The Concept of Data Model Pattern Based on Fully Communication Oriented Information Modeling (FCO-IM)

Fazat Nur Azizah[1], Benhard Sitohang[1], Guido P. Bakema[2] & Oerip S. Santoso[1]

[1]School of Electrical Engineering and Informatics, Bandung Institute of Technology
[2]Faculty of Engineering, Institute of Information Technology,
Media and Communication, HAN University of Applied Sciences, Netherlands

Abstract. Just as in many areas of software engineering, patterns have been used in data modeling to create high quality data models. We provide a concept of data model pattern based on Fully Communication Oriented Information Modeling (FCO-IM), a fact oriented data modeling method. A data model pattern is defined as the relation between context, problem, and solution. This definition is adopted from the concept of pattern by Christopher Alexander. We define the concept of Information Grammar for Pattern (IG_P) in the solution part of a pattern, which works as a template to create a data model. The IG_P also shows how a pattern can relate to other patterns. The data model pattern concept is then used to describe 15 data model patterns, organized into 4 categories. A case study on geographical location is provided to show the use of the concept in a real case.

Keywords: *context; data modeling; data model pattern; FCO-IM; Information Grammar for Pattern; problem; solution.*

1 Introduction

Patterns have been used in many areas of software engineering to help design processes in creating high quality solutions, including in data modeling. Data model patterns in particular are used to help the design of data models in order to have high quality data models by reusing proven solutions to particular data modeling problems. In this research, we focus on data model at conceptual level (see Simsion [1]). Thus, the data model patterns that we discuss in this paper are the conceptual ones.

The focus of current works on data model patterns are mainly on providing the so called domain-specific data model patterns, especially for the enterprise domain (see for example: Hay [2], Silverston [3]). There are not many discussions on the concept of the data model pattern itself. It is understandable because people are more interested in the patterns to use them in their works and not much in the concept. Nevertheless, describing a good concept on data

model pattern is important in order to give a strong base in defining the patterns themselves.

From modeling point of view, most current works on data model patterns use Entity Relationship Modeling (ERM) or Object Oriented Modeling (OOM) as the conceptual data modeling approach (see Coad, et al. [4], Fowler [5], Hay [2], Nicola, et al. [6], Silverston [3]). There is not yet many works on data model pattern that is based on fact oriented modeling (FOM) approach. FOM approach is aimed at modeling the structure of the communication about a universe of discourse (UoD), not modeling the UoD itself, which is basically the basic philosophy of ERM and OOM. A leading method in FOM is Fully Communication Oriented Information Modeling (FCO-IM) which holds the basic principles of FOM more consequently than other existing FOM methods (Bakema, et al. [7]). It is expected that the use of FOM approach, especially FCO-IM, in the discussions of data model pattern will give more insights and provide a more powerful concept for the data model pattern.

Some of our earlier works on data model pattern can be found in Azizah, et al. [8-10]. Azizah, et al. [8] is our preliminary work in which we introduced the concept data model pattern using FCO-IM, but not in further details. In Azizah, et al. [9], we presented the descriptions on several data modeling problems which become the basis of our generic data model patterns. In Azizah, et al. [10], we focus only on the concept of Information Grammar for Pattern which is a part of the solution of a data model pattern and how it forms a pattern language. This paper describes the whole concept of data model patterns based on FCO-IM and the concept of pattern by Alexander [11] (in which the concept of Information Grammar for Pattern is a part of). In this paper, we also provide examples on data model patterns and a case study of modeling using the data model patterns.

2 Foundations

2.1 Data Modeling

Data modeling is defined as a process of creating a data model by applying formal data model descriptions using data modeling techniques. Data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints (Silberschatz, et al. [12]). In general, there are 3 levels of data model that should be created during data modeling: conceptual, logical, and physical data model. Conceptual data model is a relatively technology-independent specification of data structures and is close to business requirements (Simsion [1]). ERM, OOM, and all FOM methods, including FCO-IM, are used to model data in conceptual level.

2.2 FCO-IM

FCO-IM is a fact oriented conceptual data modeling method which was created based on NIAM (Nijssen's Information Analysis Method). In FCO-IM, information analysis is carried out on *fact expressions*, i.e. sentences that express concrete facts within a Universe of Discourse (UoD). The final product of data modeling using FCO-IM is called an *Information Grammar (IG)*, which is considered as the conceptual data model. An IG stores the fact expressions in type level. These are called the *fact types*. Fact types are accompanied by data model *constraints* which are basically the rules that define valid fact expressions. Parts of a fact type are called *roles*. Roles of a fact type can be played by either an *object type* (a representation of real world object) or a *label type* (a representation of group of values). Object type is considered as a nominalized fact type. To help user to understand an IG better, an *Information Grammar Diagram (IGD)* is used. Further description of FCO-IM can be found in Bakema, et al. [7].

