1

CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

54

0-315-12027-4

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Division

Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

59940

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

MICHAEL SPYRATOS

Date of Birth — Date de naissance

17-08-43

Country of Birth — Lieu de naissance

GREECE

Permanent Address — Résidence fixe

5-179, COBOURG STR.
OTTAWA, ONTARIO
K1N 8H4

Title of Thesis — Titre de la thèse

A MODEL - INDEPENDENT APPROACH TO DATA BASES

University — Université

CARLETON UNIVERSITY

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

Ph.D

Year this degree conferred — Année d'obtention de ce grade

1982

Name of Supervisor — Nom du directeur de thèse

Dr. S.A. MAHMOUD

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date

13-01-83

Signature

A MODEL-INDEPENDENT APPROACH TO DATA BASES

by

© MICHAEL SPYRATOS, DIPL. ING., M.A.Sc.

A thesis submitted to the Faculty of
Graduate Studies in partial fulfilment
of the requirements for the degree of

Doctorate of Philosophy

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

September, 1982

81

The undersigned recommend to the Faculty of Graduate
Studies and Research acceptance of the thesis

"A Model-Independent Approach to Data Bases"

submitted by Michael Spyrtos in partial fulfilment
of the requirements for the degree of Doctor of Philosophy.

S. A. Mahmoud
Thesis Supervisor

B. Pagurek
Chairman, Department of Systems
and Computer Engineering

ABSTRACT

This thesis describes a model-independent approach to data bases.

Data models are necessary for implementation of data bases as they provide a basis for

- (a) data representation and
- (b) definition of retrieval and update languages.

However, data base design is mostly concerned with problems related to information content and its possible transformations. Lossless decomposition, independence and normal forms are examples of such problems. Our thesis is that such problems can and must be studied independently of the way data is represented.

A first attempt at this direction is presented in [30], where the data base update is considered as a primitive concept, i.e., it is defined in terms of state transitions, regardless of data representation. Sufficient conditions are given for a set of updates to "define" a data base mapping (actually such a set defines a whole equivalence class of data base mappings). We use this result as a starting point in developing our approach.

First, we present formally the isomorphism between the semilattice of equivalence classes of data base mappings and the semilattice of partitions of the data base state space. Then we show that :

- (1) Every equivalence class of data base mappings can be represented

uniquely by a set of data base updates; this provides a model-independent definition of equivalence classes of data base mappings.

- (2) The algebra of equivalence classes of data base mappings can be transformed through an isomorphism, to an algebra of sets of data base updates [29].
- (3) The expressive power of the algebra of sets of updates is greater than that of the relational algebra.
- (4) The model-independent approach is extended to account for generalized mappings where we represent incomplete information by the induced state space covering.

ACKNOWLEDGEMENTS

I would like to gratefully acknowledge all of the people whose assistance and support proved invaluable to me in this thesis undertaking.

Special thanks are due to my thesis advisor, Dr. S.A. Mahmoud, whose guidance as well as friendship and advice supported me throughout this work.

Special thanks are due to Dr. N. Spyratos for many inspiring discussions throughout this work.

Thanks are also due to Dr. S. Abiteboul of INRIA, France for many fruitful discussions and suggestions.

I would also like to thank Dr. B.A. Bowen for his encouragement and friendship.

The financial assistance of a Transport Canada University Programs Fellowship is gratefully acknowledged.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	ix
CHAPTER 1 INTRODUCTION	1
1.1 Motivation	2
1.2 Survey of existing literature	6
1.3 Approach and major contributions	9
1.4 Thesis organization	13
CHAPTER 2 THE RELATIONAL MODEL	15
2.1 Introduction	16
2.2 Definitions and notations	18
2.3 Operations on relations	21
2.4 Data manipulation languages for the relational model	23
2.5 Integrity constraints	26
2.6 Transformations (or views) of a data base	28

CHAPTER 3 MODEL-INDEPENDENT APPROACH

THE EXPRESSIVE POWER OF SATURATED SETS OF UPDATES

36

3.1	Introduction	38
3.2	Definition of mappings through updates	40
3.3	Outline of the approach	47
3.4	The lattice $\Pi(S)$ of partitions of a data base state space S	49
3.5	A semilattice isomorphism between equivalence classes of data base mappings and partitions of the data base state space	51
3.6	The intersection semilattice I_S of saturated sets of updates of a data base with state space S	56
3.7	A semilattice isomorphism between partitions of a data base state space and saturated sets of updates	60
3.8	The fundamental isomorphism theorem for data base mappings	62
3.9	The expressive power of saturated sets of updates	63
3.10	The structure of the information content of views in terms of partitions of the state space	64
3.11	The structure of the information content of views in terms of saturated sets of updates	71
3.12	Data base decompositions preserving the information content (Lossless decompositions)	74
3.13	Lossless - join decompositions of the relational model and lossless decompositions	81
3.14	Definition of integrity constraints as a saturated set of updates	90
3.15	Characterization of the family of integrity constraints	92
3.16	Conclusion	95

CHAPTER 4	GENERALIZED DATA BASE MAPPINGS	97
4.1	Introduction	98
4.2	Knowledge representation	100
4.3	The lattice of maximal coverings of a data base state space	111
4.4	Interpretation of the structure of coverings in the knowledge context	126
4.5	View complements	130
4.6	Incomplete information	137
4.6.1	Incomplete information under the "closed world assumption". Protection of information.	138
4.6.2	Incomplete information under the "open world assumption"	143
4.7	Independence	145
4.8	Conclusion	155
CHAPTER 5	CONCLUDING DISCUSSION	156
5.1	Summary and conclusions	157
5.2	Suggestions for further research	160
REFERENCES		162

LIST OF FIGURES

Figure no		Page
2.1	A data base query interpreted as a view	32
2.2	A data base update interpreted as a view	34
3.1	The definition of a data base view	41
3.2	A lossy and a lossless decomposition of a data base	82
3.3	Pictorial representation of integrity constraints	94
4.1	Three data base queries and the partitions they induce	108
4.2	A " traditional " view π_{01} and a " generalized " view σ_{31}	112
4.3	The relation $\leq$ on $C(S)$ is not antisymmetric	114
4.4	The lattices of partitions and maximal coverings of S	125
4.5	Addition of knowledge $\pi_{01} \cdot \pi_{02}$ and addition of ignorance $\pi_{01} \oplus \pi_{02}$	129
4.6	Example of a supremum maximal covering τ complementary to a view σ	135
4.7	Protection of information at secrecy level π^*	139
4.8	Counterexample : $\sigma_{21} = \sigma_{12} = 1$, does not imply independence in $MC(S)$	148
4.9	Counterexample : Not all maximal coverings are reduced ones $RC(S) \subsetneq MC(S)$	152

CHAPTER 1

INTRODUCTION

The problem considered in the thesis and the main results obtained are summarized and discussed. An outline of the thesis is presented.

1.1 Motivation

A data base can be thought of as information about the real world facts. This information is physically stored, for example, in a computer system, and it is accessed by users who either retrieve it or update it (when the represented real world facts change).

For example, we could imagine a data base concerning the employees of a company, their salaries and their work location. Such a data base would be consulted by the payment office to obtain the list of employees working at a specific location, or would be updated by the personnel office after the hiring of a new employee.

We can think therefore of a data base as a variable, denoted $BASE$, whose value at every instant represents - according to a set of conventions - some real world facts.

Consulting the data base, can be thought of as "viewing" the whole or part of the current value of $BASE$.

Updating the data base can be seen as changing the current value of $BASE$.

The formal definition of the variable $BASE$ and of the operations allowed on its values - such as queries and updates - is done by means of a basic formal structure, such as, the tree,

the graph, the relation, etc.

The choice of basic structure determines what is known in the literature as a data model. It is the set of conventions according to which data is represented and operations on data are defined.

The following major data models have emerged so far, in chronological order :

The hierarchical model [31] ,

The network model [2] ,

The relational model [15] .

The basic formal structures of these models are : the tree, the graph and the relation respectively.

Data base implementation relies heavily on the data model, as it provides the means for

- (1) data representation
- (2) formal definition of retrieval and update languages.

For example, the relational model allows for a simple tabular representation of data. On the other hand, the associated relational algebra provides a basis for the definition of a number of languages, e.g., SEQUEL.

Data base design is concerned mostly with the organization of data and as such has very little to do with the data model. Let us consider an example. Suppose that we are designing a data base concerning the suppliers of a company, namely, their names, addresses, items supplied and prices. At the design stage, the main question is how to organize this information. Let us consider first a single file called SUPPLIERS and containing all the information :

SUPPLIERS (SNAME, SADDRESS, ITEM, PRICE) .

We can see several problems with this organization.

- (a) Redundancy : ~~The address of a supplier is repeated once~~ for each additional item supplied.
- (b) Potential inconsistency (update anomalies) : as a consequence of the redundancy, we could update the address of a supplier in one record while leaving it fixed in another.
- (c) Insertion anomalies : we cannot record an address for a supplier if that supplier does not currently supply at least one item.
- (d) Deletion anomalies : should we delete all the items supplied by one supplier, we unintentionally lose track of his address.

In this example , we can overcome all the above problems if we

replace the file SUPPLIERS by the following two smaller files:

SA (SNAME, ADDRESS)

SIP (SNAME, ITEM, PRICE).

The first, SA, gives the address for each supplier exactly once. Thus there is no redundancy. Moreover, we can enter an address for a supplier even if he currently supplies no items.

The second, SIP, gives the suppliers, the items they supply and the price each supplier charges for each item. Yet some questions remain. For example, there is a disadvantage in the two-file organization: to find the address of suppliers that supply a given item, we must "merge" or "join" two files, which is expensive, whereas with the single file organization this would be very simple.

As the preceding example shows, data base design is concerned mostly with problems related to information content and its possible transformations. These problems are there, regardless of the model used for data representation. So why study these problems in the context of a specific data model? This would only obscure their understanding and hinder their clear and general statements. It is in this perspective that we propose our model-independent approach.

1.2 Survey of existing literature

The field of data bases has received great attention over the past decade. A vast majority of the theoretical work in this area is based on Codd's relational model [15], which is very general, yet mathematically tractable. For the purposes of our discussion, we recall here its basic features. A full description of the model is given in the following chapter.

According to the relational model, a data base is a set of time varying relations. Each relation is specified through a relation schema. For example:

SUPPLIERS (SNAME, SADDRESS, ITEM, PRICE)

is a relation schema. The names that appear between parentheses are the attributes and take values on specified domains, that are usually understood from the context (e.g., PRICE takes as values integers). The name SUPPLIERS is the relation variable and takes as values finite sets of tuples. A data base schema or simply a schema is a set of relation schemas. An instance or a state of a data base schema is an assignment of values to its relation variables. Some instances may be " illegal " as they do not satisfy certain properties of the real world facts that the data base is supposed to represent. These properties are the

data base integrity constraints. An example of a schema that we have seen in the previous section is the following:

SA (SNAME, SADDRESS)

SIP (SNAME, ITEM, PRICE)

The fact that each item must appear with a unique price is an example of an integrity constraint. Integrity constraints capture some of the semantics of the information content in a data base. However, it is generally admitted that the relational model, despite its rigor and generality, is rather poor in expressing semantics. The semantics of a data base mapping in its various forms (query, update, decomposition etc.) have been studied extensively. However, most of that work has studied the mapping in the relational framework, using the algebraic approach. That is, a mapping is considered simply as a fixed sequence of relational operations that when applied to the current instance of the schema, produce an instance of another schema.

Information-theoretic considerations, so obviously relevant in the study of data base design problems, were brought out very infrequently and implicitly, e.g., in the work of Rissanen [27].

Recent work [30] has examined the data base mapping from a fundamental point of view, in terms of the partition it induces on the set of legal data base instances. The refinement partial

order on partitions provides a natural hierarchy of information content; however, the mathematical notion of a partition has no real status in a data base context. This problem is resolved by defining a partition and therefore a mapping, via data base updates. The importance of this result lies in the fact that the data base update can be considered as a primitive concept, regardless of data representation. This is the first step, leading to model independence. It should be possible, by exploring the structure of data base updates, to build a model-independent theory for data bases. This is the point of view that we have adopted in our approach.

1.3 Approach and major contributions

The data base mapping is an all embracing concept in data base theory. Therefore, any attempt to build a model-independent theory of data bases must begin with a model-independent definition of a mapping. As was explained in the previous section, a prime candidate for such a definition is the data base update; in [30], sufficient conditions are given for a set of data base updates to define a data base mapping; thus, for an arbitrary equivalence class of mappings, there is a family of sets of updates, each of which defines the equivalence class of mappings.

In order to create a model-independent theory of data bases, the following problems must be resolved:

- (i) Every equivalence class of data base mappings must be represented uniquely by a set of data base updates.
- (ii) There is an algebra of equivalence classes of data base mappings that allows comparison of mappings with respect to their information content and "addition" of the information content of two mappings. This algebra must be transformed into an algebra on the sets of updates that correspond to equivalence classes of data base mappings.

In other words, for each equivalence class of data base mappings, we must so choose a unique definition set of data base updates that the family of all these unique definition sets possesses an algebra isomorphic to the algebra of equivalence classes of data base mappings.

We resolve the first problem as follows:

An equivalence class of data base mappings defines a partition on the set of legal data base instances. We associate to each equivalence class of data base mappings, the set of all data base updates that leave the corresponding partition invariant. Such a set, we term a "saturated" set of updates. As we shall explain in CHAPTER 3, the updates of a saturated set have a very neat characterization: they are precisely those updates that are translated as identities in the context of the view defined by the mapping.

We resolve the second problem as follows:

We show that the above association of sets of updates to equivalence classes of data base mappings is a semilattice isomorphism. This isomorphism transforms the algebra of equivalence classes of mappings into an algebra of sets of updates, whereby the rather complicated operations on mappings are transformed into simple set-theoretic operations on sets of updates.

This non-trivial result is the first major contribution of the thesis.

Some consequences of this result are as follows:

The information content hierarchy of mappings, represented by the refinement partial order on the data base state space partitions is converted isomorphically into the set-theoretic inclusion hierarchy on the family of saturated sets of updates.

It should be noted that the hierarchy of sets of updates

is more natural than the hierarchy of partitions, because in practice, state space partitions appear only implicitly, underlying the data base processes; this is in contrast to data base updates that constitute explicitly a fundamental notion in the data base processes. " Addition " and comparison of the information contents of two data base mappings are represented respectively by intersection and set-theoretic inclusion on the corresponding saturated sets of updates.

Through this representation, we obtain a simple characterization of lossless decompositions and we make explicit the relationship that exists between updates and integrity constraints.

Also, as a consequence we obtain another interesting result: namely, that if we use the updates as primitive concepts, then we obtain an expressive power strictly higher than that of the relational algebra in the following sense:

Every relational expression (i.e., a mapping) can be described by a set of updates, while there are sets of updates that do not correspond to mappings.

