An algebra for structured documents in the context of the object-oriented approach
Sylviane Schwer, Haider Hamza, André Flory

To cite this version:
Sylviane Schwer, Haider Hamza, André Flory. An algebra for structured documents in the context of the object-oriented approach. 19 pages. 1999. <hal-00265817>
An algebra for structured documents in the context of the object-oriented approach

Haider Hamza
INSA, L3SI 20, Avenue Albert Einstein 93421 Villeurbanne France
e-mail : hamza@iisifiy.insa-lyon.fr

Sylviane R. Schwer
Université Paris 13, LIPN, av. J. B. Clément, 93430 Villetaneuse, France
e-mail : schwer@lipn.univ-paris13.fr

André Flury
INSA, L3SI 20, Avenue Albert Einstein 93421 Villeurbanne France
e-mail : flury@iisifiy.insa-lyon.fr

Summary

The aim of this paper consists in defining an algebra allowing the request of structured documents in the context of object oriented approach. The operators of this algebra are defined in conformity with the concepts of the object model. It takes into consideration not only the embedding of structures and links between them, but also the informational aspect of the document as well as its structural aspect.

The operands of the proposed algebra are only one type: they are subdatabases. A subdatabase is made of a collection of the databases objects, grouped in classes and interconnected through links. This algebra responds to two objectives. Firstly, it represents a basic nucleus of a declarative query system; it contains the whole of the elementary operations which will be used in the resolution of the request. Secondly, it provides facilities to users to formulate their requests and manipulate the documentary database.

Key Words : Structured documents, Object-oriented, Documentary database, Query, Algebra

1. Introduction

Since the emergence of the relational model, many researches have addressed the problem of the development of documentary databases using the technology of database management relational systems (RDBMS). For the development of documentary applications, the interest of this approach lies in the possibility of taking advantage of all the functionality of RDBMS. Nevertheless the capacities of abstraction and the expressiveness of the relational model are insufficient for handling structured documents. Indeed, the representation of the fragments of the documents by several tables results in a heavy modeling and decreases considerably the performance of the system.

Researches aiming at a new generation of DBMS has been carried out to extend the models of data representation and to increase the power of manipulation languages. Indeed, new needs have emerged requiring references to the stocked information in terms of the perception of the user and not simply according to their model inside the database. The object oriented DBMS gives nowadays a quite rich model capable of representing complex data. A document can be modeled by a complex object the components of which may be formed by other objects belonging to different classes. The classes are organized in a hierarchy of inheritance in which the objects of different classes inherit the relations of inclusion between them. The technology of OODBMS is well adapted to the management of structured documents such as arborescences of objects with associated methods [Angh89, Beno89, Moua89, Hamu90].

The retrieval of informations is one of the most important functionality of a management system of documentary databases. By system of retrieval of information, we mean the whole mechanisms which allows the user to select documentary information. One of the essential objectives of a retrieval system of informations is to make easy the restitution of a portion of information from a documentary database, in response to a user's request. Different strategies of information retrieval were developed, and most of them were built around an algebra. In fact, G. ting [Guin89] has proposed an algebra made of a set of operators which permits to take into account the different informational aspects of a document. But this algebra is not well adapted to an object-oriented system, because it doesn't consider the inter-objects relationships. In the case of the algebra for the OODBMS, several works were carried out [Alb93, Ctie90, Hamu95, Liu93, Shum90, Shum90, Subo95]. However, these algebra are not really adapted to structured documents, since they only take into consideration the informational aspect of the data, and not at all their structural aspect. This is the case, for instance, of the algebra of Show and Zdonik, which is an extension of the algebra of the denormalized models that takes into account some object-oriented concepts. The EXCESS algebra [van91], is a many-sort algebra i.e. the operators are defined by types of operators. Such an approach in a documentary retrieval context puts in question the uniformity of the operators and the reusing of the results of the requests for a new query. The algebra of Liu [Liu93] is based on a clear distinction between links of association and links of aggregation. We think that in fact this distinction is not relevant: two concepts can have different visions of the same reality. While one can model a relation as a link of composition, most models among the reviewed aboved treat it as a link of association.

The objective of our work consists in defining an algebra that allows the interrogation of documents in the context of an object-oriented approach. The basis of manipulation is the embedding of structures and the links between structures [Schw97]. This algebra permits to take into account the informational aspect as well as the structural aspect of documentary databases.

