rela-tions to Represent Hierarchically Organised Data. Proc. 2nd ACM SIGACT/SIGMOD Symp. on Princ. of Database Systems, Waterloo, Ontario, Canada, 1984.

[deP82]: P. De Bra, J. Paredaens. Horizontal Decomposition and Their Impact on Query Solving. ACM SIGMOD Records, Vol. 13, Number 1, September 1982.

[Jae84]: G. Jaeschke. Recursive Algebra for Relations with Relation-valued Attributes. Technical Report Nr.84.001.003, IBM Heidelberg Scientific Centre, 1984.

[Fis83]: P. C. Fischer, S. J. Thomas. Operators for Non-First-Normal-Form Relations. Proc. IEEE Compsac, 1983.

Kam83]: Y. Kambayashi, K. Tanaka, K. Takeda. Synthesis of Unnormalized Relations Incorporating More Meaning. Int. Journal of Information Sciences (Special Issue on Databases), 1983.

[Meilo]: A. Meier, R. A. Lorie. A surrogate concept for engineering Databases.

[Sch86]: H.-J. Schek, M. H. Scholl. The Relational Model with Relation-Valued Attributes. Information Systems, Vol 11(2), 1986.

[Smi85]: Henry C. Smith. Database Design: Composing Fully Normalized Tables from a rigorous Dependency Diagram. CACM, Vol. 28, Number 8, August 1985.