Subsequently, we generalize our approach in order to include incomplete information. Incomplete information can be seen at two levels. First, according to the so-called " open world assumption " [25], the information stored in a data base is merely that which we know about the real world. So at any particular moment the data base information is simply a " subset " of the real world information. Second, supposing that the data base

contains complete information about the real world, we may want to "hide" part of that information from a particular user. What the user obtains then is incomplete information about the data base. Of course, partitions are no longer sufficient to describe incomplete information, as the information carrier is no longer a data base mapping but a "generalized mapping", i.e., a binary relation on two state spaces. We overcome this difficulty by using coverings instead of partitions. We introduce formal notions of relative information capacity and we obtain several interesting results concerning these notions. An algebra of coverings is introduced that allows us to "add" both knowledge and ignorance of the data base, contained in two coverings. We apply our theory to the notions of view complement [7], incomplete information [25], and independence [27, 12, 6]. An interesting result is that our intuitive notion of independence is not always verified when arbitrary coverings are used for knowledge representation. To our knowledge, this treatment of incomplete information through coverings is a novel approach and constitutes the second major contribution of the thesis.

1.4 Thesis organization

This thesis is organized into five chapters as follows :

CHAPTER 2 is a concise presentation of the relational model. It includes the definitions of the basic concepts, namely, relation schema and data base schema, query, update, integrity constraints, mapping, etc.

CHAPTER 3 presents our model-independent approach. It begins with a critical review of the work in [30]. Then it proceeds to the proof of the fundamental isomorphism theorem for data base mappings that transforms the algebra of equivalence classes of mappings into the algebra of saturated sets of updates. It also includes a discussion of the consequences of this theorem. In model-independent terms, it establishes:

- (1) The structure of view information content
- (2) A criterion for lossless decompositions
- (3) The definition of integrity constraints as saturated sets of updates

CHAPTER 4 is essentially a treatment of incomplete information using coverings. It begins with a discussion on knowledge representation using coverings and then introduces and studies an algebra of coverings. This algebra is subsequently applied to the study of view complements,

incomplete information and independence.

CHAPTER 5 presents the conclusions and recommendations for further research.

CHAPTER 2

THE RELATIONAL MODEL

The basic concepts of the relational model are briefly discussed to serve as a reference for the model-independent approach to data bases.

2.1 Introduction

A data base can be thought of as information about the real world facts. This information is physically stored, for example, in a computer system, and is accessed by users who either retrieve or update it (when the represented real world facts change).

For example, we could imagine a data base concerning the employees of a company, their salaries and their work location. Such a data base would be consulted by the payment office to obtain the list of employees working at a specific location, or would be updated by the personnel office after the hiring of a new employee..

We can think therefore of a data base as a variable, denoted `BASE` , whose value at every instant represents, according to a set of conventions, some real world facts..

Consulting the data base can be thought of as " viewing " the whole or part of the current value of `BASE` .

Updating the data base can be seen as changing the current value of `BASE` .

The formal definition of the variable `BASE` and of the operations allowed on its values, such as queries and updates, is done by means of a basic formal structure, such as.. the tree,

the graph, the relation, etc.

The choice of basic structure determines what is known in the literature as a data model, i.e., the set of conventions according to which data is represented and operations on data are defined.

The following major data models have emerged so far, in chronological order :

The hierarchical model [31] ,

The network model [2] ,

The relational model [15] .

The basic formal structures of these models are : the tree, the graph and the relation respectively.

Data base design is concerned mostly with the organization of data, and as such, has very little to do with the data model.

Data base implementation, however, relies heavily on the data model, as it provides the means for

- (a) data representation
- (b) formal definition of retrieval and update languages.

In this chapter we present a concise description of the relational model.

2.2 Definitions and notations

Let $A = \{ A_1, A_2, \dots, A_n \}$ be a set of names called attributes.

A set T_i of values is associated with each attribute A_i , $i = 1, 2, \dots, n$. The set T_i is called the type or the domain of A_i .

A tuple on A is a mapping that associates to each attribute A_i in A a value in T_i .

A relation on A is a finite set of tuples on A , i.e.,

a relation is a finite subset of $\prod_{i=1}^n T_i$.

A relation variable on A is a variable whose values are relations on A .

The set of all possible values of a relation variable is called the type of the relation variable.

The following is a PASCAL definition of a relation variable V (we assume here that $T_1, T_2, \dots, T_n$ are previously defined types).

```
type TUPLE = record A1:T1; A2:T2; ...; An:Tn end
type RELATION = set of TUPLE
var V : RELATION
```

Informally, we use the notation $V (A_1:T_1; A_2:T_2; \dots; A_n:T_n)$, known as a relation schema, to denote that V is a relation variable whose values are relations on A .

The following is an example of a relation schema:

PAY (EMPLOYEE : NAME ; SALARY : INTEGER) ,

where NAME and INTEGER are appropriately defined types.

Assuming that the attribute types are clearly understood from the context, we can also write:

PAY (EMPLOYEE, SALARY)

The last notation is very commonly used. Thus if we write:

LOCATION (EMPLOYEE, DEPARTMENT) ,

we have assumed tacitly some appropriate types for EMPLOYEE and for DEPARTMENT.

A (data) base variable is a (ordered) set of relational variables that we write between parentheses. Of course, if the base variable consists of just one relation variable, then we omit the parentheses.

As an example, let us define a base variable R using the

relation variables PAY and LOCATION, defined earlier:

$R = (\text{PAY}, \text{LOCATION})$.

Any assignment of values to the relation variables (from their corresponding types), is a value of the base variable.

The set of all possible values of a base variable is called the type of the base variable.

A data base is a pair $B = (R, S)$, where R is a base variable, say of type T , and S is a subset of T .

In a data base $B = (R, S)$, S denotes the set of all values that the base variable R is allowed to take in the context of B and is called the value set of B or the state space of B .

2.3 Operations on relations

In this section we define some of the operations on relations. We will use the symbol of a relation variable to indicate also its current value. For example, suppose that the current value of the relation variable PAY contains the following tuples:

$$t_1 = \{ (\text{EMPLOYEE}, \text{JOHN}), (\text{SALARY}, 1100) \}$$

$$t_2 = \{ (\text{EMPLOYEE}, \text{PETER}), (\text{SALARY}, 1500) \}$$

Then we can write: $\text{PAY} = \{ t_1, t_2 \}$, and we can say "the relation PAY" to mean "the current value of PAY".

We use any of the symbols $t.X$, $t[X]$ or t/X to denote the restriction of a tuple t to a subset X of its attribute set.

Referring to the previous example, we have:

$$t_1.\{ \text{EMPLOYEE} \} = \{ (\text{EMPLOYEE}, \text{JOHN}) \}$$

Let now V and W be two relations with attribute sets A and B respectively.

The projection of V on the subset X of A , denoted $V[X]$, is a relation on X defined as follows:

$$V[X] = \{ t.X \mid t \in V \}$$

The (natural) join of the relations V and W , denoted $V * W$, is a relation on $A \cup B$, which is defined as follows :

$$V * W = \{ \text{tuple } p \text{ on } A \cup B \mid p.A \in V \text{ and } p.B \in W \}$$

Let C be a condition (predicate) on the set of tuples on A . The selection of V with respect to C , denoted $V|C$, is a relation on A defined as follows :

$$V|C = \{ t \in V \mid C(t) = \text{true} \}$$

2.4 Data manipulation languages for the relational model

Retrieval and updating of the information contained in a data base is expressed in a data manipulation language (or "query language.")

In a relational data manipulation language or query language the notation for expressing queries is usually the most significant part. The non-query aspects of such a language are often straightforward, being concerned with the insertion, deletion and modification of tuples. On the other hand, queries, which in the most general case are arbitrary functions applied to relations, often need a rich, high level language for their expression.

There are three principal approaches to the design of query languages for the relational model, depending on whether the primitive object is the relation, the tuple or the element of the domain of some attribute.

There are two broad classes of relational query languages, characterized by the way the relations required by the query are formed from the primitive objects, i.e., via operations or via a predicate on the objects.

So we have:

- (1) Algebraic languages , where a relation of a query is expressed by applying specialized operations to relations

- (2) Predicate calculus languages where a relation of a query is expressed as a set of tuples that satisfy a predicate.

There are two classes of predicate languages :

- (a) the tuple (predicate) relational calculus and
- (b) the domain (predicate) relational calculus,

depending on whether the primitive objects are tuples or elements of the domain of some attribute.

So we have three distinct kinds of query language :

- (i) The relational algebra (an algebraic language)
- (ii) The tuple relational calculus (a predicate calculus language)
- (iii) The domain relational calculus (a predicate calculus language)

There are five basic operations on relations that serve to define the relational algebra , namely, union, set difference, cartesian product, projection and selection.

Let us note that the three abstract relational query languages mentioned above are not implemented exactly as stated in any existing Data Base Management System (DBMS) ; however,

they serve as a benchmark for evaluating existing systems; that is, each of the three abstract query languages is equivalent in expressive power to the others [17, 5]. These languages were proposed by Codd to represent the minimum capability of any reasonable query language using the relational model.

Real query languages usually provide the capabilities of the abstract languages and additional capabilities as well.

Typical examples of relational query languages are the following:

ISBL, an algebraic language,

QUEL, a tuple predicate calculus language,

Query - by - Example, a domain predicate calculus language .

Finally, let us mention two hybrid relational query languages, between algebra and predicate calculus :

SQUARE and SEQUEL.

2.5 Integrity constraints

The semantics of real world facts represented in a data base give rise to integrity constraints. These constraints are properties that the data base values must satisfy in order to represent reality faithfully.

Consider, for example, the base variable

$R = (\text{PAY}, \text{LOCATION})$,

defined earlier, and suppose its type is T . The fact that

" every employee that receives a salary must work in some department " gives rise to the following integrity constraint :

$C : \text{PAY}[\{ \text{EMPLOYEE} \}] = \text{LOCATION}[\{ \text{EMPLOYEE} \}]$

Clearly, if we want to define a data base $B = (R, S)$, which satisfies the above constraint C , then we must define its value set (its data base state space) S as follows :

$S = \{ R \in T \mid C(R) = \text{true} \}$

Clearly, integrity constraints can be seen as predicates on the type of the base variable.

Various kinds of integrity constraints have been proposed in the literature [16, 3, 4, 22, 26, 33]. Among the

earliest to appear is the functional*dependency [16, 3, 4] , which is defined as follows.

Let V be a relation variable on a set A of attributes, and let X and Y be subsets of A . We say that

" X functionally determines Y in the relation V ", denoted

$$\underline{X \rightarrow Y \text{ in } V} ,$$

iff $\forall t, p \in V, \quad t.X = p.X \quad \text{implies} \quad t.Y = p.Y .$

As an example, consider again the base variable

$$R = (\text{PAY}, \text{LOCATION}) .$$

The functional dependency,

$$C' : \{ \text{EMPLOYEE} \} \rightarrow \{ \text{DEPARTMENT} \} \text{ in LOCATION}$$

expresses the fact that :

" every employee must be assigned to one and only one department " .

2.6 Transformations (or views) of a data base

A fundamental concept of data base theory is the one of a data base transformation or view.

Suppose we are given a data base $B = (R, S)$, where the base variable R is of type T .

A data base mapping (sometimes, simply, " a mapping ") is a mapping on S .

A transformation (or a view) of a data base B is a couple $v_1 = (R_1, f_1)$, where R_1 is a base variable, say of type T_1 , and f_1 is a mapping from S into T_1 .

The data base mapping f_1 will be referred to as the view definition mapping.

The image of B under v_1 is the data base $B_1 = (R_1, S_1)$, where $S_1 = f_1(S)$.

Table 2.1 shows an example of a view definition. Note in particular how the value set S_1 is defined : the mapping f_1 is applied to the current value of R and the result is assigned as the current value of R_1 .

The concept of a view was originally introduced for a specific purpose : to give the users the possibility of working on a " subset " of the data base information relevant to a specific application. This is clearly reflected in the terms.

Relation schemas

D (DEPARTMENT, MANAGER)

E (EMPLOYEE, DEPARTMENT)

M (EMPLOYEE, MANAGER)

Integrity constraints $C_1 : \{ \text{DEPARTMENT} \} \rightarrow \{ \text{MANAGER} \} \text{ in } D$ $C_2 : \{ \text{EMPLOYEE} \} \rightarrow \{ \text{DEPARTMENT} \} \text{ in } E$ $C = C_1 \wedge C_2$ Base variables $R = (D, E)$, say of type T $R_1 = M$, say of type T_1 Data base $B = (R, S)$, where $S = \{ R \in T \mid C(R) = \text{true} \}$ Transformation or view $v_1 = (R_1, f_1)$, where $f_1 : S \rightarrow T_1$ is
such that $\forall R \in S$, $f_1(R) = f_1((D, E)) =$
 $= (D * E) [\{ \text{EMPLOYEE}, \text{MANAGER} \}]$ Assignment $R_1 : = f_1(R)$ Image data base $B_1 = (R_1, S_1)$, where $S_1 = f_1(S)$ Table 2.1

Example of a data base transformation (view).

user subschema and external schema [2] that were originally used for a view. Most recent data base systems provide a view definition facility [13], but restrict view updating, due to some difficult update translation problems [7].

The concept of a data base transformation (view) - as defined here - emerges in almost every aspect of data base theory. Among the most important data base transformations are the query and the update (It should be noted that the definition mapping of a view that represents a data base update is necessarily an operation on the data base state space S).

In the following two examples, we see how a query and an update of a data base can be interpreted as data base views:

Example 2.1

A query interpreted as a data base view

Consider the data base $B = (R, S)$ defined in Table 2.1 and the query

q : "list all employees and their managers"

The view $v_1 = (R_1, f_1)$, defined also in Table 2.1, is the interpretation of the query q as follows:

The mapping f_1 tells us how to process the base variable R

in order to obtain the query answer, and the variable R_1 holds the result of the processing.

The processing of this query requires a join of the two relations D and E , and then a projection on the following set of attributes : $\{ \text{EMPLOYEE, MANAGER} \}$.

Let us call this processing f_1 , i.e.,

processing f_1 : " join the relations D and E and project
on the attribute set
 $\{ \text{EMPLOYEE, MANAGER} \}$ " .

The processing f_1 applied to the current value of R yields the answer $f_1(R)$ to the query.

As an example, a current value of R , and the corresponding answer to the query q (or the corresponding current value of the view v_1) i.e., $R_1 = f_1(R)$, are shown in Figure 2.1

Example 2.2

An update interpreted as a data base view

A similar argument applies in the case of updates.

Consider again the data base $B = (R, S)$ defined in Table 2.1 and the following data base update :

D		E	
DEPARTMENT	MANAGER	EMPLOYEE	DEPARTMENT
1	x	a	1
2	y	b	1
		c	2

Current data base value R

M	
EMPLOYEE	MANAGER
a	x
b	x
c	y

Query answer $R_1 = f_1(R)$

Figure 2.1 A data base query interpreted as a view.

update : " If the department 1 appears in the current value of R then insert the employee d in the department 1 " .

This update defines a mapping u from S into S and therefore, it can be interpreted as a view $w = (R, u)$, as follows :

The mapping u tells us how to process the data base variable R , and the result of the processing is returned to R .

We have the following definition of the processing u :

processing u : " Insert the employee d in the department 1, only if the department 1 appears in the current value of R " .

The processing u applied to the current value of R yields the updated value $R_1 = u(R)$, of the data base .

As an example, a current value of R , and the corresponding updated data base value (the corresponding current value of the view w), i.e., the relation $R_1 = u(R)$, are shown in Figure 2.2.