Consider the following examples of fact expressions:

```
The name of product PAP192 is Johnson paper.
 "   "   "     "    PEN202 "  Goldstein pen.
 "   "   "     "    DSK401 "  Jerry's disk.
```

An IG can be considered as an abstraction of concrete fact expressions. The abstraction is carried out by taking into account only the common parts of fact expressions to form a fact type. For example: from the fact expressions, we can create the following IG:

```
Name of Product
F2 : "The name of <Product : O2> is <product name>."
O2 : 'product <product code>'
UC3 : "Name of Product is uniquely identified by Product."
UC4 : "Product is uniquely identified by product code."
TC2 : "Every Product must be present in Name of Product."
```

The IG consists of the followings:

1. `F2` is a fact type called `Name of Product`.
2. `F2` has two roles (the parts between < >). The first role is played by object type `Product` (which is expressed using object type expression `O2`). The second role is played by a label type `product name`.
3. `UC3` and `UC4` are constraints involved in the IG. Both are called *uniqueness constraints*. A uniqueness constraint defines that the values that may be filled in particular role(s) must be unique.

4. TC2 is a totality constraint which states that every Product must have a name.

The IGD for the example is shown in Figure 1. Roles are presented as rectangles with unique numbers in them. The fact types are formed by the roles, for instance: F2 consists of role #5 and #6. Object type, in this case: Product, is shown by a circle surrounding some roles. Label types, in this case: product name, are shown as dash-lined circles. Uniqueness constraints UC3 and UC4 are presented as two-way arrows over roles. Totality constraint TC2 is represented as a dot in the Product end of the connecting line between Product and role #5.

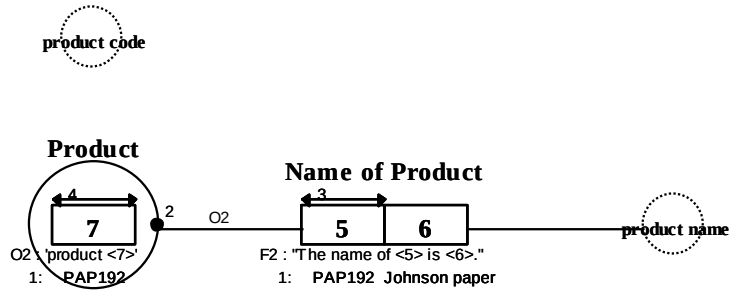


Figure 1 An example of IGD

A proper IG should be able to be used to regenerate the fact expressions from which the modeling is started. In this manner, an IG can be validated against the facts given by the domain experts. For example: suppose we provide the value PAP192 for product code and Johnson paper for product name we will have the following fact expression regenerated:

```
The name of product PAP192 is Johnson paper.
```

2.3 Pattern and Data Model Pattern

Our work is based on the concept of pattern by Christopher Alexander, a physical architect who wrote several books on patterns. According to Alexander [11], *each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.* As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves. The pattern is, in short, at the

same time a thing, which happens in the world, and the rule which tells us how to create that thing, and when we must create it (Alexander [11], Appleton [13]).

Based on the definition, three basic elements of a pattern are defined: context, problem, and solution. Most authors agree that other elements are required to fully describe a pattern, such as: name, forces, rationale, resulting context, etc. (see Table 1).

Table 1 Elements of a pattern (Appleton [13])

Element	Description
Name	A meaningful designation to refer to the pattern and the knowledge and structure it describes.
Forces	A description of the relevant forces and constraints and how they interact/conflict with one another and with goals to achieve.
Examples	One or more sample applications which illustrate: a specific initial context; how the pattern is applied to it and transforms it, and the resulting context.
Resulting Context	The state or configuration of the system after the pattern has been applied, including the consequences (both good and bad) of applying the pattern, and other problems and patterns that may arise from the new context. It describes the post conditions and side-effects of the pattern.
Rationale	A justifying explanation of steps or rules in the pattern, and also of the pattern as a whole in terms of how and why it resolves its forces in a particular way to be in alignment with desired goals, principles, and philosophies.
Related Patterns	The static and dynamic relationships between this pattern and others within the same pattern language or system.
Known Uses	Known occurrences of the pattern and its application within existing systems.

A number of publications exist with respect to data model patterns. David Hay wrote a book on data model patterns for enterprise information system using the ERM approach, specifically the CASE*Method (Hay [2]). The current works on data model patterns are not only considered important in the area of data modeling, but also in object modeling. Currently there are more researches in object oriented patterns in comparison to data model patterns. Other examples of works in the area of object-oriented patterns and data model patterns include “universal data model” by Silverston [3], “analysis patterns” by Fowler [5], “object-oriented patterns” by Coad [4], and so on. Both object-oriented patterns and data model patterns are considered complementing each other.

3 Data Model Pattern Concept

3.1 Definition of Data Model Pattern

Because we use the definition of pattern by Christopher Alexander, a data model pattern is also defined as a three-part rule which expresses a relation between a certain *context*, a *problem*, and a *solution*, each related to data modeling.

$$R = (C, P, S, \pi) \quad (1)$$

We establish the definition of pattern from the theory of *relation*. R is defined as a relation between C, P, and S. C, P, and S are defined as sets of statements of context, problem, and solution respectively; all of which are related to data modeling. Thus, π is a set of tuples which is the subset of Cartesian product: C x P x S.