This paper is divided into 5 sections. The first one being this introduction, section 2 is devoted to the basic concepts of the proposed algebra. Section 3 is dedicated to the description of the algebraic operators and section 4 to examples of the algebraic operations. The last section consists in our conclusion and perspectives.

2. Formalization of a documentary database

In this part we define the basis of our algebra. Starting from a clean definition of what is a documentary database, we defined what is a subdatabase, the single type of our algebra and their operands which fit to the object-oriented ontology.

2.1. Documentary database

As usual in database theory, a documentary database is defined y a schema and an instance of this schema.

Definition 1: Documentary database schema

The documentary database schema can be defined by a finite acyclic connect graph. Formally, a schema $\Sigma$ of documentary database is defined by the triplet $(C, \rho, \lambda)$ where:

- $C = \{c\}$ is a set of classes, where each class $c$, is characterized by some properties $p_{c0}, p_{c1}, ..., p_{ck}$.
- $\rho: C \rightarrow 2^{C}$ is an hierarchy function which associates a set of classes to a given class. We note by $\rho^{-1}$ the inverse relation of the hierarchy function, $\rho^+$ the transitive hierarchy function of order $n$, $\rho^*$ the transitive hierarchy function and $\rho^t$ the strict transitive hierarchy function which are defined as follows:
  - $p^n(c)\doteq c$
  - $p^n(c)\doteq p^{n-1}(c))$
  - $p^*(c)\doteq p^n(c)$
  - $p^t(c)\doteq p^*(c) \cap p^*(c)$
- $\lambda: C \times C \rightarrow 2^{C}$ is a Roll function which associates a set of rolls to a pair of classes. The schema is endowed by the following properties :
  1) $\text{Root}(\Sigma) = \{c \in C \mid \forall c' \in C, c' \notin \lambda(c)\} \cup \emptyset$ (due to the fact that the graph is acyclic and finite)
  2) $(\forall c \in C). (c \notin \lambda(p^t(c)))$ (due to acyclicity)
  3) $(\forall c \in \text{Root}(\Sigma)). (p^*(c) \cap \emptyset)$ (this is the definition of the Root)
  4) $(\forall c, c' \in \text{Root}(\Sigma)). (p^*(c) \cap p^*(c') \cap \emptyset)$ (this is due to the connection property)

Example:

Let $\Sigma = (C, \rho, \lambda)$ be the schema of the documentary database represented by fig. 1. The graph which corresponds to this schema is represented by fig. 2.
The documentary database schema can also be defined by a finite acyclic connected graph. We denoted it a $G \rightarrow$ graph.

**Definition 1**: Documentary database

A documentary database is defined by: $(E, O, A)$ where $E \rightarrow C, p, r)$ is a schema of the database, $O$ is a set of objects and $A$ a set of inter-objects links $A \subseteq O \times O$. If we note by $\tau$ the total function $(e \rightarrow C)$ which returns for every object its class, then $(E, O, A)$ must verify the following constraint:

- $\forall (a_i, a_j) \in A, \tau(a_j) \in \tau(\tau(a_i))$

**2.2. Documentary subdatabase**

A user who wants to formulate queries on a database, is rarely interested by the whole entities in the database. In most cases, he wants to formulate queries on a portion of this database. For example, a user would like to formulate queries only on the Author class and Document class of our illustration.
We defined a subdatabase of a database D as a database which is merged into the database D.

Definition 3: Subschema
A subschema \( \psi \) = \((C_\psi, \rho_\psi, \lambda_\psi)\) of a schema \( \Sigma = (C_\Sigma, \rho_\Sigma, \lambda_\Sigma)\) is included in the \( \Sigma \) graph, such that:
- \( C_\psi \subseteq C_\Sigma \)
- \( \forall \rho \in C_\psi \quad \rho_\psi \subseteq \rho_\Sigma \)
- \( \lambda_\psi(e_1, e_2) \subseteq \lambda_\Sigma(e_1, e_2) \)

Remarks:
- A schema is a subschema of itself.
- A class of a schema is a subschema of the schema.
- \( \emptyset = (C_\Sigma, \rho_\Sigma, \lambda_\Sigma) \) is a subschema of any schema.