In concluding this section, let us note that a data base abstraction can also be seen as a view, whereas a data base

D		E	
DEPARTMENT	MANAGER	EMPLOYEE	DEPARTMENT
1	x	a	1
2	y	b	1
		c	2

Current data base value R

D		E	
DEPARTMENT	MANAGER	EMPLOYEE	DEPARTMENT
1	x	a	1
2	y	b	1
		c	2
		d	1

Updated data base value $u(R)$

Figure 2.2 A data base update interpreted as a view.

decomposition can be seen as a view or as a set of views. Finally, even integrity constraints can be seen as data base views [32]. As a result, views are involved explicitly or implicitly in a number of important data base areas. Most prominent among them are the following:

Equivalence of data bases [8, 9, 10] ,
Independent decompositions [27, 12, 6] ,
Normal forms [16, 8, 11, 19, 23] ,
Translation of view updates [7, 18] ,
Integrity constraints [16, 3, 4, 22, 26, 33, 32]

The importance of transformations in the data base processes provides the motivation for our studying, in the next chapter, the structure of the information content of data base mappings, in a fundamental, model - independent approach, using only data base updates.

CHAPTER 3

MODEL-INDEPENDENT APPROACHTHE EXPRESSIVE POWER OF SATURATED SETS OF UPDATES

In a recent work [30], conditions are given for a set of updates to "define" a given data base mapping.

In our approach, we choose for each mapping the largest definition set, which we term "a saturated set of updates", and we show that there exists a double semilattice isomorphism between :

- (1) The set of equivalence classes of data base mappings
(two mappings being equivalent iff they induce the same partition on the state space) ,
- (2) The set of partitions of the data base state space ,
- (3) The family of all saturated sets of updates .

Next we show that the information content of a data base mapping, which is expressed in [30] as a pair of partitions, can be completely characterized simply by the partition induced on the state space. Consequently the structure of information content of data base mappings is established to be the product semilattice of the state space partitions or, isomorphically, the intersection-semilattice of the saturated sets of updates. The latter is a natural model-independent representation, using only one basic concept, the data base update, and involving only set-theoretic inclusion and intersection of saturated sets of updates.

We use these results to characterize lossless decompositions and to make explicit the relationship that exists between integrity constraints and data base updates .

3.1 Introduction

Work on data base semantics has studied in various forms the data base mapping, a mapping from instances of a schema to instances of another schema - be it decomposition, query, update, or view definition. (In fact, recent work on algebraic dependencies [32], showed that even integrity constraints can be considered as data base mappings.)

Most of that work has studied data base mappings in the relational framework, using the algebraic approach. Information - theoretic considerations, so obviously relevant in the study of data bases, were brought out very infrequently and implicitly, e.g., in the work of Rissanen [27].

In a recent work [30], the view information content is expressed as a couple (π, π') of two partitions: the partition π , induced on the data base state space S by the view definition mapping and the partition π' , induced on a given set X of "applicable" updates by the mapping that translates applicable updates at the view level. (An update is applicable with respect to π if it maps each block of π into some block of π .) The partition π is considered to represent the static part, i.e., the data base values, whereas π' represents the dynamic part, i.e., the operations of the view. For any two views, for which the updates in X are applicable, the information content of the view (π_1, π'_1)

includes the one of the view (π_2, π'_2) iff π_1 refines π_2 and π'_1 refines π'_2 .

We will use these ideas as a starting point in developing our approach. In the following section we introduce some basic definitions and notations and we explain how a data base mapping can be associated to a partition of the data base state space and how a partition of a data base state space can be determined by a set of updates.

3.2 Definition of mappings through updates

Let us consider an example (Figure 3.1) of a data base. We suppose that the data base variable consists of a single relation variable R with relation schema :

$R (\text{EMPLOYEE} : \{ a, b \} ; \text{DEPARTMENT} : \{ x, y \})$

In other words, we have a single relation variable R over the attribute set : $\{ \text{EMPLOYEE}, \text{DEPARTMENT} \}$, and the domains of EMPLOYEE and DEPARTMENT are respectively, $\{ a, b \}$ and $\{ x, y \}$.

We assume that the integrity constraints of the data base consist of a single functional dependency :

$\{ \text{EMPLOYEE} \} \rightarrow \{ \text{DEPARTMENT} \}$

Then the data base state space S consists of the nine legal data base instances as shown in Figure 3.1 (in order to simplify the notation we have written : ax instead of $\{ (a, x) \}$, etc.) The instances in S are also numbered 1 through 9 , for later reference.

Suppose now that we want to define a data base view that contains only the employees. We proceed as follows :

Data base $B=(R,S)$ View $v=(r,f)$ π

A set $U' \subset U$ of allowable data base updates

- i : Change nothing (i.e., identity on S.)
- p : Change the department of a .
- q : Change the department of b .
- j : Change the department of both a and b .
- k : If the data base contains exactly one employee, then change his department .
- l : If a is not in the data base, then insert a in department x .
- m : If a is not in the data base, then insert a in department y .
- n : If b is not in the data base, then insert b in department x .
- t : If b is not in the data base, then insert b in department y .
- c : Delete the tuple of a .
- e : Delete the tuple of b .
- w : Close down department x .
- z : Close down department y .

Figure 3.1 The definition of a data base view.

- (1) Define a relation variable r with schema :

$$r (\text{EMPLOYEE} : \{ a, b \})$$

- (2) Let $P(\{ a, b \})$ denote the power set of the set $\{ a, b \}$. We define a mapping $f : S \rightarrow P(\{ a, b \})$, such that for every instance s in S , $f(s)$ is the projection of s on the attribute EMPLOYEE .
- (3) If s is the current instance of R then $f(s)$ is the current instance of r .

The view definition mapping f induces on S the following partition :

$$\pi = \{ \{ 1 \}, \{ 2, 3 \}, \{ 4, 5 \}, \{ 6, 7, 8, 9 \} \}$$

The blocks of π can serve as model - independent "names" for the view instances, so in our discussion we identify the set of view instances with π .

Let U denote the set of all data base updates. (A subset U' of U is shown in Figure 3.1). Clearly, each update in U is a mapping from S into S as it transforms each data base instance into another data base instance.

Consider the set $D = \{ p, q, j, k \} \subset U$.

Recalling that any update u in U is a set of pairs of data base instances, i.e.,

$$u = \{ (s, u(s)) / s \in S \},$$

it is easy to verify that the relation

$$u \sim_d d \in D$$

is an equivalence relation on S , whose equivalence classes are the blocks of π . Clearly, the set D of updates can determine the partition π .

In general, a set D of update operations on S is called a definition set of a partition π of the data base state space S , if the union of updates in D is an equivalence relation that induces π on S .

Clearly, if D is a definition set of π , then every u in D maps each block of π into itself. It should be noted that there may be several definition sets for a given π .

Definition 3.1 $\forall \pi \in \Pi(S)$, the "saturated set of updates with respect to π " is the set I_π of all updates on S that map each block of π into itself.

It is easy to see that for any partition π of the data base state space S , the saturated set of updates I_π is the largest definition set of π .

We denote by I_S the family of all saturated sets of updates on a state space S .

Referring again to the example in Figure 3.1, we observe that with respect to π , the updates in U fall into two categories :

(1) The updates which "make sense" in the context of the view, and which we call applicable on π . Formally, a data base update u in U is applicable on π iff

$\forall B \in \pi$, there exists $B' \in \pi$, such that $u(B) \subset B'$,

i.e., u is applicable iff it transforms each block of π to only one block of π ; for example, the updates $i, p, q, j, k, l, m, n, t, c, e$; in Figure 3.1, are all applicable on π .

We denote by $A(\pi)$ the set of all updates in U that are applicable on π .

(2) The updates that are not applicable on π , and therefore "make no sense" in the context of the view, for example, the updates w, z , in Figure 3.1, are not applicable on π .

Applicable data base updates can be "interpreted" or "translated" into view updates as follows: Let U_1 denote the set of all view updates, i.e., the set of all mappings from π into π . For any u in $A(\pi)$, define the translation of u on π , denoted $h_\pi(u)$, to be a mapping from π into π , such that for any $B \in \pi$,

$$h_\pi(u)(B) = B', \text{ where } B' \in \pi \text{ is such that } B \subset B'.$$

The above translation process defines a mapping h_π from $A(\pi)$ into U_1 that we call the translator of π .

It is easy to see that every saturated set of updates contains i , the identity mapping on S , and that every $u \in I_\pi$ is translated to i_1 , the identity mapping on π . For this reason, I is also called the identity set of π .

Given a subset X of $A(\pi)$, the translator h_π induces a partition π'_X on X as follows: $\forall u, v \in X$, u and v belong to the same block of π'_X iff $h_\pi(v) = h_\pi(u)$.

The set $R(X)$ of all views, for each of which X is a set of applicable updates, has been studied in [30]. A view in $R(X)$ is represented by the pair (π, π'_X) , where π is

the partition induced by the data base mapping on the state space and π'_X is the partition induced on X by the translator h_π of π . π represents the static part (values) and π'_X the dynamic part (updates) of the view .

The information content of a view in $R(X)$ is represented by the pair (π, π'_X) . A view (π_1, π'_{1X}) is considered to have more information content than the view (π_2, π'_{2X}) iff π_1 refines π_2 and π'_{1X} refines π'_{2X} .

3.3 Outline of the approach

In our approach, we take the ideas discussed so far as a starting point and proceed as follows:

First, we study the structure of the set F_S of all (data base) mappings on (a state space) S . In particular, we define the binary operation "concatenation" on F_S ; the concatenation "puts together" a set of mappings; in particular, a data base decomposition is defined as the concatenation of its components mappings (views).

Then we establish an equivalence relation, $\equiv$, on F_S , two mappings being equivalent iff they induce on S the same partition.

We define the binary operation "concatenation" on the set $F_S/\equiv$ of all equivalence classes as follows: the concatenation of two equivalence classes is the equivalence class of the concatenation of two mappings, one from each equivalence class.

There exists a semilattice isomorphism between the family $F_S/\equiv$ of all equivalence classes of mappings on S and the set $\Pi(S)$ of all partitions of S :

Next, we show that the set I_S of all saturated sets of updates on S is an intersection semilattice, isomorphic to $\Pi(S)$.

The double semilattice isomorphism, $F_S/\equiv \rightarrow \Pi(S) \rightarrow I_S$, allows us to identify an equivalence class of data base mappings, a state space partition, and a saturated set of updates. Thus we can identify data base transformations (views) to sets of updates of the original data base.

Then, we show that (referring to [30]) the mapping that maps each $\pi \in \Pi(S)$ to π' is monotonic with respect to refinement. As a result, the view information content, represented as (π, π') in [30], can be simply represented by π . This motivates our defining the product semilattice $\Pi(S)$ to be the structure of information content of data base mappings.

Finally, by the above double semilattice isomorphism, the intersection semilattice I_S is established to be the model-independent structure of information content of data base mappings.

As a first consequence of the above, a simple criterion for a decomposition to be lossless is established in model-independent terms.

As a second consequence, it follows that the integrity constraints that define the original data base can also be defined by a set of updates of the free data base.

Thus we obtain an operational definition of data bases where integrity constraints, information content of data base transformations and decompositions are characterized simply in terms of update operations, and it is known that update operations may be defined independently of the data model, e.g., by a set of axioms, as it is done in abstract data types.

In the remaining sections of this chapter, we establish the theoretical framework of the above concepts; the associated results constitute a major contribution of this thesis. We start with a brief review of the lattice structure of the family $\Pi(S)$ of all partitions of S .

3.4 The lattice $\Pi(S)$ of partitions of a data base state space S

On the set $\Pi(S)$ of all partitions of a set S , we define the relation "refinement", denoted $\leq$, and two operations, "the product" and "the sum", denoted $\cdot$ and $+$ respectively, as follows:

$$\forall \pi_1, \pi_2 \in \Pi(S) :$$

$$\pi_1 \leq \pi_2 \quad \text{iff} \quad \forall B_1 \in \pi_1, \text{ there exists } B_2 \in \pi_2 \text{ such that } B_1 \subset B_2 ;$$

$$\pi_1 \cdot \pi_2 = \{ B_1 \cap B_2 \neq \emptyset \quad / \quad B_1 \in \pi_1 \text{ and } B_2 \in \pi_2 \} ;$$

$$\pi_1 + \pi_2 = \{ \{ t \in S \quad / \quad \text{there exists a sequence in } S ,$$

$$s = s_0, s_1, s_2, \dots, s_n = t , \text{ such that}$$

$$s_i \text{ and } s_{i+1} \text{ belong to the same block}$$

$$\text{of } \pi_1 \text{ or to the same block of } \pi_2 ,$$

$$0 \leq i \leq n - 1 \} \quad / \quad s \in S \} .$$

It can be shown ([24]) that the refinement is a partial order on $\Pi(S)$, and that the product and the sum are indeed

binary operations on $\Pi(S)$; moreover, $\Pi(S)$ is a lattice under refinement, in which the infimum and supremum binary operations are the product and the sum respectively. It follows that :

$$\forall \pi_1, \pi_2 \in \Pi(S) :$$

$$\inf (\{ \pi_1, \pi_2 \}) = \pi_1 \cdot \pi_2 ;$$

$$\sup (\{ \pi_1, \pi_2 \}) = \pi_1 + \pi_2 ;$$

$$\pi_1 \leq \pi_2 \quad \text{iff} \quad \pi_1 = \pi_1 \cdot \pi_2 \quad \text{iff} \quad \pi_2 = \pi_1 + \pi_2 ;$$

The first and last elements of $\Pi(S)$ are respectively, the fine and the coarse partitions of S , denoted 0 and 1 , i.e.,

$$0 = \{ \{ s \} / s \in S \} ,$$

$$1 = \{ S \} .$$

Clearly, $\forall \pi \in \Pi(S) , \quad 0 \leq \pi \leq 1 .$

3.5/ A semilattice isomorphism between equivalence classes of data base mappings and partitions of the data base state space.

We denote F_S the set of all mappings on a set S . Each $f \in F_S$ induces on S a "natural" equivalence relation

$$\epsilon_f = \{ (s_1, s_2) \in S^2 \mid f(s_1) = f(s_2) \}.$$

We define on F_S a binary operation, called "concatenation" and denoted by " $*$ ", as follows:

$$\forall f, g \in F_S, \forall s \in S, f * g(s) = (f(s), g(s)).$$

In other words, $f * g$ maps each s in S to the ordered pair $(f(s), g(s))$.

In the data base context, the concatenation $f_1 * f_2 * \dots * f_n$ of the data base mappings $f_1, f_2, \dots, f_n$ is a composite mapping $d = f_1 * f_2 * \dots * f_n$.

An example of a composite mapping is a data base decomposition.

Theorem 3.1. $\forall f, g \in F_S, \epsilon_{f * g} = \epsilon_f \cap \epsilon_g$.

Proof. Let $(s_1, s_2) \in \epsilon_{f * g}$; then $f * g(s_1) = f * g(s_2)$; this implies

$(f(s_1), g(s_1)) = (f(s_2), g(s_2))$; it follows that $f(s_1) = f(s_2)$ and

$g(s_1) = g(s_2)$; it follows that $(s_1, s_2) \in \epsilon_f \cap \epsilon_g$; so, $\epsilon_{f * g} \subset \epsilon_f \cap \epsilon_g$.

Let $(s_1, s_2) \in \epsilon_f \cap \epsilon_g$; it follows that $f(s_1) = f(s_2)$ and $g(s_1) = g(s_2)$;

then $f \circ g(s_1) = f \circ g(s_2)$, so $(s_1, s_2) \in \epsilon_{f \circ g}$; thus $\epsilon_f \cap \epsilon_g \subseteq \epsilon_{f \circ g}$, Q.E.D

Definition 3.1 We define on F_S an equivalence relation, denoted by $\equiv$, as follows: $\forall f, g \in F_S$, $f \equiv g$ iff $\epsilon_f = \epsilon_g$.