3.1.1 Context

According to Alexander [11], the *context* of a pattern defines the situations in which a *problem* recurs and a *solution* is desirable. It also tells the pattern's applicability (Appleton [13]). We define context as a set of statements which describes situations in which data modeling is required. Thus, each pattern will be defined for a particular statement of such situations.

3.1.2 Problem

According to Alexander [11], the problem of a pattern defines the goals/objectives that the pattern wants to reach within the given *context* as well as the *system of forces* which is required to be "balanced" in order to achieve the goals. The *system of forces* can be viewed as the challenges, obstacles, as well as opportunities that one encounters, in this case, in a particular data modeling situation (*context*), that are required to be reconfigured (by means of the *solution*) in the pursuit of particular objectives.

Thus, in FCO-IM based data model patterns, we define that the statement of a problem consists of the following elements:

1. *Forces*, i.e. the statement of the system of forces of a pattern. In FCO-IM based data model patterns, it contains the typical fact expressions as well as constraints that one encounters within a particular context. Examples, scenario, and other descriptions may be given to explain the element.

2. *Intent*, i.e. the statement of the goal(s) to be achieved. In our data model pattern the goal is: to create an FCO-IM Information Grammar (IG) that works for a particular situation (context).

3.1.3 Solution

The *solution* of a pattern describes the rules to configure the *system of forces* in order to achieve the *goals* within a situation prescribed in the *context* (Alexander [11]). Because the goal of our FCO-IM based data model patterns is to create an Information Grammar (IG), a statement of *solution* is basically a template, based on which one can produce an IG. We call this template: *Information Grammar for Pattern (IG_P)*.

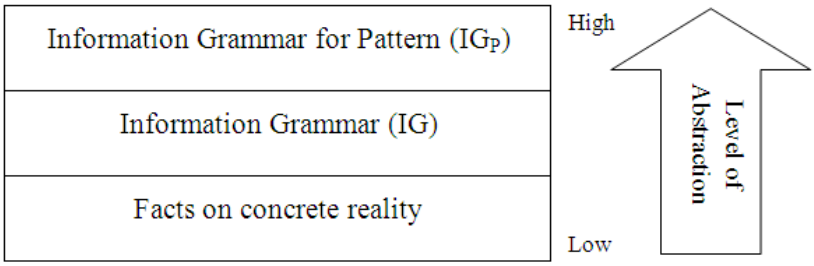


Figure 2 Abstraction levels from concrete facts to IG_P.

Figure 2 depicts the level of abstraction from facts on concrete reality to IG_P. An IG can be viewed as an abstraction of facts on concrete reality. Given several proper examples of real world facts, an IG can be used to regenerate the fact expressions (see again section 2.2). In the same manner, an IG_P is basically an abstraction of several IGs which is formed by taking into account only particular common parts of the IGs, just as an IG is formed by taking into account only common parts of fact expressions. See section 3.2 for further explanation on the concept of IG_P.

To fully describe the solution, an IG_P is required to be accompanied by other elements: the *resulting context* and the *rationale*. The *resulting context* defines the post conditions after the solution of a pattern is applied. In other words, it defines what the consequences of the application of the pattern. In our research, this means defining what kind of data model that can be resulted from the application of the solution, or in some cases, what other patterns may be emerged from the application of the pattern. The *rationale*, on the other hand, explains how the solution can be used to resolve the forces within the given context in order to achieve the goals.

3.1.4 Other Elements of Pattern

As suggested by Alexander [11] and Appleton [13], other elements of a pattern should also be described. The *example* and the *known uses* of patterns are other elements of pattern that give the description of the pattern even more clearer by giving instances of the application of the pattern. While an *example* explains in full detail of how a pattern is used in a particular case (it can be real or made-up case), *known uses* explain real world cases in which the application of the pattern is found. Every pattern must also be given a *name* and *aliases* that embodies the knowledge described within the pattern. It is used to introduce a pattern and to give the first glimpse of what the pattern might be.

Data model patterns are related to each other. The element *related patterns* is used to list the relationships of a data model pattern to others. The IG_P (see section 3.2) can be used to state explicitly the relation of a data model pattern to other patterns. In this case, the content of *related patterns* element must be consistent with the IG_P.

3.2 Information Grammar for Pattern (IG_P)

An *Information Grammar for Pattern* (IG_P) provides the template to produce an FCO-IM conceptual data model: the *Information Grammar* (IG). It does not only provide the configuration to create an IG for a particular situation, but it also defines how patterns relate to each other. An IG_P of a pattern can contain the rule in which other patterns must be generated.

As the central element of the solution of an FCO-IM based data model pattern, the structure of IG_P is required to be explored further. Since an IG_P is used to generate an IG, all FCO-IM notations are used (see section 2.2 and further in Bakema, et al. [7]). Nevertheless, there are some requirements in the description of an IG_P in which new notations are needed. The new notations are listed in Table 2.