Example:
Fig. 4 represents the graph of the subschema \( \psi = (C_\psi, \rho_\psi, \lambda_\psi) \) which is defined as:
\[ C_\psi = \{ \text{Document (title, type, creation-date), Chapter (title, creation-date), Reference (reference) } \} \]
\[ \rho_\psi(\text{Document}) = \{ \text{Chapter, Reference} \} \]
\[ \rho_\psi(\text{Reference}) = \emptyset \]

![Fig. 4: graphical representation of a subschema](image)

Definition 4: An instance of a subschema
An instance of a subschema \( \psi = (C_\psi, \rho_\psi, \lambda_\psi) \) is a set of objects and a set of links between objects which respectively correspond to the classes in the subschema and to the links between classes. Formally, an instance is defined by the pair:
\( (O, A) \)
where \( O \) is a set of objects and \( A \) a set of inter-objects links \( A \subseteq O \times O \) with the following constraint:
- \( \forall (x, y) \in A, t(y) \in \rho(t(x)) \)

Example:
Fig. 5 gives a graphical representations of an instance of the subschema that is represented in fig. 4.

![Fig. 5: An example of an instance of a subschema](image)

Definition 5: Subdatabase
A subdatabase is defined by a subschema and by an instance of this subschema. Formally, a subdatabase is defined by the triple:
\( (\psi, O, A) \)

Example:
An example of a subdatabase is shown in fig. 6.

![Fig. 6: A graphical representation of a subdatabase](image)
Definition 8: Distance between two subschemas

Given two subschemas \( \psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1}) \) and \( \psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2}) \), the distance between \( \psi_1 \) and \( \psi_2 \), denoted by \( D(\psi_1, \psi_2) \), is defined as:

\[
D(\psi_1, \psi_2) = \min \{ \text{Len}(P) | P \in \text{All}(\psi_1, \psi_2) \}
\]

Example:
The following is a distance between subschemas \( \psi_1 \) and \( \psi_2 \) shown in figure 6:

\[
D(\psi_1, \psi_2) = \min \{ \text{Len}(\text{Document, Reference}) \} = 2
\]

Definition 9: The upper limit of two subschemas

Given two subschemas \( \psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1}) \) and \( \psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2}) \), the upper limit \( \psi_1 \lor \psi_2 \) is the least schema containing \( \psi_1 \) and \( \psi_2 \). The upper limit can be defined as the union of \( \psi_1 \) and \( \psi_2 \) and all the paths between \( \psi_1 \) and \( \psi_2 \) that have their lengths equal to \( D(\psi_1, \psi_2) \). Let us consider the set \( S \) of all the paths between \( \psi_1 \) and \( \psi_2 \) that have their lengths equal to \( D(\psi_1, \psi_2) \). The set \( S \) is defined as:

\[
S = \left\{ C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1} \right\} \cup C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2} \right\} \cup \{ P \mid \text{Len}(P) = D(\psi_1, \psi_2) \}
\]

Formally, the upper limit of two subschemas \( \psi_1 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1}) \) and \( \psi_2 = (C_{\psi_2}, \rho_{\psi_2}, \lambda_{\psi_2}) \), is defined as:

\[
\psi_1 \lor \psi_2 = (C_{\psi_1}, \rho_{\psi_1}, \lambda_{\psi_1})
\]

where:

- \( C_{\psi_1} = C_{\psi_1} \cup C_{\psi_2} \)
- \( \rho_{\psi_1} = \rho_{\psi_1} \cup \rho_{\psi_2} \)
- \( \lambda_{\psi_1} = \lambda_{\psi_1} \cup \lambda_{\psi_2} \)

Example:
An example of an upper limit of two subschemas is shown in fig. 8.

5. Algebraic operators

We have shown that the set of subdatabase of a database is closed under \( \lor \) and \( \land \). We now prove that this set is also closed under algebraic operators: select, project, associate, union, intersection, difference, join, grouping, split, distribution.
3.1. Select (σ)

The Select is a unary operator, which operates on a subdatabase to produce a new subdatabase in which objects and inter-objects links satisfy a specified predicate. The resultant subdatabase has the same schema as the operand subdatabase, and the instances of the resulting subdatabase are a subset of the instances of the operands which satisfy the predicate. The select operation is denoted σ(X[P]) where X is an operand subdatabase and P is a predicate. The predicate P is a logical expression that is evaluated to true or false. This logical expression is composed by terms interconnected by logic operators (and, or, not). Each term can be, either a condition on an attribute value, or a condition on an inter-object link. The condition on an attribute values have the following form:

\[ a \theta b \]

where \( a \) and \( b \) represent attributes.
- If \( a \) and \( b \) are integer constants, then \( \theta \) can be: \( =, <, >, \leq, \geq \)
- If \( a \) and \( b \) are Boolean, then \( \theta \) can be: \( =, \neq \)
- If \( a \) and \( b \) are string constants, then \( \theta \) can be: \( =, \neq, <, >, \leq, \geq \)
- If \( a \) is a text and \( b \) is a string, then \( \theta \) can be: \( =, \neq, <, >, \leq, \geq \)