$\forall f \in F_S$ we denote by $\bar{f}$ the equivalence class of f (with respect to $\equiv$)

Lemma 3.1 $\forall f, g \in F_S$, $f \circ g \equiv g \circ f$.

Proof. It follows immediately from Theorem 3.1.

We denote $F_S / \equiv$ the quotient set of F_S by $\equiv$, i.e., the set

$$\{ \bar{f} = \{ g \in F_S / g \equiv f \} / f \in F_S \}$$

of all equivalence classes of $\equiv$.

Definition 3.3 On $F_S / \equiv$ we define a binary operation denoted by " $\wedge$ ", and a binary relation, denoted by " $\leq$ ", as follows:

$\forall f, g \in F_S$, $\bar{f} \wedge \bar{g} = \overline{f \circ g}$; i.e., $\bar{f} \wedge \bar{g}$ is the equivalence class of the concatenation of f and g . By Theorem 3.1, $\wedge$ is well defined.

$\forall f, g \in F_S$, $\bar{f} \leq \bar{g}$, iff $\epsilon_f \subseteq \epsilon_g$.

Theorem 3.2 The relation $\leq$ is a partial order on $F_S / \equiv$.

Proof. Trivial.

Theorem 3.3 $F_S/\equiv$ is an infimum semilattice under $\leq$, and the infimum binary operation on $F_S/\equiv$ is $\wedge$.

Proof Let $\bar{f}, \bar{g} \in F_S/\equiv$. By Theorem 3.1, $\varepsilon_{f \star g} \subset \varepsilon_f$ and $\varepsilon_{f \star g} \subset \varepsilon_g$, so $\bar{f \star g} \leq \bar{f}$ and $\bar{f \star g} \leq \bar{g}$, and so $\bar{f \star g}$ is a lower bound of $\{\bar{f}, \bar{g}\}$. Let $\bar{h}$ be a lower bound of $\{\bar{f}, \bar{g}\}$. Then $\bar{h} \leq \bar{f}$ and $\bar{h} \leq \bar{g}$ imply $\varepsilon_f \subset \varepsilon_h$ and $\varepsilon_g \subset \varepsilon_h$; it follows that $\varepsilon_h \subset \varepsilon_f \cap \varepsilon_g = \varepsilon_{f \star g}$. Therefore $\bar{h} \leq \bar{f \star g}$, and so there exists the infimum of $\{\bar{f}, \bar{g}\}$ and it is equal to $\bar{f \star g} \in F_S/\equiv$. So $F_S/\equiv$ is an infimum semilattice under $\leq$, and the infimum binary operation is $\wedge$, because
 $\forall \bar{f}, \bar{g} \in F_S/\equiv, \inf \{\bar{f}, \bar{g}\} = \bar{f \star g} = \bar{f} \wedge \bar{g}$.

Theorem 3.4 The mapping $\omega : (F_S/\equiv, \leq) \rightarrow (\Pi(S), \leq)$, such that $\forall \bar{f} \in F_S/\equiv, \omega(\bar{f}) = S/\varepsilon_f$, the quotient set of S by ε_f , is a semilattice isomorphism. Thus ω preserves the semilattice partial orders; the first and last elements of the semilattices are, respectively:
 $\bar{i}$ = the set of all injections on S (i denotes the identity on S),
 $\bar{c}$ = the set of all constant mappings on S (c denotes a constant mapping on S),
 $\omega(\bar{i}) = 0 = \{\{s\}/s \in S\}$, the 'fine' partition of S ,
 $\omega(\bar{c}) = 1 = \{S\}$, the 'coarse' partition of S .

Proof. First, let us note that we have denoted with the same

symbol, $\leq$, the partial order on $F_S/\equiv$, defined above, and the refinement partial order on $\Pi(S)$. Let $\bar{f}, \bar{g} \in F_S/\equiv$, and suppose $\omega(\bar{f}) = \omega(\bar{g})$; then $S/\epsilon_f = S/\epsilon_g$; it follows that $\epsilon_f = \epsilon_g$, so $\bar{f} = \bar{g}$; thus ω is injective.

Let $\pi = \{B_j \subset S / j \in J\} \in \Pi(S)$. Define $f \in F_S$, such that $f(B_j) = j$, $\forall j \in J$. Clearly $\omega(\bar{f}) = \pi$, and so ω is surjective.

Let $f, g \in F_S$. Then $\omega(\bar{f} \wedge \bar{g}) = \omega(\bar{f} * \bar{g}) = S/\epsilon_{f * g}$. We have:

$$S/\epsilon_{f * g} = \{\{t \in S / f * g(t) = f * g(s)\} / s \in S\} =$$

$$\{\{t \in S / f(t) = f(s) \text{ and } g(t) = g(s)\} / s \in S\} =$$

$$\{\{t \in S / f(t) = f(s)\} \cap \{t \in S / g(t) = g(s)\} / s \in S\}$$

Recalling that $S/\epsilon_f = \{\{t \in S / f(t) = f(s)\} / s \in S\}$ and

$$S/\epsilon_g = \{\{t \in S / g(t) = g(s)\} / s \in S\},$$

and the definition of partition product we conclude that

$$S/\epsilon_{f * g} = S/\epsilon_f \cdot S/\epsilon_g, \text{ and so } \omega(\bar{f} \wedge \bar{g}) = \omega(\bar{f}) \cdot \omega(\bar{g}).$$

Thus ω is a bijection that preserves the semilattice operations, i.e., a semilattice isomorphism. Consequently, it preserves the semilattice partial orders. Indeed:

$$\forall f, g \in F_S, \bar{f} \leq \bar{g} \text{ iff } \bar{f} = \bar{f} \wedge \bar{g} \text{ iff } \omega(\bar{f}) = \omega(\bar{f} \wedge \bar{g}) \text{ iff}$$

$$\omega(\bar{f}) = \omega(\bar{f}) \cdot \omega(\bar{g}) \text{ iff } \omega(\bar{f}) \leq \omega(\bar{g}).$$

Finally, it is clear that if i, c denote respectively the identity and a fixed constant mapping on S , and if $0, 1$ denote respectively the fine and the coarse partition of S , then $\bar{i}, \bar{c}, 0, 1$ are, respectively, the first and last elements of $F_S / \equiv$ and $\Pi(S)$; and

$$\omega(\bar{i}) = S/\epsilon_i = 0, \quad \omega(\bar{c}) = S/\epsilon_c = 1, \quad \text{Q.E.D.}$$

3.6 The intersection semilattice I_S of saturated sets of updates of a data base with state space S

We use the following terminology:

S denotes the data base state space, i.e., the set of all data base instances that satisfy the integrity constraints;

U denotes the set of all (legal) data base updates, i.e., the set of all mappings from S into S ;

i denotes the identity mapping on S ;

$\Pi(S)$ denotes the set of all partitions of S ;

0 denotes the fine partition of S , i.e., $0 = \{\{s\} / s \in S\}$;

1 denotes the coarse partition of S , i.e., $1 = \{S\}$;

$P(U)$ denotes the power set of U ;

I_π (where π is a partition of S), denotes the set of all updates in U , that map each block of π into that block, i.e.,

$$I_\pi = \{u \in U / u(B) \subset B, \forall B \in \pi\}$$

We call I_π the saturated set of (updates with respect to) π ;

I_S denotes the family of all saturated sets of updates on S , i.e.,

$$I_S = \{I_\pi / \pi \in \Pi(S)\}.$$

Theorem 3.5 (i) The family I_S of all saturated sets of updates on S is an infimum semilattice under set-theoretic inclusion. The infimum binary operation on I_S is the intersection.

(ii) I_S has first and last elements, $I_0 = \{i\}$, and $I_1 = U$.

Proof. (i) $P(U)$ is an infimum semilattice under set-theoretic inclusion with infimum binary operation, the intersection; as $I_S \subset P(U)$, it suffices to show that I_S is closed under intersection. Consider any $I_{\pi_1}, I_{\pi_2} \in I_S$; π_1, π_2 are partitions of S and so is their product $\pi_1 \cdot \pi_2$. Clearly $I_{\pi_1 \cdot \pi_2} \in I_S$.

I shall show that $I_{\pi_1} \cap I_{\pi_2} = I_{\pi_1 \cdot \pi_2}$, $\forall \pi_1, \pi_2 \in \Pi(S)$.

Let u be in $I_{\pi_1 \cdot \pi_2}$ and let B_1 be a block of π_1 ; then,

$$B_1 = B_1 \cap S = B_1 \cap (u \cup \{ B_2 \subset S / B_2 \in \pi_2 \}) =$$

$$u \cup \{ B_1 \cap B_2 \subset S / B_2 \in \pi_2 \}.$$

It follows that:

$$u(B_1) = u(u \cup \{ B_1 \cap B_2 \subset S / B_2 \in \pi_2 \}) =$$

$$= u \{ u(B_1 \cap B_2) \subset S / B_2 \in \pi_2 \} = u \{ B_1 \cap B_2 \subset S / B_2 \in \pi_2 \} =$$

$$= B_1 \cap (u \cup \{ B_2 \subset S / B_2 \in \pi_2 \}) = B_1 \cap S = B_1.$$

Hence $u(B_1) \subset B_1$, which implies that u is in I_{π_1} . It follows that:

$$I_{\pi_1 \cdot \pi_2} \subset I_{\pi_1}$$

By a similar argument, it follows that :

$$I_{\pi_1 \cdot \pi_2} \subset I_{\pi_2}$$

Therefore, it follows that:

$$I_{\pi_1 \cdot \pi_2} \subset I_{\pi_1} \cap I_{\pi_2}$$

The next step is to show that $I_{\pi_1} \cap I_{\pi_2} \subset I_{\pi_1 \cdot \pi_2}$. Let u be in

$I_{\pi_1} \cap I_{\pi_2}$, and consider any block B of $\pi_1 \cdot \pi_2$; then $B = B_1 \cap B_2$,

for some B_1 in π_1 and some B_2 in π_2 ; it follows that:

$$u(B) = u(B_1 \cap B_2) \subset u(B_1) \subset B_1; \text{ and}$$

$$u(B) = u(B_1 \cap B_2) \subset u(B_2) \subset B_2$$

It follows that :

$$u(B) \subset B_1 \cap B_2 = B$$

Therefore u is in $I_{\pi_1 \cdot \pi_2}$; this implies that :

$$I_{\pi_1} \cap I_{\pi_2} \subseteq I_{\pi_1 \cdot \pi_2}.$$

Therefore,

$$I_{\pi_1} \cap I_{\pi_2} = I_{\pi_1 \cdot \pi_2} \in I_S.$$

Thus I_S is an infimum semilattice under set-theoretic inclusion and

the corresponding infimum binary operation is the intersection.

Clearly $\forall I_{\pi_1}, I_{\pi_2} \in I_S$, $I_{\pi_1} \subseteq I_{\pi_2}$ iff $I_{\pi_1} = I_{\pi_1} \cap I_{\pi_2}$.

(ii) Recalling the definition of the fine and coarse partitions of S

and the definition of I_{π} , $\pi \in \Pi(S)$, we easily conclude that:

$$I_0 = \{i\} \text{ and } I_1 = U.$$

Observing that $i \in I_{\pi}$, $\forall \pi \in \Pi(S)$, we easily conclude that $\{i\}$ and

U are respectively the first and the last element of I_S . Q.E.D.

3.7 A semilattice isomorphism between partitions of a data base state space and saturated sets of updates

Theorem 3.6 The mapping $I : (\Pi(S), \leq) \rightarrow (I_S, \subseteq)$, that corresponds to each partition $\pi \in \Pi(S)$, its saturated set of updates I_π , is an infimum semilattice isomorphism. (Consequently, the mapping I preserves the partial orders $\leq$ and $\subseteq$ of the semilattices; in particular the first and the last elements $\{i\}$ and U of I_S are the isomorphic images of the first and last elements, 0 and 1, of $\Pi(S)$ respectively: $I(0) = I_0 = \{i\}$, and $I(1) = I_1 = U$).

Proof. First we show that I is injective. Let π_1, π_2 be in $\Pi(S)$ and suppose $\pi_1 \neq \pi_2$; it follows that there exist $B_1 \in \pi_1$ and $B_2 \in \pi_2$ such that $B_1 \neq B_2$ and $B_1 \cap B_2 \neq \emptyset$; therefore, we can take a state $s_1 \in B_1 \cap B_2$ and a state s_2 that belongs to only one of B_1, B_2 . Without any loss of generality, we may assume that $s_2 \in B_1 - B_2$.

Define now the following mapping u from S into S :

$$u(s) = s, \quad \forall s \in S - \{s_1\} \quad \text{and}$$

$$u(s_1) = s_2.$$

It follows that u is in I_{π_1} but not in I_{π_2} . Clearly $I_{\pi_1} \neq I_{\pi_2}$, and thus I is injective.

Clearly I is surjective; because, as $I_S = \{ I_\pi / \pi \in \Pi(S) \}$, any $I_\pi \in I_S$ is the image of $\pi \in \Pi(S)$ through I , i.e., $I(\pi) = I_\pi$.

Therefore I is a bijection.

Next, we show that I preserves the semilattice operations, product and intersection. Let $\pi_1, \pi_2 \in \Pi(S)$. In the proof of Theorem 3.5, we have shown that : $\forall \pi_1, \pi_2 \in \Pi(S)$, $I_{\pi_1} \cap I_{\pi_2} = I_{\pi_1 \cdot \pi_2}$. As, by

definition, $I(\pi_1 \cdot \pi_2) = I_{\pi_1 \cdot \pi_2}$, it follows that :

$$\forall \pi_1, \pi_2 \in \Pi(S), \quad I(\pi_1 \cdot \pi_2) = I(\pi_1) \cap I(\pi_2) ,$$

i.e., the bijection I preserves the semilattice operations and therefore I is a (infimum) semilattice isomorphism.

Consequently, I preserves the semilattice partial orders. Indeed:

$$\forall \pi_1, \pi_2 \in \Pi(S), \quad \pi_1 \leq \pi_2 \quad \text{iff} \quad \pi_1 = \pi_1 \cdot \pi_2 \quad \text{iff} \quad I(\pi_1) = I(\pi_1 \cdot \pi_2)$$

$$\text{iff} \quad I(\pi_1) = I(\pi_1) \cap I(\pi_2) \quad \text{iff} \quad I(\pi_1) \subset I(\pi_2) .$$

Finally, the first and the last element of $\Pi(S)$ are mapped, through I , to the first and the last element of I_S respectively:

$$I(0) = I_0 = \{i\} , \quad \text{and} \quad I(1) = I_1 = U , \quad \text{Q.E.D.}$$

3.8 The fundamental isomorphism theorem for data base mappings

The previous discussion can be summarized in the following :

Theorem 3.7 There exists a double infimum semilattice isomorphism between the following :

- (i) The set $\Pi(S)$ of all partitions of a data base state space S
- (ii) The family $F_S/\equiv$ of all equivalence classes of data base mappings on S (two mappings being equivalent iff they induce the same partition on S)
- (iii) The set I_S of all saturated sets of updates on S

$$(F_S/\equiv, \leq, \wedge) \xrightarrow{\omega^{-1}} (\Pi(S), \leq, \cdot) \xrightarrow{I} (I_S, \subset, \cap)$$

In the remainder of this chapter, we discuss some consequences of the fundamental isomorphism theorem.