An example of an IG_P (from *Single Identification Pattern*, see section 3.4) is as the following:

```
(object):
[(F1):"[(expression-1)]<(object-id-1#(1))|(G1#(1))>[(expression-
2)<(object-id-2#(2))|(G1#(2))>]*[(expression-3)]."]
(O1):'[(expression-4)]<(object-id-1#(1))|(G1#(1))>[(expression-
5)][<(object-id-2#(2))|(G1#(2))>]*[(expression-6)]'
(UC1):"(object) is uniquely identified by (object-id-
1#1)|(G1#(1))[, (object-id-2#(2))|(G1#(2))]*."
```

Table 2 New notations for IG_P

No.	Notation	Meaning
1.	(α)	α is something to be "generated". It means that α is required to be replaced by one of the following: - A *term*, i.e. a word/phrase that is used to name an object type, a fact type or part of a fact type, or a label type, or part of a sentence. - A *pattern*; it means that α will be replaced wit*h a term* that come with the application of a pattern. The application of a pattern will also introduce other fact types, object types, or label types. There are two ways to indicate a pattern: 1) the name/code of the pattern 2) the name/code of a category in which a pattern can be chosen.
2.	α#β	β is used to indicate a role number or role alias. This expression is used to indicate that β is played by α.
3.	[α]	α is generated 0 or 1 time.
4.	[α]*	α is generated 0 or n times.
5.	[α]+	α is generated 1 or n times.
6.	α|β	Either α or β is generated, but not both.
7.	α||β	Either α or β is generated and can be both.

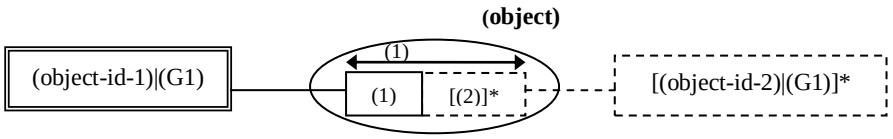
The following information is stated in the IG_P:

1. `expression-1, expression-2, expression-3, expression-4, expression-5, expression-6, object, object-id-1, object-id-2, F1, O1, UC1` are to be replaced with *terms*.
2. `1` and `2` are to be replaced by role numbers or aliases.
3. `G1` is to be replaced with a pattern from category `G1`: *Patterns based on the identification of an object* (see section 3.4 for a list of all pattern categories as well as the related patterns).
4. The fact type expression `F1` is optional. It means that when the pattern is used, `F1` can be generated or not.
5. A pattern of category `G1` (which leads to an object type) or a term to replace `object-id-2` (which leads to a label type) is expected to play role `#2`. There can be several instances of a pattern on category `G1` or a term to replace `object-id-2`. In this way, every pattern is related to one another.

As an FCO-IM IG is accompanied by a diagram, an IG_P is also equipped with a diagrammatic version of it. The concept of the diagram is based on the concept of FCO-IM Information Grammar Diagram (IGD). The diagram of IG_P is called the IG_P Diagram (IG_PD). We add the following symbols from the concept of IGD:

1. A dashed-lined box or line is used to indicate that a defined term or pattern can be generated or not.
2. A double-lined box or line is used to indicate that a defined term or pattern must be generated.

The corresponding IG_PD for the IG_P example is shown in Figure 3.



[(F1) : "[(expression-1)]<(1)>[(expression-2)][<(2)>]*[(expression-3)]."]
(O1) : '[(expression-4)]<(1)>[(expression-5)][<(2)>]*[(expression-6)]'

Figure 3 An example of IG_PD

According to the diagram shown in Figure 3, in role `#1`, either a pattern of category `G1` (which leads to an object type) or a term to replace `object-id-1` (which leads to a label type) must be generated. Role `#2` on the other hand can be generated or not and when it is generated, it will be played by an object type (from the application of a pattern of category `G1`) or a label type (from the replacement of `object-id-2` with a particular term).

An example of an IG that is generated from the IG_P is as the following:

```
Student:
F2:"There is a student <firstname> <surname>."
O2:'student <firstname> <surname>'
UC2:"Student is uniquely identified by firstname, surname."
```

1. `expression-1` is replaced by the phrase: "`There is a student `".
2. `expression-3` and `expression-6` are omitted.
3. `expression-5` are replaced by a single space.
4. `expression-4` is replaced by the phrase: "`student `".
5. `object` is replaced by the word: "`Student`".
6. `object-id-1` is replaced by the word: `firstname` (which becomes a label type).
7. `expression-2` and `object-id-2` are instantiated 1 time. `expression-2` is replaced by a single space, while `object-id-2` is replaced by the word: `surname` (which becomes a label type).
8. `F1` is replaced by `F2`, `O1` is replaced by `O2`, `UC1` is replaced by `UC2`.
9. Role `#1` is replaced by role `#3` and role `#2` is replaced by role `#4`. These replacements do not appear in the IG. Instead, it can be found in the resulting IGD.

The corresponding IGD based on the IG and IG_PD is as shown in Figure 4.

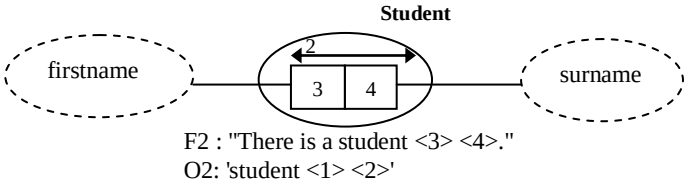


Figure 4 An example of an IGD generated based the IG_P and IG_PD example

3.3 Example of a Data Model Pattern

The example used in Section 3.2 is an example of IG_P for *Single Identification Pattern* (see Section 3.4). In this section, we provide a full example of a data model pattern called the *Parent-Child Pattern* (code: G2P3, a pattern from category G2, see Section 3.4).