To express the conditions on the cardinalities of links, we use the symbol \( \ni \). The conditions on the cardinalities of links have the following form: \( \forall \theta \exists \beta \alpha \), where \( \beta \) and \( \alpha \) are classes, \( \theta \) is a comparison operator that can be: \( =, \neq, <, >, \leq, \geq \), and \( \alpha \) is a numeric constant. For example, the predicate `Document.\#Author = 2` means that the object of the Document class must be linked to only two objects of the Author class.

Example:

`σ(title = "computer", author = 2)`

To express the conditions on the cardinalities of links, we use the symbol \( \ni \). The conditions on the cardinalities of links have the following form: \( \forall \theta \exists \beta \alpha \), where \( \beta \) and \( \alpha \) are classes, \( \theta \) is a comparison operator that can be: \( =, \neq, <, >, \leq, \geq \), and \( \alpha \) is a numeric constant. For example, the predicate `Document.\#Author = 2` means that the object of the Document class must be linked to only two objects of the Author class.

Example:

`σ(title = "computer", author = 2)`
3.4. Union, Intersection and Difference (\(\cup, \cap, -\))

The **union** operator is a binary operator which constructs a new subdatabase by combining two subdatabases. The subschema of the result subdatabase is defined as the upper limit of the subschemas of the operands subdatabases. The instance of the result subdatabase is constituted by the union of the set of instances of operands which are extended to the subschema of the result subdatabase. The union operation is denoted \(X \cup Y\).

The **intersection** operator, as the union operator, is a binary operator which constructs a new subdatabase by combining two subdatabases. The subschema of the result subdatabase is defined as the lower limit of the subschemas of the operands subdatabases. The instance of the result subdatabase is constituted by the intersection of the set of instances of the subdatabases operands. The intersection operation is denoted \(X \cap Y\).

The **difference** operator, as the union and intersection operators, is a binary operator which constructs a new subdatabase by subtracting from a subdatabase \(X\) of another subdatabase \(Y\). The subschema of the result subdatabase is defined as the subschema of \(X\). The instance of the result subdatabase is constituted by the instance of \(X\) to which is subtracted the instances of the objects of \(Y\). The difference operation is denoted \(X - Y\).

**Example:**
Examples of union, intersection and difference operations are shown in fig. 14.

3.5. Join

Unlike the relational data model, object-oriented models many relationships between objects can be represented within the objects themselves. Hence the join operations are used less frequently in the object algebra than in the relational algebra. However, the existing objects in a database may not explicitly reflect all relationships required by the queries. An explicit join is still needed to handle the cases when the relationship being queried upon is not defined within the object classes. Such relationship is called **value-based relationship** in contrast with those specified explicitly in the object model. For example, suppose we have a **layer document class** and a **technical document class**. A query «Find the layer document which has the same key words as the technical document», requires a value-based join between the classes **layer document** and **technical document** through the attribute key-words.

The join condition links the properties of only one class of \(X\) and the properties of only one class in \(Y\). The subschema of the result subdatabase is defined as the concatenation of the subschemas of the subdatabases operands through a new class \(c\). This new class is defined as a set of tuples. The tuple is made of two attributes, the first attribute is a set of references to the class \(c_1\) of \(X\), the second attribute is a set of references to a class \(c_2\) of \(Y\). The graphical structure of class \(c\) is shown in fig. 15. The instance of the result subdatabase is constituted of a portion of \(X\) that satisfies the join condition, a portion of \(Y\) that satisfies the join condition, and the union of the instances of

![Fig. 13. An example of associate operation (Z := X \(\cap\) Y)](image)

![Fig. 14. Examples of union, intersection and difference operations](image)
operands which are extended to the subschema of the result subdatabase. The Join operation is denoted by $X \bowtie [P] Y$ where $X$ and $Y$ are the subdatabases operands and $P$ represents the join predicate.