3.9 The expressive power of saturated sets of updates

The importance of the isomorphism theorem for data base mappings lies in the fact that the natural set-theoretic intersection algebra of saturated sets of updates becomes the algebra of information content of data base mappings.

The information content hierarchy of mappings, represented by the 'refinement' partial order on partitions, is converted isomorphically into the set-theoretic inclusion hierarchy on the family of saturated sets of updates. This representation allows us to replace complicated operations between data base mappings by simple set-theoretic operations: intersection and inclusion. So, the hierarchy of sets of updates is more natural than the hierarchy of partitions in a data base context, because in practice, state space partitions appear only implicitly, i.e., underlying the data base processes, in contrast to data base updates which constitute explicitly a fundamental concept in the data base processes). "Addition" of the information content of views is represented by intersection of their corresponding saturated sets of updates. Comparison of the information content of two views corresponds to set-theoretic inclusion on their saturated sets of updates.

We make use of these facts in order to establish a simple characterization of the decompositions that preserve information (lossless decompositions) and to make explicit the relationship that exists between updates and integrity constraints.

3.10 The structure of the information content of views in terms of partitions of the state space

The set $R(X)$ of all views, for each of which X is a set of applicable updates, has been studied in [30]. A view in $R(X)$ is represented by the pair (π, π'_X) , where π is the partition induced on the state space by the data base mapping, and π'_X is the partition induced on X by the translator h_π of π ; π represents the static part (values), and π'_X represents the dynamic part (updates) of the view. The information content of a view in $R(X)$ is represented by the pair (π, π'_X) . A view (π_1, π'_{1X}) is considered to have more information content than the view (π_2, π'_{2X}) , iff π_1 refines π_2 and π'_{1X} refines π'_{2X} .

In this section, we will establish a simpler representation of view information content in terms of partitions of the state space only. We use the following notation:

For a given set X of (update) operations on the state space S , we denote by :

P_X , the set of all partitions of S whose applicable update set $A(\pi)$ is a superset of X , i.e.,

$$P_X = \{ \pi \in \Pi(S) \mid A(\pi) \supset X \},$$

$h_{\pi_X} = h_\pi / X$, the restriction of the mapping h_π to X ,

$\varepsilon_{h_{\pi_X}}$, the natural equivalence relation induced on X by h_{π_X} ,

$\Pi(X)$, the set of all partitions of X ,

$\pi'_X = X / \varepsilon_{h_{\pi_X}} \in \Pi(X)$, the quotient set of X by $\varepsilon_{h_{\pi_X}}$, i.e.,

the partition induced on X by h_{π_X} ,

$$R(X) = \{ (\pi, \pi'_X) / A(\pi) \supset X \} = \{ (\pi, \pi'_X) / \pi \in P_X \}.$$

Theorem 3.8 The mapping $g_X : P_X \rightarrow \Pi(X)$, such that $\forall \pi \in P_X$, $g_X(\pi) = \pi'_X$, that associates to each partition π in P_X , the partition π'_X that h_{π_X} induces on X , is monotonic with respect to refinement, i.e., $\forall \pi_1, \pi_2 \in P_X$, $\pi_1 \leq \pi_2$ implies $\pi'_{1X} \leq \pi'_{2X}$.

Proof. Let $\pi_1, \pi_2 \in P_X$ be such that $\pi_1 \leq \pi_2$.

It follows that $\pi_1 = \pi_1 \cdot \pi_2$.

Therefore, $\pi'_{1X} = (\pi_1 \cdot \pi_2)'_X$.

By Lemma [30], it follows that

$$(\pi_1 \cdot \pi_2)'_X = \pi'_{1X} \cdot \pi'_{2X}.$$

It follows that

$$\pi'_{1X} = \pi'_{1X} \cdot \pi'_{2X}$$

As $\Pi(X)$ is a lattice, it follows that

$$\pi'_{1X} \leq \pi'_{2X}, \quad \text{Q.E.D.}$$

Corollary 3.8.1 $\forall (\pi_1, \pi'_{1X}), (\pi_2, \pi'_{2X}) \in R(X)$,
 $\pi_1 \leq \pi_2$ and $\pi'_{1X} \leq \pi'_{2X}$ iff $\pi_1 \leq \pi_2$.

From the above Corollary 3.8.1, it follows that for any $\pi_1, \pi_2 \in \Pi(S)$, the condition $\pi_1 \leq \pi_2$ alone, implies that the view π_1 has more information content than view π_2 , with respect to any set X of updates that are applicable to both views. In other words, in order to compare the information contents of two views, we only need to compare the static parts of the views and, in addition, the result of our comparison will be valid for any set X of updates that are applicable to both views.

Motivated by the above discussion, we replace the definition of view information content - which is given as (π, π'_X) in [30] - as follows :

Definition 3.4 The information content of a view is the partition π induced on the data base state space by the view definition mapping.

A data base view is a carrier of information concerning the data base in the following sense :

Our a priori information concerning the " current " data base state s , is merely that s is an element of the data base state space S . Consider now a view v , and let π denote the state space partition induced by its definition mapping f . Knowing the current view - v state t , increases our information on s , because we know now that s is an element of the block $B = f^{-1}(t)$ of π , which is a subset of S .

In conclusion, a view is indeed an information carrier for the underlying data base whose " measure ", i.e., information content, according to Definition 3.4, is assigned to be the induced data base state space partition.

Let us see to what extent this Definition 3.4 captures the semantics of view information content.

We discuss first how comparison of views as information carriers is expressed through state space partitions. Having identified a view with its induced partition, we observe that for any two views $\pi_1, \pi_2 \in \Pi(S)$ such that $\pi_1 \leq \pi_2$, each π_1 - block is subset of a corresponding π_2 - block . Thus, knowing the current π_1 - state , i.e., knowing the block B_1 of π_1 in which the current data base state s is , gives us more information about s than if we knew the current π_2 - state , i.e., the block B_2 of π_2 to which s belongs (because B_1 is a subset of B_2).

Clearly, the refinement partial order on data base state space partitions expresses the hierarchy of view information content. We formalize this fact with the following definition:

Definition 3.5 $\forall \pi_1, \pi_2 \in \Pi(S)$,
 π_1 " is at least as informative as " π_2 iff $\pi_1 \leq \pi_2$.

Let us note that as $0 \leq \pi \leq 1 \quad \forall \pi \in \Pi(S)$, the views 0 and 1 are respectively the most and the least informative ones. In fact, knowing (S and) the current view - 0 - state, i.e., a (singleton) 0 - block in which the current data base state s is, we know s exactly. Thus the view 0 contains all the information about the data base. On the other hand, if we know (S and) the current view - 1 - state, i.e., the single block S of 1 in which s is, we do not actually obtain any information about the current data base state s.

We discuss next how the " addition " (merging) of views as information carriers is expressed through state space partitions.

Consider a set $d = \{ \pi_j \in \Pi(S) / j \in J \}$ of views, where

$J = \{1, 2, \dots, n\}$, and the view $\pi = \pi_1 \cdot \pi_2 \dots \pi_n$.

If we know the current π - state, i.e., some block $B \in \pi$,

then we also know the set of current states of all views in d:

it is the set of blocks $\{ B_j \in \pi_j \text{ such that } B_j \supset B / j \in J \}$.

If we know the current states of all views in d, i.e., some

set $\{ B_j / j \in J \}$ of blocks - one from each view -

then we also know the current view - π state : it is the block

$$B' = \bigcup_{j \in J} B'_j$$

of π ; therefore , the collection $d = \{ \pi_1, \pi_2, \dots, \pi_n \}$ of views and the view $\pi = \pi_1 \cdot \pi_2 \dots \pi_n$ are equivalent information carriers with respect to the data base. This property is captured by Definition 3.4. We can see that immediately, by recalling from the isomorphism ω , of Theorem 3.7 , that if the views $v_1, v_2, \dots, v_n$ induce on S the partitions $\pi_1, \pi_2, \dots, \pi_n$ respectively, then their concatenation $v_1 * v_2 * \dots * v_n$ induces on S the partition $\pi_1 \cdot \pi_2 \dots \pi_n$ and so , by Definition 3.4, the concatenation $v_1 * v_2 * \dots * v_n$ has information content equal to $\pi_1 \cdot \pi_2 \dots \pi_n$. Clearly, the partition product expresses the " addition " (or merging, or concatenation) of views. We formalize this fact with the following definition:

Definition 3.6 The information content of a set of views $d = \{ \pi_i \in \mathcal{H}(S) \mid i = 1, \dots, n \}$, is the infimum of d , $\inf(d) = \pi_1 \cdot \pi_2 \dots \pi_n$.

From Definition 3.5 and Definition 3.6 , we obtain immediately the following theorems :

Theorem 3.9 Two arbitrary sets d_1, d_2 of data base views ,
 $d_1 = \{ \pi_{1j} \in \Pi(S) / j=1, \dots, n \}$, $d_2 = \{ \pi_{2k} \in \Pi(S) / k=1, \dots, m \}$
 carry the same information content iff

$$\pi_{11} \cdot \pi_{12} \cdots \pi_{1n} = \pi_{21} \cdot \pi_{22} \cdots \pi_{2n} .$$

Theorem 3.10 A data base decomposition
 $d = \{ \pi_1, \pi_2, \dots, \pi_n \} \subset \Pi(S)$, preserves the data base
 information, i.e., it is lossless, iff $\pi_1 \cdot \pi_2 \cdots \pi_n = 0$.

In conclusion : the induced partition on the data base state
 space S is a natural measure of the view information content ; the
 product semilattice $\Pi(S)$ is the natural structure of the information
 contents of all the data base views; it provides a convenient algebra,
 in which the partition product operation and the refinement partial
 order allow us to " add " (merge) and to compare, respectively,
 the information content of the data base views .

3.11 The structure of the information content of views in terms of saturated sets of updates

We have seen in the preceding section that the semilattice $(\Pi(S), \leq, \dots)$ of partitions of a data base state space S provides a natural structure for modelling the view information content: the partition product expresses the information content of a set of views, whereas the refinement partial order on partitions expresses the hierarchy of view information content.

However, the basic concept of this model - the data base state space partition - has no real status in the data base context.

Using the fundamental isomorphism theorem for data base mappings (Theorem 3.7), we overcome the above problem by replacing our model with one in which the basic notion is the update, a fundamental concept in data base processes.

Thus, by virtue of the isomorphism

$$I : (\Pi(S), \leq, \dots) \rightarrow (I_S, \subset, \cap),$$

the semilattice $(I_S, \subset, \cap)$ of saturated sets of updates on a data base state space S is the view information content model. Therefore, we convert the results of the preceding section as follows:

Definition 3.7 The information content of a view (which induces on the data base state space S the partition) π is I_π , the saturated set of updates of the view, i.e., the set of all the data base updates that leave the view invariant.

Definition 3.8 For any two views v_1, v_2 (inducing on S the partitions π_1, π_2), we define that v_1 " is at least as informative as " v_2 iff $I_{\pi_1} \subset I_{\pi_2}$,

i.e., iff every data base update that leaves v_1 invariant, leaves v_2 also invariant.

Let us note that as $I_0 = \{i\} \subset I_\pi \subset I_1 = U \quad \forall \pi \in \Pi(S)$, and $\omega^{-1}(0)$ is the set of injections on S , and $\omega^{-1}(1)$ is the set of constant mappings on S , any injective view is at least as informative as any view, and any view is at least as informative as a constant view.

Clearly, the information content of the data base itself is equal to the information content of any injective view, i.e., $I_0 = \{i\}$.

Definition 3.9 The information content of a set

$d = \{ \pi_j \in \Pi(S) / j = 1, 2, \dots, n \}$ of views is :

$$I(\pi_1, \pi_2, \dots, \pi_n) = \bigcap_{j=1}^n I(\pi_j)$$

Theorem 3.11 Two arbitrary sets d_1, d_2 of data base views,

$$d_1 = \{ \pi_{1j} \in \Pi(S) / j=1, \dots, n \}, \quad d_2 = \{ \pi_{2k} \in \Pi(S) / k=1, \dots, m \},$$

carry the same information content iff

$$\bigcap_{j=1}^n I(\pi_{1j}) = \bigcap_{k=1}^m I(\pi_{2k})$$

Theorem 3.12 A data base decomposition,

$$d = \{ \pi_1, \pi_2, \dots, \pi_n \} \subseteq \Pi(S),$$

preserves the data base information, i.e., it is lossless, iff

$$\bigcap_{j=1}^n I(\pi_j) = \{ i \}$$

In conclusion: The saturated set of updates of the induced partition is a natural measure of the view information content; the intersection semilattice I_S of all saturated sets (of updates of a data base state space S) is the natural structure of the information content of all the data base views; it provides a convenient algebra in which the set-theoretic intersection and inclusion allow us to "add" (merge) and to compare, respectively, the information content of the data base views.

3.12. Data base decompositions preserving the information content (Lossless decompositions)

The Theorem 3.12, of the preceding section, provides a model - independent criterion for a decomposition to preserve the data base information content (lossless decomposition). This criterion can also be stated as follows :

Corollary 3.12.1 A data base decomposition preserves the data base information (is lossless) iff any change of the data base state implies a change of the state of at least one component view.

We apply our criterion for lossless decompositions in two examples of a lossy and a lossless decomposition.

Consider a data base variable (Figure 3.2), consisting of a single relation variable, STUDENT, over the attribute set $\{ N, C, G, T \}$, the attribute symbols standing for NAME, COURSE, GRADE, and TEACHER respectively, which take values in some appropriate domains. The tuple (n, c, g, t) is in the relation STUDENT iff the student n takes course c currently by teacher t and has current grade average g on this course. The data base state space S is defined by the integrity constraints which we assume to be two functional dependencies : $\{ C \} \rightarrow \{ T \}$ and $\{ N, C \} \rightarrow \{ G \}$, which must hold in every instance of the relation variable STUDENT.

Relation schema :

STUDENT (NAME, COURSE, GRADE, TEACHER)

Integrity constraints (Functional dependencies) :

{ COURSE } $\rightarrow$ { TEACHER } /,

{ NAME, COURSE } $\rightarrow$ { GRADE }

Projectional views

Induced partitions

π_1

π_2

π_3

$d_{12} = \{ \pi_1, \pi_2 \}$ is a lossy decomposition .

$d_{13} = \{ \pi_1, \pi_3 \}$ is a lossless decomposition .

Figure 3.2 A lossy and a lossless decomposition of a data base .

We consider the three views that result by projection of STUDENT over the attribute sets

$\{ N, C, G \}$, $\{ N, G, T \}$, $\{ C, T \}$

and which induce on S the partitions π_1, π_2, π_3 respectively, as shown in Figure 3.2.

Example 3.1

A lossy decomposition

Using our criterion, we will show that the data base decomposition

$d_{12} = \{ \pi_1, \pi_2 \}$ is lossy.

Consider the data base instances:

$s_1 = \{ (n_1, c_1, g_1, t_1), (n_1, c_2, g_1, t_2) \}$,

$s_2 = \{ (n_1, c_1, g_1, t_2), (n_1, c_2, g_1, t_1) \}$,

where $c_1 \neq c_2$ and $t_1 \neq t_2$.