1. **Name**
   Parent-Child Pattern, also known as: *Tree Pattern*.
2. **Context**
   There are several objects of the same type arranged into a hierarchy. Some of the objects have higher level of hierarchy than the others. The former objects are usually called the parents, while the latter are usually called the children. Thus, the relationship is called parent-child. There can be several levels of hierarchy, where a parent can be a child at the same time. There is one and only one object which possesses the highest hierarchy which is called the root.
3. **Problem**
   - **Intent**
     To create an Information Grammar (IG) for modeling the parent-child relationships among objects of the same type.
   - **Forces**
     Suppose have two objects called A and B with A is the parent and B is the child, then the facts that state the relationship between A and B typically show the hierarchy, for example:
     "A is the parent of B."
     "A is higher than B."
     "B is the child of A."
     "B is lower than A."
     Sometimes, the parent-child relationship is given other names, such as supervision. Thus, the facts may look like:
     "A supervises B."

"B is supervised by A."
The rules that define the relationship are:

- B cannot have other parent than A, but A can have more than one child.
- An object cannot be the parent or the child of itself either directly or indirectly through a chain (A is the child of B, B is the child of C, C is the child A, for example).
- There is an object which has the highest level in the hierarchy which is called the root.
- All objects have the same way of identification (thus, they are of the same type).

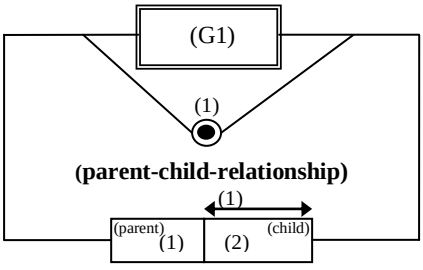
4. Solution

- **IG_P**

The IG_P for the pattern is as the following:

```
(parent-child-relationship):
(F1):"<(G1#(parent))>(expression-1)<(G1#(child))>."||
(F2):"<(G1#(child))>(expression-2)<(G1#(parent))>."
(UC1):"(parent-child-relationship) is uniquely identified by
(G1#(child))."
[(TC1):"Every (G1) must be present either as a parent or as a
child in (parent-child-relationship)."]
(C1):"To ensure that the value of a parent node is not the same
the value of a child node: in a tuple, (G1#(parent)) cannot be
the same value as (G1#(child))."
(C2):"To ensure that there is at least one root of the tree:
there must be at least one value of (G1#(parent)) which is not
(G1#(child))."
```

The IG_PD for the IG_P is as shown in Figure 5.



Other constraints:
(C1) : "To ensure that the value of a parent node is not the same the value of a child node: in a tuple, <(1)> cannot be the same value as <(2)>."
(C2) : "To ensure that there is at least one root of the tree: there must be at least one value of <(1)> which is not in <(2)>."

(F1) : "<(1)>(expression-1)<(2)>."
(F2) : "<(2)>(expression-2)<(1)>."

Figure 5 IG_PD for *Parent-Child Pattern.*

- **Rationale**
  The Fact type `F1` or `F2` provides the modeling of the typical fact expressions of parent-child relationship. `expression-1` and `expression-2` must be replaced by a phrase which expresses the parent-child hierarchy. Uniqueness constraint `UC1` over role `#2/child` (which is the child part of the parent-child relationship) keeps the rule that a child has only one parent, but a parent can have more than one child by assigning a uniqueness constraint over the. Constraint `C1` ensures that that a child cannot be its own parent at the same time. Constraint `C2` ensures that there must be at least one object that will be the root of a tree structure. If there is more than one of such object, then there is more than one tree defined. The combination of constraint `UC1` and `C2` ensures that there will be no object that indirectly becomes a child/parent of itself. `UC1` will prevent an object from being in the hierarchy more than once as a child (which is at the same time probably a parent), while `C2` will prevent a root object to be a child at the same time. Totality constraint `TC1` ensures that every object will be a part of a hierarchy. It can be omitted, however, in the case it is okay for an object not to be part of a hierarchy.
  As suggested in the IG_PD, there is only one application of a pattern of category `G1` for both parent and child. The application of a pattern of category `G1` provides an object type. This maintains the rule that the parent and child are of the same type.
- **Resulting Context**
  An FCO-IM Information Grammar (IG) that expresses parent-child relationship between several objects of the same type is the result of the application of the pattern.

5. Examples

An example of this pattern is on a supervision hierarchy of employees in a company in which an employee can be the supervisors of several other employees. An example of such hierarchy is shown in Figure 6.

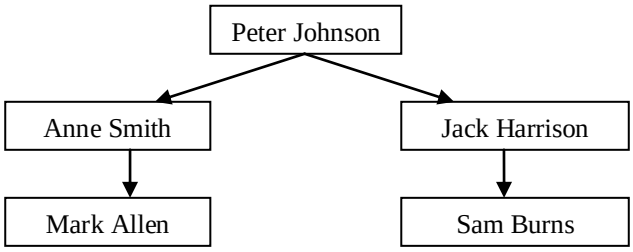


Figure 6 An example of supervision hierarchy.