Example:

Let $X$ and $Y$ be the subdatabases represented by the fig. 16. An example of join operation is:

$X \bowtie [\text{Chapter.datecreation} = \text{Part.datecreation}] Y$

if, we suppose that the values of the attributes $\text{datecreation}$ of the $\text{referenceguide}$ class and of the $\text{userguide}$ class are:

- $c_i$,$\text{creationdate} = 26-03-62$, $c_j$,$\text{creationdate} = 26-03-62$
- $p_i$,$\text{creationdate} = 05-07-62$, $p_j$,$\text{creationdate} = 19-03-62$
- $p_j$,$\text{creationdate} = 26-03-62$, $p_i$,$\text{creationdate} = 26-03-62$
- $p_j$,$\text{creationdate} = 03-07-62$, $p_i$,$\text{creationdate} = 03-07-62$

The result of the join operation is shown in fig. 16. The subschema of the result subdatabase is composed of the subschema of $X$ and of $Y$, and of a new $\text{Chapter-Part}$ class which refer to $\text{Chapter}$ class and $\text{Part}$ class. The $\text{Chapter-Part}$ class is composed of two objects $f_i$ and $f_j$. The object $f_i$ is linked to the objects $\{c_i, c_j\}$ of the $\text{Chapter}$ class and it is also linked to the objects $\{p_i, p_j\}$ because the two sets of objects have the same value of the $\text{datecreation}$ attribute and the join condition is fulfilled. For the same reason, the object $f_j$ is linked to the objects $\{p_i, p_j\}$. However, the object $c_i$ of $\text{Chapter}$ class is not present in the result, because it doesn’t fulfill the join condition. Consequently, the objects $\{c, c_j\}$ of the $\text{Selection}$ class are also omitted from the result.

3.6 Grouping ($\gamma$)

The grouping is an unary operator. This operator is also used in NST-Algebra [Gut09]. It is used to transform an association relationship to an aggregation relationship. The grouping operation is defined as $\gamma(X)\{e_1, e_2, a\}$ where $X$ is an operand subdatabase and $\{e_1, e_2, a\}$ is the association relationship to be transformed to an aggregation relationship.

Example:

An example of a grouping operation is shown in fig. 17.

3.7 Split ($\phi$)

The split operation is the inverse of the grouping operation. Indeed, the split operation is used to transform an aggregation relationship to an association relationship. The split operation is defined as $\phi(X)\{e_1, e_2, a\}$ where $X$ is an operand subdatabase and $\{e_1, e_2, a\}$ is the aggregation relationship to be transformed to an association relationship.

Example:

An example of the split operation is shown in fig. 18.
3.8 Distribution (δ)

The distribution operator is used to restructure a subdatabase. This restructuring is made by distributing a property of class c₁ to a class c₂. The distribution is valid, if there exists a link between c₁ and c₂. The distribution operation is defined as δ(X) = (c₁, c₂, p) where X is a subdatabase operand, p is the property of c₁ to be distributed to class c₂.

Example:

An example of the distribution operation is shown in figure 19.

Fig. 19 Example of the distribution operation

4. Examples of expressions of algebraic queries

A storage and a query system of structured documents using the OODIMS technology was proposed in [Chi94]. This system was developed in the Verso project of INRIA. It is based on an extension of the model and query language of O2 OODIMS [Men90]. In order to request documents by their structure and content, O2SQL was extended to:
- take into account new functions of modeling like the ordered n-tuples and unions of types
- include predicates of textual retrieval like contains.

We will present examples of queries to compare our algebraic approach with the O2SQL query language.

Example 1:

«Find the documents containing the word Computer in the title and which are constituted of two chapters.»

The request is written as follows in the O2SQL query language:

Select x
From y in Document
Where x.title contains « Computer » and count (x.chapter) = 2

The algebraic expressions of the request are:

R₁ := \[Document, Chapter\] (Document.title contains « computer »)
R₂ := \[R₁\] (Document.chapter = 2)
R₃ := \[R₂\] (Document)
R₄ := \[R₃\] (Document.chapter = 2)
R₅ := \[R₄\] (Document)

Rapport interne LIPN 1999

This example represents an interrogation on the content of the document as well as its structure. The formulation of such a query under an algebraic form allows to choose uniformly the specification of the condition on the content and on the structure. However, the interrogation, using O2SQL, can only be made by using key words such as: Count.

Example 2:

«Find the documents that are only constituted of two chapters with the following condition: each chapter belonging to three documents.»

The algebraic expression of the query is:

R₁ := \[Document, Chapter\] (Document.chapter = 2)
R₂ := \[R₁\] (Document.chapter = 3)
R₃ := \[R₂\] (Document)
R₄ := \[R₃\] (Document)

To our knowledge, this query cannot be expressed in O2SQL. Indeed, such a query expresses a condition on the structure starting from the document to chapters and inversely, from chapters to the document.