Clearly, s_1 and s_2 satisfy the integrity constraints and so, $s_1, s_2 \in S$.

We define the data base update u as follows :

$u(s_1) = s_2$ and $u(s) = s \quad \forall s \in S - \{s_1\}$.

In other words, u changes only one state of S , the state s_1 (which contains only two tuples both for the same student taking two courses by different teachers , and having the same grade in both courses) to the state s_2 (resulting from s_1 by switching the teachers of the courses).

We have :

$$s_1[E\{N, C, G\}] = \{(n_1, c_1, g_1), (n_1, c_2, g_1)\},$$

$$s_2[E\{N, C, G\}] = \{(n_1, c_1, g_1), (n_1, c_2, g_1)\},$$

$$s_1[E\{N, G, T\}] = \{(n_1, g_1, t_1), (n_1, g_1, t_2)\},$$

$$s_2[E\{N, G, T\}] = \{(n_1, g_1, t_1), (n_1, g_1, t_2)\}.$$

It follows that s_1 and s_2 belong to the same block of π_1 ; and they also belong to the same block of π_2 . Therefore :

$$u \in I(\pi_1) \cap I(\pi_2).$$

As $u \neq i$, the decomposition d_{12} is lossy, i.e., it does not preserve the information content of the data base.

Clearly, d_{12} cannot be a lossless - join decomposition ([31]).

This is easily verified; we have: $\{N, C, G\} \cap \{N, G, T\} = \{N, G\}$, but neither $\{N, G\} \rightarrow \{N, C, G\}$ nor $\{N, G\} \rightarrow \{N, G, T\}$ is implied by the given integrity constraints.

Example 3.2A lossless decomposition

Using our criterion, we will show that the data base decomposition
 $d_{13} = \{ \pi_1, \pi_3 \}$ is lossless.

Let $u \in I(\pi_1) \cap I(\pi_3)$ and $s \in S$. It follows that :

$$u(s)[\{ N, C, G \}] = s[\{ N, C, G \}] \quad \text{and} \quad u(s)[\{ C, T \}] = s[\{ C, T \}] .$$

Let $(n, c, g, t) \in u(s)$. It follows that :

$$(n, c, g) \in s[\{ N, C, G \}] \quad \text{and} \quad (c, t) \in s[\{ C, T \}] ;$$

therefore , we have (n, c, g, t') , $(n', c, g', t) \in s$, for some appropriate values t', n', g' .

As $\{C\} \rightarrow \{T\}$ holds in s , $t' = t$; it follows that
 $(n, c, g, t) \in s$ and therefore $u(s) \subset s$.

Let now $(n, c, g, t) \in s$. It follows that :

$$(n, c, g) \in u(s)[\{ N, C, G \}] \quad \text{and} \quad (c, t) \in u(s)[\{ C, T \}] ;$$

therefore , we have (n, c, g, t') , $(n', c, g', t) \in u(s)$, for some appropriate values t', n', g' .

As $\{C\} \rightarrow \{T\}$ holds in $u(s)$ $t' = t$; it follows that $(n, c, g, t) \in u(s)$ and thus $s \subset u(s)$, and therefore $u(s) = s$; it follows that $u = i$, so $I(\pi_1) \cap I(\pi_3) \subset \{i\}$. Clearly, $i \in I(\pi_1) \cap I(\pi_3)$; thus $I(\pi_1) \cap I(\pi_3) = \{i\}$, i.e., d_{13} is a lossless decomposition.

The important thing to note here is that, in both examples, we have arrived at our conclusions without making essential use of the given dependencies. Of course, we used the given dependencies in order to decide whether the instances s_1 and s_2 were legal; however, the nature of these dependencies did not influence in any way whatsoever the process of deciding whether the decompositions were lossless. On the contrary, the relational theory relies heavily on the nature of the dependencies in order to decide losslessness. In our Example 3.1, for instance, one must check whether either $\{N, G\} \rightarrow \{C\}$ or $\{N, G\} \rightarrow \{T\}$ is implied by the given dependencies; whereas this criterion is only valid for functional dependencies, our criterion (Theorem 3.12) is valid for any kind of dependencies.

In concluding, let us note that the losslessness criterion of the relational theory (lossless - join decomposition) applies only to the decompositions that are derived by projections (projectional decompositions) and only for some particular families of integrity constraints; the isomorphism theorem for data base mappings offers as an application, through Theorem 3.12, an

alternative, model - independent criterion of lossless decomposition, using only data base updates, which is in fact more general than the lossless - join criterion of the relational model, for the following reasons :

- (1) It is model - independent.
- (2) It applies to any kind of views, not just projections.
- (3) Every lossless - join decomposition is necessarily lossless, but there are lossless decompositions that are not lossless - join decompositions.

3.13 Lossless - join decompositions of the relational model and lossless decompositions

We discuss here the relationship that exists between our model - independent criterion for a decomposition to be lossless and the lossless - join criterion for a decomposition in the relational model context.

We consider a data base with data base state space S . We assume that the data base is represented in the relational model by a single relation variable s over the attribute set A . In the relational model context the set S can be defined by a set of integrity constraints, for example, a set of dependencies such as functional, multivalued, etc.

Let $R = \{ R_j \subset A \mid j = 1, 2, \dots, k \}$ be a covering

of A , i.e., R is such that the union of its members is A . R is called a data base scheme ([31]).

Let us recall that a relation s over an attribute set A is a finite set of mappings on A , each of which maps each attribute $A_j \in A$ to an element of a specified set D_j , called the domain of the attribute A_j .

We will denote by $s[Y]$, the projection of s into a subset Y of A , i.e., the set of the restrictions on Y of all the mappings in s .

We will be discussing the decomposition

$$d = \{ \pi_j \in \Pi(S) \mid j = 1, 2, \dots, k \}$$

resulting from projections of the relation variable s on the members of R ; such a decomposition will be called projectional.

For $j = 1, 2, \dots, k$, π_j denotes the partition induced on S by the projection of s on R_j .

For $s \in S$, $\pi_j(s)$ denotes the set of states in S that have the same projection on R_j as s ; in other words $\pi_j(s)$ denotes the block of s in the partition π_j , induced on S through projection on R_j .

We define on S the project - join mapping m_R ([9]), as follows:

$$\forall s \in S, \quad m_R(s) = \bigstar_{j=1}^k s[R_j] =$$

$$= \{ \mu, \text{ a mapping on } A \mid \text{there exist mappings } \lambda_1, \lambda_2, \dots, \lambda_k$$

in s such that μ and λ_j agree on R_j , for $j=1,2,\dots,k \}$...

In the case where S is defined by a set of dependencies, the mapping m_R is said to preserve dependencies ([9]) iff

$$m_R(S) \subseteq S.$$

In general, we say that the project $\leftarrow$ join mapping m_R preserves the integrity constraints iff $m_R(S) \subset S$, i.e., iff m_R is a data base update (a mapping from S into S).

Lemma 3.3 If a projectional decomposition is a lossless - join decomposition, then it is a lossless decomposition.

In other words : If a decomposition $d = \{ \pi_j \in \Pi(S) / j=1,2,\dots,k \}$ (resulting from projections over the members of a data base scheme $R = \{ R_j \subset A / j=1,2,\dots,k \}$ of a data base consisting of a single relation over an attribute set A) is lossless - join , then d is lossless, i.e.,

$$\bigcap_{j=1}^k I(\pi_j) = \{ i \} .$$

Proof. As d is lossless - join, $m_R = i$, i.e., the identity mapping on S .

Let $u \in \bigcap_{j=1}^k I(\pi_j)$; then u is applicable in all components

of the decomposition and also leaves them invariant ; therefore :

$$\forall s \in S, \quad u(s)[R_j] = s[R_j], \quad j = 1,2,\dots,k \quad (1)$$

$$\text{Thus, } \forall s \in S, \quad u(s) = m_R(u(s)) = \bigstar_{j=1}^k u(s)[R_j] =$$

$$= \bigstar_{j=1}^k s[R_j] = m_R(s) = s.$$

Thus, $u = i$ and therefore $\bigcap_{j=1}^k I(\pi_j) \subset \{i\}$.

Clearly, $i \in \bigcap_{j=1}^k I(\pi_j)$ and therefore $\bigcap_{j=1}^k I(\pi_j) = \{i\}$, Q.E.D.

Example 3.3

Consider a data base variable that consists of a single relation variable R over the attributes A, B, C . We define a data base by imposing as integrity constraints, the functional dependency

$$\{B\} \rightarrow \{A\} \quad (1)$$

and let S denote the corresponding state space.

We know from the relational theory that the decomposition d that results from projections on the attribute sets $\{A, B\}$ and $\{B, C\}$ is a lossless - join decomposition; then, by Lemma 3.3, d is also lossless; we can arrive at the same conclusion by direct application of our criterion for losslessness, i.e., Corollary 3.12.1.

Suppose that the current data base state s_1 changes; then, a tuple deletion or a tuple insertion must have occurred in the current data base state. Let s_2 denote the updated state.

(1) Suppose a deletion occurred.

Let $(a, b, c) \in s_1 - s_2$. Assume that none of the view states changed. Then for some appropriate values a_1, c_1 , we will have :
 $(a, b, c_1), (a_1, b, c) \in s_2$; by the given functional dependency it follows that $a_1 = a$ and thus $(a, b, c) \in s_2$, a contradiction.

(2) Suppose an insertion occurred.

Let $(a, b, c) \in s_2 - s_1$. Assume that none of the view states changed. Then for some appropriate values a_1, c_1 , we will have :
 $(a, b, c_1), (a_1, b, c) \in s_1$; by the given functional dependency it follows that $a_1 = a$ and thus $(a, b, c) \in s_1$, a contradiction.

Thus, in either case, a change of the data base state necessarily implies a change of state in at least one component view of d .
 Therefore, by Corollary 3.12.1, the data base decomposition d is lossless, Q.E.D.

Lemma 3.3 shows that the lossless - join property of a decomposition in the relational model context implies the lossless decomposition property, expressed with the model - independent criterion of saturated sets of updates. The converse of Lemma 3.3 is not true, in general, i.e., a lossless projectional data base decomposition is not necessarily a lossless - join decomposition. We demonstrate this by the following counterexample.

Counterexample 3.1

Referring to the definitions and notations at the beginning of this section, consider the following:

$A = \{ B, C \}$ is an attribute set ,

$R = \{ R_1, R_2 \}$ is a scheme of A , where

$R_1 = \{ B \}$ and $R_2 = \{ C \}$,

$S = \{ s, s' \}$ is the state space of a data base variable which consists of a single relation variable over A , where

$s = \{ (b_1, c_1), (b_2, c_2) \}$ and

$s' = \{ (b'_1, c'_1), (b'_2, c'_2) \}$, where all the attribute values are distinct.

Let π_1, π_2 be the partitions induced on S by the projectional views over the attribute sets R_1 and R_2 .

Clearly, $\pi_1 = \pi_2 = 0$ and thus the saturated set of each view contains only the identity mapping i ; therefore

$$I(\pi_1) \cap I(\pi_2) = \{ i \}$$

22
OF / DE

By Theorem 3.12, the projectional decomposition over the scheme R is lossless; however, it is not a lossless - join decomposition because :

$$\begin{aligned} m_R(s) &= s[B] * s[C] = \{ b_1, b_2 \} * \{ c_1, c_2 \} = \\ &= \{ (b_1, c_1), (b_1, c_2), (b_2, c_1), (b_2, c_2) \} \neq s, \text{ Q.E.D.} \end{aligned}$$

The following lemma provides a condition under which a lossless projectional decomposition is necessarily a lossless - join decomposition:

Lemma 3.4 A lossless projectional decomposition over a scheme R is necessarily a lossless - join decomposition iff the project - join mapping m_R is a data base update.

In other words : A lossless projectional decomposition $d = \{ \pi_j \in \Pi(S) / j = 1, 2, \dots, k \}$ (of a data base consisting of one relation variable over an attribute set A and having state space S) over a scheme $R = \{ R_j \subset A / j=1, \dots, k \}$ is necessarily a lossless - join decomposition iff the project - join mapping m_R is a data base update, i.e., iff $m_R(S) \subset S$, i.e., iff m_R preserves the integrity constraints of the data base.

Proof. Suppose that the project - join mapping m_R is a

data base update and that d is a lossless projectional decomposition over the scheme R . Let $s \in S$. As

$$m_R(s) = \bigstar_{j=1}^k s[R_j], \quad \text{it follows that}$$

$$m_R(s)[R_j] = s[R_j] \quad \text{for } j = 1, 2, \dots, k.$$

Therefore, $m_R(s)$ and s belong to the same block of π_j for $j = 1, 2, \dots, k$.

It follows that $m_R \in I(\pi_j)$ for $j = 1, 2, \dots, k$.

Therefore, $m_R \in \bigcap_{j=1}^k I(\pi_j)$.

As d is a lossless decomposition, it follows (by Theorem 3.12) that :

$$\bigcap_{j=1}^k I(\pi_j) = \{ i \}.$$

It follows that $m_R = i$, and therefore the decomposition d is a lossless - join decomposition.

Conversely, suppose that the lossless projectional decomposition d is also a lossless - join decomposition ; then, of course, $m_R = i$, and so m_R is indeed a data base update, Q.E.D.

Lemma 3.4 should be interpreted as follows :

The project - join mapping m_R maps data base states into data base states ; however, m_R is not necessarily a " legal " data base update because, in general, $m_R(S)$ need not be a subset of the data base state space S . If m_R is a legal data base update, then the condition

$$\bigcap_{j=1}^k I(\pi_j) = \{ i \}$$

guarantees that the projectional decomposition d over the scheme R is a lossless - join decomposition .

In conclusion, Lemma 3.3 and Lemma 3.4 offer an " operational " interpretation of the lossless - join property .

3.14 Definition of integrity constraints as a saturated set of updates

As we have discussed in CHAPTER 2 , the semantics of the real world facts represented in a data base give rise to integrity constraints. These constraints are properties that the data base values must satisfy in order to faithfully represent reality.

Let us consider a data base variable R of type T . In order to specify a data base $B = (R, S)$ of type T , we need to specify the set $S \subseteq T$ of all legal values (data base state space) , i.e., the set of all values in T that satisfy the integrity constraints c :

$$S = \{ R \in T \mid c(R) = \text{true} \} .$$

Thus the integrity constraints are presented in the form of a partition of the data type T that has at most two blocks, i.e., unless all data type values in T are legal, the integrity constraints induce on T the partition

$$\pi = \{ S, T-S \} .$$

The fundamental isomorphism theorem motivates the following definition:

Definition 3.10 The integrity constraints of a data base of type T are a saturated set D of operations on T which corresponds to a partition of T that has at most two blocks.

The interpretation of a saturated set of operations on the data type as integrity constraints is as follows :

As any $u \in D$ maps S into itself and also maps $T - S$ into itself, the integrity constraint $D = I(\pi)$ is the set of all update operations on T that are legal with respect to the state spaces S and $T - S$. Clearly, the set of legal updates of the data base with state space S , is the set of the restrictions on S of the operations in D .

In conclusion, the meaning of a saturated set of operations as integrity constraints is as follows:

A saturated set D of operations on T as the integrity constraints of a data base $B = (R, S)$ of type T is the set of operations on T whose restrictions on S form the set of all allowable update operations on T for the data base B . Each update operation in D maps legal states to legal states and illegal states to illegal states.