An IG that is generated based on the IG_P is as the following:

```
Supervision:
F1:"<Employee:O1> supervises <Employee:O1>."
Employee:
O1:'employee <firstname> <surname>'
UC1:"Supervision is uniquely identified by Employee#child."
UC2:"Employee is uniquely identified by firstname, surname."
TC1:"Every Employee must be present either as a parent or as a
child in Supervision."
C1:"To ensure that the value of a parent node is not the same the
value of a child node: in a tuple, Employee#parent cannot be the
same value as Employee#child."
C2:"To ensure that there is at least one root for the tree: there
must be at least one value of Employee#parent which is not
Employee#child."
```

The bold parts are generated based on the IG_P of *Parent-Child Pattern*. The rests are generated by a pattern of category G1, in this case *Single Identification Pattern* (code: G1P1, see section 3.4).

The IGD generated based on the IG_PD is as shown in Figure 7.

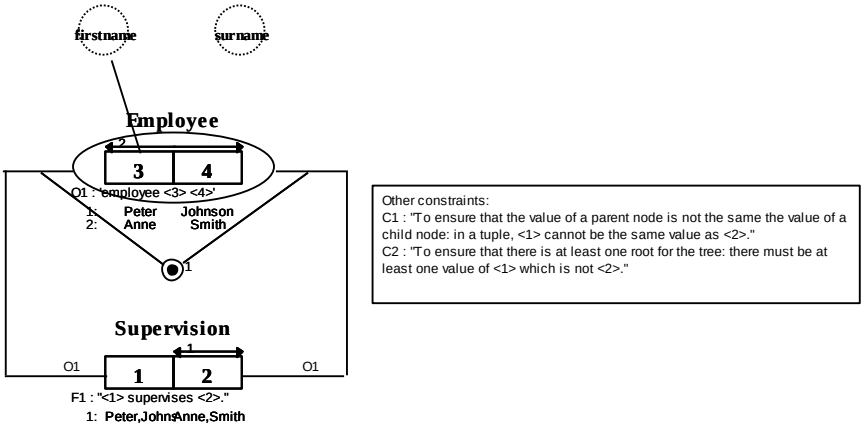


Figure 7 IGD for Supervision example based on the IG_PD of *Parent-Child Pattern.*

6. Known Uses

Some cases in which this pattern can be used are: the hierarchy of employees within an organization (as shown as example), the plant taxonomy in biology, and the hierarchy of geographical locations are some of the cases in which the pattern is used.

7. Related Patterns
As shown in the IG_P, from this pattern, one can generate a pattern of category G1 (see section 3.4 for the list of all patterns in category G1).

3.4 List of Data Model Patterns

Most existing works on data model patterns emphasize on the so called domain-specific data model patterns, especially for enterprises (see for example: Hay [2], Silverston [3]). However, these data model patterns are only useful when you have to model the particular domain.

We have described several generic data model patterns based on the structures of the fact expressions. The fact expressions can be of different meanings, but as long as they retain the same structures, they are modeled in the same fashion. These structures are encountered repeatedly by data modelers in their works. Thus, in our opinion, data model patterns that are based on such structures will be more useful. We define four categories from which particular structures can be determined based on the facts:

1. *The facts expressing the existence of an object.*
   Expressing an object is a central concept in FCO-IM which distinguishes FCO-IM with other data modeling methods including the fact oriented ones. This leads to different ways of identifying an object depending upon how it is expressed. Data model patterns belong to this category are grouped in pattern category 1: *Patterns based on the identification of an object*.
2. *The facts expressing a collective relationship between two or more objects.*
   Several facts may describe a collective relationship between two or more objects which forms a special structure that can be found in a lot of data modeling cases, such as tree structure. The collective relationship usually cannot be determined by only a single fact, but must be determined based on the observation of several facts altogether. The objects are commonly of the same object types. Data model patterns belong to this category are grouped together in pattern category 2: *Patterns on the collection of objects*.
3. *The facts expressing the relationship between two or more objects.*
   Some facts declare the relationship between two or more objects without having a collective relationship among them. The objects are commonly of different object types. Data model patterns belong to this category are grouped together in pattern category 3: *Patterns based on the relation between two or more objects.*
4. *The facts expressing architectural relationship among two or more groups of objects which are connected together in a particular relationship.*
   Objects that are related in particular relationships can be linked together to form an architectural level relationship. Data model patterns belong to this

category are grouped into pattern category 4: *Patterns based on the architecture of the objects.*

For each category, we defined several data model patterns. Table 3 provides the list of the categories as well as the related data model patterns that we have described so far. Some of the patterns are described briefly in Azizah, et al. [9]. The idea of pattern G4P1 is based on our work described in Liem, et al. [14].

Table 3 List of pattern categories and the respective data model patterns.