5. Conclusion

We have presented in this paper an algebra for structured documents in an object-oriented approach. We have defined the concept of subdatabase on which is based the definition of algebraic operators. Indeed, the operands and the query result are subdatabases, as a consequence, the closing condition and the operators orthogonality conditions are respected because the operands and the results of the query operators are structured in the same way. The operators can be regrouped into two categories. A first category that deals with information retrieval operators (select, project, associate, union, intersection, difference and join operators). A second category that deals with information restructuring operators (grouping, split, distribution operators).

Appendix

The formal definition of the algebraic operators are given below:

- Select

Let us consider a subdatabase \(X = (\psi, X, O₂, A₂)\) (\(\psi = (C₂, p₂, λ₂)\)). The select operation is defined as:

\[T = \delta X | P = \left(\psi, X, O₂, A₂\right) \text{ where } \psi = (C₂, p₂, λ₂) \text{ such as:}
- \quad C₂ = C₁,
- \quad \Psi \subseteq \mathbb{E} C₂,
- \quad p₂ = p₁(c),
- \quad \psi(c; c₂) \subseteq C₂,
- \quad \psi(c; c₂) = \lambda₂(c; c₂)
- \quad \chi = \left\{ \frac{o \in O₂}{P(o) = \text{true}} \right\}
- \quad A₂ = \left(\chi(c; c₂) / (x; y) \subseteq A₂ : P(x; y) = \text{true and } P(y; x) = \text{true} \right\}

- Project

Let us consider a subdatabase \(X = (\psi, X, O₂, A₂)\) (\(\psi = (C₂, p₂, λ₂)\)). The project operation is defined as:

\[T = \left\{ \chi(c; c₂) / C₂ \subseteq C₂,
- \quad \psi(c; c₂) \subseteq C₂,
- \quad \lambda₂(c; c₂) \subseteq \lambda₂(c; c₂)
- \quad \chi = \left\{ \frac{o \in O₂}{T(o) \subseteq C₂} \right\}
- \quad A₂ = \left(\chi(c; c₂) / (x; y) \subseteq A₂ : T(x) \subseteq P(y) \subseteq \text{true} \right\}

- Associate
Let us consider a subdatabase $X=(\psi_X, O_X, A_X)$ and $Y=(\psi_Y, O_Y, A_Y)$ and a database $D=(\mathbb{Z}, A)$. The association operation is defined as:

$Z = X \times Y = \{(x,y) | x \in X, y \in Y\}$ such as:

- $C_Z = C_X \cup C_Y$
- $\psi_Z = \psi_X \times \psi_Y$
- $O_Z = O_X \cup O_Y$
- $A_Z = A_X \cup A_Y$

**Union**

Let us consider a subdatabase $X=(\psi_X, O_X, A_X)$ and $Y=(\psi_Y, O_Y, A_Y)$, then

$Z = X + Y = (\psi_Z, O_Z, A_Z)$ such as:

- $C_Z = C_X \cup C_Y$
- $\psi_Z = \psi_X \times \psi_Y$
- $O_Z = O_X \cup O_Y$
- $A_Z = A_X \cup A_Y$

**Intersection**

Let us consider a subdatabase $X=(\psi_X, O_X, A_X)$ and $Y=(\psi_Y, O_Y, A_Y)$, then

$Z = X \cap Y = (\psi_Z, O_Z, A_Z)$ such as:

- $C_Z = C_X \cap C_Y$
- $\psi_Z = \psi_X \cap \psi_Y$
- $O_Z = O_X \cap O_Y$
- $A_Z = A_X \cap A_Y$

**Difference**

Let us consider a subdatabase $X=(\psi_X, O_X, A_X)$ and $Y=(\psi_Y, O_Y, A_Y)$, then

$Z = X - Y = (\psi_Z, O_Z, A_Z)$ such as:

- $C_Z = C_X - C_Y$
- $\psi_Z = \psi_X - \psi_Y$
- $O_Z = O_X - O_Y$
- $A_Z = A_X - A_Y$

**Join**

Let us consider a subdatabase $X=(\psi_X, O_X, A_X)$ and $Y=(\psi_Y, O_Y, A_Y)$, then

$Z = X \bowtie Y = (\psi_Z, O_Z, A_Z)$ such as:

- $C_Z = C_X \cup C_Y$
- $\psi_Z = \psi_X \times \psi_Y$
- $O_Z = O_X \cup O_Y$
- $A_Z = A_X \cup A_Y$

$\textbf{References}$
Rapport interne LIPN 1999