3.15 Characterization of the family of integrity constraints

We wish to characterize the set of all integrity constraints that can be defined on a data base type T , i.e., the set P_T of all partitions of T that have at most two blocks, or its isomorphic image $C_T = I(P_T)$. We first show the following:

Lemma 3.5 Any two partitions of a set T that each has two blocks are incomparable with respect to refinement.

Proof. Consider two distinct partitions π_1, π_2 of T , each of which has two blocks. For any partition π of T and for any $t \in T$, let $\pi(t)$ denote the block of π that contains t . As the partitions are distinct, there exists $s, t \in T$ such that $t \in (\pi_j(s) - \pi_k(s))$ for some appropriate permutation (j, k) of $\{1, 2\}$. Thus π_j does not refine π_k . Assuming that π_k refines π_j we obtain:

$$\pi_k(t) \subset \pi_j(t) \quad \text{and} \quad \pi_k(s) \subset \pi_j(s) ,$$

and as $\pi_j(t) = \pi_j(s)$ we conclude that $\pi_j(s) \neq T$, contradicting our hypothesis; therefore the partition π_k does not refine the partition π_j , Q.E.D.

Clearly a partition of T that has two blocks does not

refine any partition that has more than two blocks; it follows that any partition of T that has at most two blocks refines only itself and the trivial coarse partition 1 of T .

Clearly, the only partitions that refine only themselves and 1 are the members of P_T . Therefore we obtain the following characterization for P_T :

$$P_T = \{ \pi \in \Pi(T) \mid \forall \pi_1 \in \Pi(T) - \{\pi\}, \pi \leq \pi_1 \text{ implies } \pi_1 = 1 \}.$$

Then by the fundamental isomorphism theorem we obtain the following characterization for C_T :

$$C_T = \{ D \in I_T \mid \forall D_1 \in I_T - \{D\}, D \subset D_1 \text{ implies } D_1 = U \},$$

where U and I_T denote respectively the set of update operations on T and the family of saturated sets of updates on T .

In other words, the family C_T of integrity constraints on a data base type T consists of all the saturated sets of updates on T that occupy the two highest ranking levels in the semilattice hierarchy $(I_T, \subset)$ of all the saturated sets of updates on T .

Figure 3.3 shows a pictorial representation of the family C_T of all the integrity constraints that can be defined on a data base type T .

In conclusion, we have seen that a particular class of saturated sets of updates can represent all the integrity constraints.

Figure 3.3 Pictorial representation of integrity constraints.

3.16 Conclusion

In this chapter we have given a complete characterization of data base mappings in terms of saturated sets of updates. This is one of the major results of this thesis. The expressive power of the saturated sets of updates is explored in various contexts, such as the view information content structure, lossless data base decompositions and integrity constraints representation. These results complete the operational approach to data bases by providing a model - independent representation of the data base information carriers in terms of a unique set of updates, the saturated set. Whereas the operational approach uses arbitrary definition sets for a mapping and represents the view information content as a pair of partitions, we proceeded as follows :

We represented a data base mapping with the largest possible definition set of updates which we termed " saturated " and we showed the existence of a double semilattice isomorphism between the set of equivalence classes of data base mappings (two mappings being equivalent iff they induce equal partitions on the state space) , the set of partitions of the state space , and the family of saturated sets of updates.

Next, we showed that the information content of a data base mapping can be completely and simply characterized by the partition induced on the state space.

Consequently, we established the product semilattice of state space partitions, or isomorphically, the intersection semilattice of saturated sets of updates to be the structure of information content of data base mappings.

The latter is a natural, model - independent representation of the structure of information content of data base mappings, using only one basic concept - the update - and has the advantage over the former, that it replaces refinement and partition product by simple, set - theoretic inclusion and intersection of sets of updates, to represent comparison and " addition " of the information content of data base mappings.

Next, we applied these results in order to establish a model - independent characterization of lossless decompositions and finally, we characterized those saturated sets of updates that represent all the integrity constraints of a data base.

CHAPTER 4

GENERALIZED DATA BASE MAPPINGS

An analysis of some information - theoretic aspects of data bases is undertaken. The data base mapping is extended to a more general data base information carrier, the relation on two state spaces. We introduce the state space covering as a measure of the information content of a relational information carrier. Relations (generalized data base mappings) as information carriers and their representation by coverings are used to model incomplete information representation. An algebra for the information content of a relation as an information carrier is presented.

4.1. Introduction

In Chapter 3 we have examined some information - theoretic aspects of data bases. The basic data base information carrier, the data base mapping, has been studied in a model - independent approach. The fundamental isomorphism theorem (Theorem 3.8) identifies equivalence classes of data base mappings to induced state space partitions and also to saturated sets of updates. Thus, the refinement partial order on partitions or, isomorphically, the set - theoretic inclusion on saturated sets of updates expresses the hierarchy of information content of data base mappings.

However, the data base mapping is not the only data base information carrier. In the data base processes, there are cases where a data base state can be transformed to more than one view state. This can be, for example, the case in a data base that contains incomplete information. Clearly, these " generalized data base mappings " are not represented by mappings, but by relations which are subsets of the cartesian product of the state spaces of the data base and of its view.

In this chapter we generalize the concept of a partition, induced on the data base state space by a data base mapping, to that of coverings, induced on the state spaces of the data base and of its view by the data base relation. Thus, we introduce a theory for the representation of information

(knowledge) on the data base through state space coverings which, we believe, can form a basis for future data base investigations. In a mathematically concise manner it encompasses several central and recurring concepts found in the literature, such as incomplete information, view complement and independence. We introduce formal notions of relative information capacity and we obtain several interesting results concerning these notions. The advantage of the proposed representation is that it can treat complete and incomplete information in a uniform way.

In section 2 we explain how coverings can represent information content (knowledge) of the data base or part thereof. An algebra of coverings is studied in section 3 . This algebra allows us to " add " knowledge or ignorance of the data base contained in two coverings and also to compare the information content of two coverings; the interpretation of the structure of coverings in the knowledge context is presented in section 4 . In the remaining sections 5, 6, 7 we apply our theory to the notions of view complement, incomplete information and independence. An interesting result is that our intuitive notion of independence is not always verified when arbitrary coverings are used for knowledge representation.

4.2 Knowledge representation

Let S, V be two sets (state spaces) and consider a relation $R \subseteq S \times V$ on S and V . We assume that $\forall s \in S, \forall v \in V$ there exist $v' \in V, s' \in S$ such that $(s, v'), (s', v) \in R$.

The relation R induces on S and V the following coverings:

$$\sigma = \{ B_s = \{ v \in V / (s, v) \in R \} / s \in S \} \in C(V),$$

$$\sigma' = \{ B_v = \{ s \in S / (s, v) \in R \} / v \in V \} \in C(S).$$

Also the relation R induces on σ and σ' the following injections:

$$f_R : \sigma \rightarrow S \quad \text{and} \quad f'_R : \sigma' \rightarrow V \quad \text{such that}$$

$$\forall B_s \in \sigma, \quad \forall B_v \in \sigma' \quad f_R(B_s) = s \quad \text{and} \quad f'_R(B_v) = v,$$

i.e., f_R maps each block of σ to that value of V to which every element of the block is R -related, and f'_R maps each block of σ' to that value of S to which every element of the block is R -related.

Let us recall that a covering of a set S is a family of

non - empty subsets of S whose union is S . We denote by $C(S)$ the set of coverings of S .

In general, in a relation R , every element s of S is related to more than one element of V , and every element of V is related to more than one element of S .

Assuming S and V denote the state spaces of a data base and of a view of it, it is clear that a relation $R \subset S \times V$ is a carrier of information concerning the data base and the view: Knowing the "current" view state v , we are informed that the current data base state s is in the block $B_v \subset S$; similarly, knowing that the "current" data base state is s , we are informed that the current view state is in $B_s \subset V$.

The difference between a mapping and a relation as information carriers lies in the fact that any two distinct view states represent disjoint blocks of data base states in the case of a mapping, whereas this is not true for a relation in general.

We can say that a data base mapping is a "disjoint information carrier", whereas a relation is an "overlapping information carrier", in general.

The above distinction of information carriers is, of course, reflected in the fact that a relation, in general, induces a covering on the data base state space S , whereas a mapping, which is a special relation, induces on S a pairwise disjoint covering, i.e., a partition.

Let us see some cases of data base transformations which correspond to relations (generalized mappings) in the state spaces of the data base and of its view.

Consider a data base with state space S and two views w_1 and w_2 of it; let π_1, π_2 denote the partitions induced on S by the views w_1, w_2 respectively. The operational context is such that a user does not know the current data base state, but only the current w_1 - state (i.e., he has incomplete information of the data base). from which he must deduce the current state of w_2 . For any $s \in S$, let B_1^S, B_2^S denote respectively the π_1 - block and the π_2 - block that contain s .

Clearly, when the current data base state is s , the current w_1 - state is $B_1^S \in \pi_1$; however, all the user can deduce at view - w_2 level, knowing merely B_1^S , is a " set of candidate current w_2 - states ", i.e., the set of π_2 - blocks each of which intersects B_1^S . Thus the correspondences between S and π_2 , and between π_1 and π_2 are two generalized mappings, i.e., the two relations :

$$R = \{ (s, B_2) \in S \times \pi_2 / B_1^S \cap B_2 \neq \emptyset \} \subset S \times \pi_2,$$

$$R_{21} = \{ (B_1, B_2) \in \pi_1 \times \pi_2 / B_1 \cap B_2 \neq \emptyset \} \subset \pi_1 \times \pi_2.$$

Let us have a closer look at the meaning of these relations as information carriers.

Case 1 : The relation- R between the state spaces of the data base and the view w_2 .

The set of data base states to which a block $B_2 \in \pi_2$ is R - related is the following:

$$U\{ B_1 \in \pi_1 / B_1 \cap B_2 \neq \emptyset \} .$$

The set of π_2 - blocks to which a given $s \in S$ is R - related is the following:

$$\{ B_2 \in \pi_2 / B_2 \cap B_1^S \neq \emptyset \} .$$

Thus, R induces on S and π_2 the following coverings:

$$\sigma' = \{ \{ s \in S / (s, B_2) \in R \} / B_2 \in \pi_2 \} =$$

$$= \{ \{ B_1 \in \pi_1 / B_1 \cap B_2 \neq \emptyset \} / B_2 \in \pi_2 \} \in C(S) ,$$

$$\sigma = \{ \{ B_2 \in \pi_2 / (s, B_2) \in R \} / s \in S \} =$$

$$= \{ \{ B_2 \in \pi_2 / B_2 \cap B_1^S \neq \emptyset \} / s \in S \} \in C(\pi_2) .$$

Either of the coverings σ, σ' characterizes the relation R with respect to information content. The fact that the current

data base state if a given value s will be manifested at the level of the generalized view as any view - w_2 state (i.e., any π_2 - block) that belongs to the set

$$\{ B_2 \in \pi_2 / B_2 \cap B_1^S \neq \emptyset \}$$

which, of course, contains B_2^S .

Case 2 : The relation R_{21} between the state spaces of the views w_1 and w_2 .

The relation R_{21} is completely characterized with respect to information content by either of the following coverings :

$$\sigma'_{21} = \{ \{ B_1 \in \pi_1 / B_1 \cap B_2 \neq \emptyset \} / B_2 \in \pi_2 \} \in C(\pi_1) ,$$

$$\sigma_{21} = \{ \{ B_2 \in \pi_2 / B_2 \cap B_1 \neq \emptyset \} / B_1 \in \pi_1 \} \in C(\pi_2) .$$

Clearly, $\sigma_{21}, \sigma'_{21}$ are information carriers for the views w_1 and w_2 ; for example, σ'_{21} informs us that when the current π_2 - state is B_2 , the current π_1 - state is a block in

$$\{ B_1 \in \pi_1 / B_1 \cap B_2 \neq \emptyset \} .$$

Similarly, σ_{21} informs us that when the current π_1 - state is B_1 , the current π_2 - state is a block in

$$\{ B_2 \in \pi_2 / B_2 \cap B_1 \neq \emptyset \}.$$

Let us note that a set of blocks of a given partition is completely determined by the union of its blocks; therefore, each of the coverings $\sigma_{21} \in C(\pi_2)$ and $\sigma'_{21} \in C(\pi_1)$ can be completely characterized by a covering of S as follows:

$$\bar{\sigma}_{21} = \{ \cup \{ B_2 \in \pi_2 / B_2 \cap B_1 \neq \emptyset \} / B_1 \in \pi_1 \} \in C(S),$$

$$\bar{\sigma}'_{21} = \{ \cup \{ B_1 \in \pi_1 / B_1 \cap B_2 \neq \emptyset \} / B_2 \in \pi_2 \} \in C(S).$$

Clearly, either of the coverings $\bar{\sigma}_{21}$ $\bar{\sigma}'_{21}$ of S completely characterizes the relation (generalized view) R_{21} with respect to information content, and thus both are information carriers for the views w_1 and w_2 .

It is clear from the above discussion that the coverings of the data base state space S are information carriers for the data base or part thereof.

In general, the information (knowledge) carried by a covering (information carrier) $\sigma_1 \in C(S)$, given another covering $\sigma_2 \in C(S)$, is represented by the covering:

$$\sigma_{21} = \{ \cup \{ B_2 \in \sigma_2 / B_2 \cap B_1 \neq \emptyset \} / B_1 \in \sigma_1 \} \in C(S).$$

Let us see in a simple example of Case 2 how a covering of the data base state space represents knowledge of a view, given another view.

Example 4.1

Let us consider a Data Base (DB) containing the names of employees of a company who are assigned to a project. At any given time, at most two employees, named Peter and John, can be assigned to that project; thus, the possible data base instances are subsets of the set { Peter, John } as follows:

<u>Data base state space S</u>	<u>Reference number</u>
$\emptyset$	1
{ Peter }	2
{ John }	3
{ Peter, John }	4

We denote by S the data base state space, but we shall identify states with their reference numbers, i.e.,

$$S = \{ 1, 2, 3, 4 \} .$$

Consider the following queries:

Q_1 : Is " John " in the data base ?

Q_2 : Is " Peter " in the data base ?

Q_3 : What is the number of employees in the data base ?

The set of possible answers to query Q_1 is { Yes, No } ; the set of possible answers to query Q_2 is the same ; and the set of possible answers to query Q_3 is { 0, 1, 2 } .

Figure 4.1 (a) shows the answers to these queries for each data base state.

Figure 4.1 (b) shows the partitions π_{01} , π_{02} , π_{03} that are induced on S by the three queries. The partition 0 is the fine partition of S , i.e., each block of 0 is a singleton block.

Before issuing any queries, all we know about the data base is that it can contain at most the two names: Peter and John . Therefore, any of the states 1, 2, 3, 4 is possible. Suppose now that we ask the query Q_1 and the answer is " No " . Then we know that the only possible states are 1 or 2 . Similarly, if the answer is " Yes " , then the possible instances are 3 or 4 . In other words, (a block of) the partition π_{01} represents our knowledge (about the current state) of the data base, given

Figure 4.1 Three DB queries and the partitions they induce on the state space.

(the answers to) query Q_1 . We shall denote this knowledge by DB/Q_1 . Similarly, π_{02} represents DB/Q_2 , and π_{03} represents DB/Q_3 .