Pattern Category	**Pattern**	
	Code	**Name of Pattern**
Code: G1 Patterns based on the identification of an object	G1P1	Single Identification Pattern
	G1P2	Recursive Identification Pattern
	G1P3	Set Identification Pattern
	G1P4	Generalized Identification Pattern
	G1P5	Synonymy Pattern
	G1P6	Homonymy Pattern
	G1P7	Subtype Pattern
Code : G2 Patterns on collection of objects	G2P1	Graph Pattern
	G2P2	Sequence Pattern
	G2P3	Parent-Child Pattern
Code : G3 Patterns based on the relation between two objects	G3P1	Attribute Pattern
	G3P2	Mapping Pattern
	G3P3	Assembly-Part Pattern
	G3P4	Supertype-Subtype Pattern
Code : G4 Patterns based on the architecture of the objects	G4P1	Viewpoints to A Dataset Pattern

4 Case Study: Geographical Location

The example shown in the description of *Parent-Child Pattern* has shown how two patterns work together to create an Information Grammar (IG). In this section, we provide a case study on geographical location in which three patterns are used together to generate an IG.

4.1 Case Description

Countries, in general, are divided into geographical locations which belong to a geographical hierarchy. The country itself is also considered as a geographical location. For instance: the country Indonesia is divided into provinces; each province is divided into districts/municipalities (*kabupaten/kota*); each district is divided into subdistricts (*kecamatan*); etc. Other countries may have different

way of organizing their geographical locations. Several facts on the hierarchy of geographical location are:

```
Location Indonesia has higher geographical hierarchy than
location Jawa Timur.
Location Jawa Timur has higher geographical hierarchy than
location Ngawi.
```

Each geographical location (or location, for short) has a particular geographical level such as country, province, district, etc. Several facts related to the geographical level of a location are:

```
The geographical level of location Indonesia is country.
The geographical level of location Jawa Timur is province.
The geographical level of location Ngawi is district.
```

Several rules related to the case are defined as the following:

1. Each location is given a unique name. This is not the case in the real world, however. We define this to simplify the case.
2. The name of a geographical level name is also unique.
3. The geographical level of a location must be known.
4. Every recorded location must be present in one hierarchy.

4.2 Modeling the Case Using the Data Model Patterns

Parent-Child Pattern (which is described in Section 3.3) is used to model the hierarchy of location. The geographical level of a location is modeled using *Attribute Pattern* (see again Table 3). The IG_P for *Attribute Pattern* is as the following:

```
(attribute-of-object):
(F1):"[(expression-1)]<(G1#(1))>[(expression-2)]<(attribute's-
name)|(G1#(2))>[(expression-3)]."||
(F2):"[(expression-4)]<(G1#(2))>[(expression-5)]<(attribute's-
name)|(G1#(1))>[(expression-6)]."
(UC1):"(attribute-of-object) is uniquely identified by (G1#(1))."
[(UC2):"(attribute-of-object) is uniquely identified by
(attribute's-name)|(G1#(2))."]
[(TC1):"Every (G1#(1)) must be present in (attribute-of-
object)."]
[(TC2):"Every (G1#(2)) must be present in (attribute-of-
object)."]
```

When *Parent-Child Pattern* and *Attribute Pattern* are applied, another pattern is required to be generated. The identification of location and geographical level is dealt with *Single Identification Pattern* (which IG_P is described in section 3.2).

An IG that can be generated based on the application of the three patterns is shown below. Rows marked by number 1 are generated from *Parent-Child Pattern*. Rows marked by number 2 are generated from *Attribute Pattern*. Rows marked by number 3 are generated from *Single Identification Pattern*.

```
1  Geographical Hierarchy:
1  F1:"<Location:O1> has higher geographical hierarchy than
   <Location:O1>."
1  UC1 : "Geographical Hierarchy is uniquely identified by
   Location#child."
1  TC1 : "Every Location must be present either as a parent
   or as a child in Geographical Location."
1  C1:"To ensure that the value of a parent node is not the
   same the value of a child node: in a tuple, Location#parent
   cannot be the same value as Location#child."
2  Geographical Level of Location:
2  F2:"The geographical level of <Location:O1> is <Geographical
   Level:O2>."
2  UC3 : "Geographical Level of Location is uniquely identified
   by Location."
2  TC2 : "Every Location must be present in Geographical Level of
   Location."
3  Location:
3  O1:'location <location name>'
3  UC2:"Location is uniquely identified by location name."
3  Geographical Level:
3  F3:"There is a geographical level <geographical level name>."
3  O2:'<geographical level name>'
3  UC4:"Geographical Level is uniquely identified by geographical
   level name."
```

The corresponding IGD generated based on the IG_Ps of the patterns as well as their IG_PDs is shown in Figure 8.

5 Conclusions

In this paper, we described our work on the concept of *[conceptual] data model pattern* based on the classical definition of pattern by Christopher Alexander which is redefined based on the relation theory. Based on our definition, a pattern is a relation between the statements of *context*, *problem*, and *solution*. In describing the patterns, other supporting elements are also required, such as name, examples, and known uses.

-

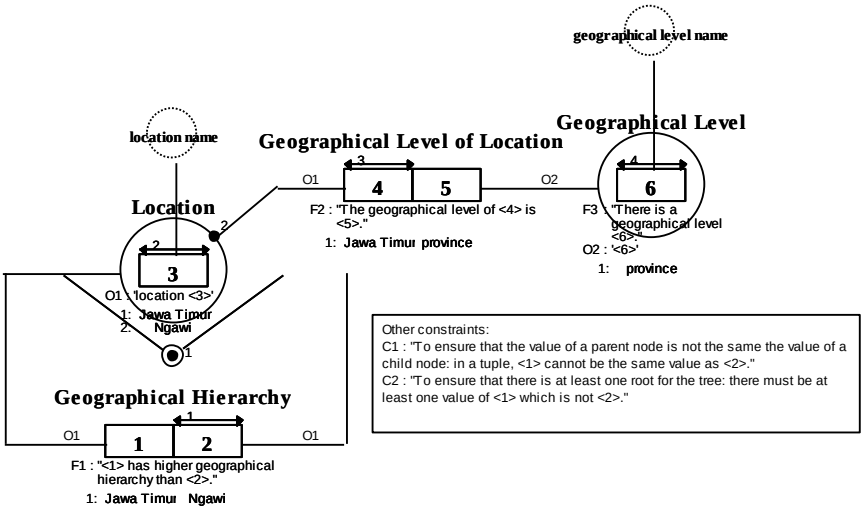


Figure 8 IGD for geographical location case study.

Our concept of data model pattern is also based on the concept of *Fully Communication Oriented Information Modeling (FCO-IM)*, a fact oriented data modeling approach. We defined a concept called *Information Grammar for Pattern (IG_P)* in solution part of a data model pattern which works as a template to generate an Information Grammar (IG), the FCO-IM conceptual data model. An IG_P can be used to show how data model patterns can relate to each other by means of generating other patterns. In the case study, this generating nature of the data model patterns helps building an IG. Based on the concept, we have defined 15 data model patterns which are arranged into 4 categories.

Our work is expected to contribute in the area of data modeling by providing a concept on data model pattern. This work also provides a contribution in the development of FCO-IM method by providing an extension of FCO-IM notations for defining a data model pattern. Although we did not discuss how other data modeling methods describe a data model pattern, we have observed that the new concept provides a precise way for describing a data model pattern as well as for applying them in a real case. The comparison of our concept of data model patterns with other concepts is left as a subject of a next paper.

Based the concept of data model pattern, we developed the concept of *pattern language* of data model patterns. This concept has been described briefly in Azizah, et al. [10]. We are also required to test whether our concept of data model patterns and pattern language, as well as the data model patterns can be

used to provide high quality conceptual data models. This is left for the subject of our next paper.

References

[1] Simsion, G. & Witt, G., *Data Modeling Essentials*, Third Edition, Morgan Kaufmann Publishers, 2005.

[2] Hay, D.C., *Data Model Patterns: A Convention of Thought*, Dorset House Publishing, New York, 1996.

[3] Silverston, L., *The Data Model Resource Book*, Revised Edition, **(1&2)**, John Wiley & Sons Inc., 2001.

[4] Coad, P., North D. & Mayfield, M., *Object Models: Strategies, Patterns, and Applications*, Prentice Hall, 1997.

[5] Fowler, M., *Analysis Patterns Reusable Object Models*, Addison Wesley, 1996.

[6] Nicola, J., Mayfield, M. & Abney, M., *Streamlined Object Modeling: Patterns, Rules, and Implementation*, Prentice Hall, 2001.

[7] Bakema, G., Zwart, J. P. & Lek, H. van der, *Fully Communication Oriented Information Modeling (FCO-IM)*, HAN University, 2002.

[8] Azizah, F.N. & Bakema, G., *Data Modeling Patterns using Fully Communication Oriented Information Modeling (FCO-IM)*, ORM Workshop 2006 (part of OnTheMove Federated Conferences and Workshops 2006), working papers, Montpellier, France, 2006.

[9] Azizah, F.N., Bakema G.P., Sitohang, B. & Santoso, O.S., *Generic Data Model Patterns using Fully Communication Oriented Information Modeling (FCO-IM)*, International Journal on Electrical Engineering and Informatics, **(1)**, 2009.

[10] Azizah, F.N., Bakema G.P., Sitohang, B. & Santoso, O.S., *Information Grammar for Patterns (IG_P) for Pattern Language of Data Model Patterns Based on Fully Communication Oriented Information Modeling (FCO-IM)*, 2010 ORM Workshop (part of 2010 OnTheMove Federated Conferences and Workshops), working papers, Crete, Greece, 2010.

[11] Alexander, C., *The Timeless Way of Building*, Oxford University Press, USA, 1979.

[12] Silberschatz A., Korth H.F. & Sudarshan S., *Database System Concepts*, Fourth Edition, McGraw Hill, 2002.

[13] Appleton, B, *Pattern and Software: Essential Concepts and Terminology*, http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html, (19/04/2006).

[14] Liem, I. & Azizah, F. N., *Metadata Approach in Modeling Multi Structured Data Collection Using Object Oriented Concepts*, proceeding in International Conference on Networking and Information Technology (ICNIT) 2010, June 2010, Manila, Philippines.