Next, let us try to represent the knowledge of a view Q_3 of the data base, given (only) the answers to a query Q_1 . (Let us note that for the purposes of our analysis we make no distinction between a view and a query, as a view is simply a query whose definition is stored by the data base system).

Before issuing any queries, all we know about the current state of the view Q_3 is that it is one of the three possible states that are represented by the three blocks

$\{ 1 \}$, $\{ 2, 3 \}$, $\{ 4 \}$

of the partition π_{03} . Suppose now that we ask query Q_1 and the answer is " No ". We know then that the current data base state is either 1 or 2 ; thus, the only possible instances for the view Q_3 are the π_{03} - blocks that contain at least one of the states 1, 2 , i.e., the blocks: $\{ 1 \}$ and $\{ 2, 3 \}$ of π_{03} .

It should be noted that the above blocks are completely characterized by their union $\{ 1, 2, 3 \}$; (clearly, the above blocks can be " computed " by the data base instances 1, 2, 3). What is important to note here is that the current view Q_3 - state may be determined by the instance 3 which is not the current data base state. Similarly, if the answer to query Q_1 is " Yes ", the only possible instances for the current data base state are 3 or 4 ; it follows that the only possible instances for the current state

of the view Q_3 are the π_{03} -blocks $\{2, 3\}$ and $\{4\}$; these blocks are completely characterized by their union $\{2, 3, 4\}$. In other words, our knowledge of view Q_3 given (only) the view Q_1 , which we will denote by Q_3/Q_1 , is the collection of two subsets of S , namely $\{\{1, 2, 3\}, \{2, 3, 4\}\}$.

What is important to note is that the members of that collection overlap; the union of these members equals S .

Clearly, the above collection is a covering of S . Thus, we conclude that the above covering of S represents the knowledge Q_3/Q_1 of the view Q_3 , given the view Q_1 .

In general, we conclude that the knowledge Q_3/Q_1 of a view Q_3 , given the view Q_1 , is represented by the following covering of the data base state space:

$$\sigma_{31} = \{ \cup \{ B \in \pi_{03} / B \cap A \neq \emptyset \} \cdot / A \in \pi_{01} \} \in C(S).$$

4.3 The lattice of maximal coverings of a data base state space

In this section we present an algebra of coverings that allows us to "add" the knowledge and also to "add" the ignorance contained in two coverings of the data base state space S .

Definition 4.1 A covering of a set S is a family of non-empty subsets of S whose union is S .

We wish to establish a hierarchy, i.e., a partial order on the set $C(S)$ of all coverings of S , in order to express the hierarchy of knowledge represented by coverings, similarly to the way in which refinement partial order on the set $\Pi(S)$ of all partitions of S provides a hierarchy of information content, represented by partitions.

Let us consider an example: In Figure 4.2, the covering π_{01} (which is in fact a partition) of S is a refinement of the covering σ_{31} , i.e., each block of π_{01} is a subset of a block of σ_{31} . Intuitively we consider that this is the reason why π_{01} is us to know more than σ_{31} . This motivates us to define the relation "refinement" on $C(S)$:

Definition 4.2 For any σ, τ in $C(S)$, we say that " σ refines τ ", denoted $\sigma \leq \tau$, iff $\forall B \in \sigma$, there exists $B' \in \tau$, such that $B \subset B'$.

Figure 4.2 A "traditional" view π_{01} and a "generalized" view σ_{31}

The relation "refinement" is clearly reflexive and transitive; it is not a partial order on $C(S)$, however, because it is not antisymmetric; the latter can be easily established by the counter-example in Figure 4.3.

The reason why refinement is not an antisymmetric relation on $C(S)$ is that a covering in $C(S)$ may contain two blocks, one of which is a subset of the other. We overcome this problem by restricting ourselves to the set of all "maximal" coverings of S .

Definition 4.3 A covering σ of S is called "maximal" iff all its blocks are maximal elements of σ , i.e., iff $\forall B, B' \in \sigma, B \subset B'$ implies $B' = B$.

We denote by $MC(S)$ the set of all maximal coverings of S . Clearly, $MC(S) \subset C(S)$.

We define an operation on $C(S)$, denoted by "max", which associates to each covering of S a maximal covering of S as follows:

Definition 4.4 The operation $\max : C(S) \rightarrow MC(S)$, is such that $\forall \sigma \in C(S)$, $\max(\sigma)$ = the set of maximal elements of σ .

Clearly, every element of $\max(\sigma)$ is a maximal element of $\max(\sigma)$, and thus $\max(\sigma)$ is a maximal covering of S for every covering σ of S .

Figure 4.3 The relation $\leq$ on $C(S)$ is not antisymmetric.

Counterexample :

$$S = \{1, 2, 3, 4\}, \quad \sigma = \{\{1, 2, 3\}, \{4\}\}, \quad \tau = \{\{1, 2, 3\}, \{2\}, \{4\}\}$$

Clearly, σ and τ are coverings of S .

Every block of σ is subset of a block of τ ; every block of τ is subset of a block of σ .

Thus, $\sigma \leq \tau$ and $\tau \leq \sigma$. However, $\sigma \neq \tau$.

On the set $MC(S)$ of maximal coverings of S we define two binary operations, the "product", denoted by $\circ$, and the "sum", denoted by $\oplus$, as follows:

Definition 4.5 $\forall \sigma, \tau \in MC(S)$:

$$\sigma \circ \tau = \max(\{ A \cap B \mid A \in \sigma, B \in \tau \}), \quad \sigma \oplus \tau = \max(\sigma \cup \tau).$$

As the following theorem shows, the relation refinement restricted on $MC(S)$ is indeed a partial order on $MC(S)$.

Theorem 4.1 The set $MC(S)$ of the maximal coverings of a finite set S is a distributive lattice under the refinement partial order and has first and last elements, the fine and the coarse partitions of S respectively; the infimum and supremum binary operations are the product, $\circ$, and the sum, $\oplus$, respectively.

Proof. We first show that the refinement relation is a partial order on $MC(S)$.

Reflexivity. Clearly, every $\sigma \in MC(S)$ refines itself, so $\sigma \leq \sigma$.

Transitivity. Let $\sigma, \tau, \phi \in MC(S)$ be such that $\sigma \leq \tau$ and $\tau \leq \phi$. Let $B \in \sigma$; then $\sigma \leq \tau$ implies that there exists $B' \in \tau$ such that $B \subset B'$; also $\tau \leq \phi$ implies that there exists $B'' \in \phi$ such that $B' \subset B''$. It follows that $B \subset B''$, so σ refines ϕ , i.e., $\sigma \leq \phi$.

Antisymmetry. Let $\sigma, \tau \in MC(S)$ be such that $\sigma \leq \tau$ and $\tau \leq \sigma$.

Let $B \in \sigma$; then $\sigma \leq \tau$ implies that there exists $B' \in \tau$ such that $B \subset B'$; also $\tau \leq \sigma$ implies that there exists $B'' \in \sigma$ such that $B' \subset B''$; it follows that:

$$B \subset B' \subset B'' \quad (1)$$

and therefore,

$$B \subset B'' \quad (2)$$

As B and B'' are members of a maximal covering σ , we conclude from (2) that

$$B = B'' \quad (3)$$

From (1) and (3) it follows that $B = B' \in \tau$ and thus $\sigma \subset \tau$.

In a similar way we can show that $\tau \subset \sigma$ and therefore $\sigma = \tau$; thus, the refinement is antisymmetric on $MC(S)$; finally, we conclude that the refinement is a partial order on $MC(S)$.

We next show that for any $\sigma, \tau \in MC(S)$, there exists their supremum and it is equal to $\sigma \vee \tau$, an element of $MC(S)$. Let $\sigma, \tau \in MC(S)$.

Clearly, $\sigma \leq \sigma \vee \tau$ and $\tau \leq \sigma \vee \tau$; it follows that

$$\max(\sigma) \subset \max(\sigma \vee \tau) \text{ and } \max(\tau) \subset \max(\sigma \vee \tau);$$

as σ, τ are maximal coverings, the operation $\max$ leaves them invariant; it follows that

$$\sigma \leq \max(\sigma \vee \tau) \text{ and } \tau \leq \max(\sigma \vee \tau).$$

It follows that $\max(\sigma \vee \tau)$ is an upper bound of $\{\sigma, \tau\}$.

Let ϕ be any upper bound of $\{\sigma, \tau\}$; let $B \in \max(\sigma \vee \tau)$;

it follows that $B \in \phi$ and therefore B belongs to at least one of σ, τ .

As $\sigma \leq \phi$ and $\tau \leq \phi$, there exists $B' \in \phi$ such that $B \subset B'$.

It follows that $\max(\sigma \cup \tau) \leq \phi$ and therefore $\max(\sigma \cup \tau)$ is the least upper bound of $\{\sigma, \tau\}$, i.e.,

$$\sup \{\sigma, \tau\} = \max(\sigma \cup \tau) = \sigma \cup \tau \in MC(S).$$

We next show that for any $\sigma, \tau \in MC(S)$, there exists their infimum and it is equal to $\sigma \cap \tau$, an element of $MC(S)$. Let $\sigma, \tau \in MC(S)$.

Clearly, $\sigma \cap \tau = \max(\{A \cap B / A \in \sigma, B \in \tau\}) \leq \{A \cap B / A \in \sigma, B \in \tau\}$.

Any element of $\sigma \cap \tau$ is of the form $A \cap B$, for some $A \in \sigma$ and for some $B \in \tau$; as $A \cap B$ is subset of both A and B , we conclude $\sigma \cap \tau \leq \sigma$ and $\sigma \cap \tau \leq \tau$; it follows that $\sigma \cap \tau$ is a lower bound of $\{\sigma, \tau\}$.

Suppose ϕ is a lower bound of $\{\sigma, \tau\}$, so $\phi \leq \sigma$ and $\phi \leq \tau$.

Let $B \in \phi$; then there exists $B' \in \sigma$ and $B'' \in \tau$ such that $B \subset B'$ and $B \subset B''$; it follows that $B \subset B' \cap B''$.

Recalling the definition of the operation product $\circ$ (Definition 4.5), we conclude that either $B' \cap B''$ belongs to $\sigma \cap \tau$, or a superset B_0 of $B' \cap B''$ belongs to $\sigma \cap \tau$; in either case B is a subset of an element of $\sigma \cap \tau$; it follows that $\phi \leq \sigma \cap \tau$ and therefore $\sigma \cap \tau$ is the greatest lower bound of $\{\sigma, \tau\}$, i.e.,

$$\inf \{\sigma, \tau\} = \sigma \cap \tau \in MC(S).$$

Thus, $MC(S)$ is a lattice under refinement.

Clearly, the fine and the coarse partitions of S , $\{\{s\} / s \in S\}$

and $\{S\}$, denoted by 0 and 1 respectively, are maximal coverings of S .

It is easy to see that 0 and 1 are the first and the last elements respectively of the lattice $(MC(S), \leq)$.

We next show that the lattice $(MC(S), \leq)$ is distributive.

Let $\sigma_1, \sigma_2, \sigma_3 \in MC(S)$. It follows that :

$$\begin{aligned}
 \sigma_1 \circ (\sigma_2 \bullet \sigma_3) &= \max(\{B_1 \cap B / B_1 \in \sigma_1, B \in (\sigma_2 \bullet \sigma_3)\}) = \\
 &= \max(\{B_1 \cap B / B_1 \in \sigma_1, B \in \max(\sigma_2 \cup \sigma_3)\}) = \\
 &= \max(\{B_1 \cap B / B_1 \in \sigma_1, B \in (\sigma_2 \cup \sigma_3)\}) = \\
 &= \max(\{B_1 \cap B_2 / B_1 \in \sigma_1, B_2 \in \sigma_2\} \cup \{B_1 \cap B_3 / B_1 \in \sigma_1, B_3 \in \sigma_3\}) = \\
 &= \max([\max(\{B_1 \cap B_2 / B_1 \in \sigma_1, B_2 \in \sigma_2\})] \cup [\max(\{B_1 \cap B_3 / B_1 \in \sigma_1, B_3 \in \sigma_3\})]) = \\
 &= \max([\sigma_1 \circ \sigma_2] \cup [\sigma_1 \circ \sigma_3]) = \\
 &= (\sigma_1 \circ \sigma_2) \bullet (\sigma_1 \circ \sigma_3) \quad \text{Q.E.D.}
 \end{aligned}$$

Corollary 4.1.1 The infimum and the supremum, or the product $\circ$ and the sum $\bullet$, are binary operations on the lattice $(MC(S), \leq)$ which are idempotent, commutative and associative and $\circ$ is distributive with respect to $\bullet$.

Corollary 4.1.2 For every $\sigma, \tau \in MC(S)$, $\sigma \leq \tau$ iff
 $\sigma \circ \tau = \sigma$ iff $\sigma \odot \tau = \tau$.

Both corollaries are immediate consequences of the fact that $MC(S)$ is a lattice under $\leq$.

Remark 4.1 It should be noted that the set $\Pi(S)$ of all partitions of S consists of maximal coverings of S , and so $\Pi(S) \subset MC(S)$. It should also be noted that the operation product, " $\circ$ ", on $MC(S)$ is an extension of the operation product, " $\cdot$ ", on $\Pi(S)$; however, the operation sum, " $\odot$ ", on $MC(S)$ is not an extension of the operation sum, " $+$ ", on $\Pi(S)$; in other words, in the lattices $(\Pi(S), \cdot, +)$, $(MC(S), \circ, \odot)$, the sum operations, " $+$ " and " $\odot$ ", have different meanings; this is exhibited in the following example:

Example 4.2 Let $S = \{1, 2, 3, 4\}$ and consider the partitions
 $\pi_1 = \{\{1, 2, 3\}, \{4\}\}$, $\pi_2 = \{\{1, 2\}, \{3, 4\}\}$.

We have of course $\pi_1, \pi_2 \in \Pi(S) \subset MC(S)$.

It is clear by their definitions that " $\cdot$ " and " $\circ$ " will always give the same result when applied to elements of $\Pi(S)$. In this example we obtain the following:

$$\pi_1 \cdot \pi_2 = \{B_1 \cap B_2 \neq \emptyset \mid B_1 \in \pi_1, B_2 \in \pi_2\} =$$

$$= \{\{1, 2\}, \{3\}, \{4\}\}.$$

$$\begin{aligned}
 \pi_1 \circ \pi_2 &= \max(\{ B_1 \cap B_2 \mid B_1 \in \pi_1, B_2 \in \pi_2 \}) = \\
 &= \max(\{ \{ 1,2 \}, \emptyset, \{ 3 \}, \{ 4 \} \}) = \\
 &= \{ \{ 1,2 \}, \{ 3 \}, \{ 4 \} \} .
 \end{aligned}$$

Thus, the results obtained by the two product operations are the same. However, the two sum operations, "+" and "⊕", do not, in general, yield the same result when applied to elements of $\Pi(S)$.

As proof, let us calculate $\pi_1 + \pi_2$ and $\pi_1 \oplus \pi_2$.

The block B of $\pi_1 + \pi_2$ that contains the element 3 includes the blocks: $\{ 1,2,3 \}$ of π_1 and $\{ 3,4 \}$ of π_2 ; therefore,

$B = \{ 1,2,3,4 \}$; it follows that $B = S$ and therefore

$$\pi_1 + \pi_2 = 1 .$$

On the other hand,

$$\begin{aligned}
 \pi_1 \oplus \pi_2 &= \max(\pi_1 \cup \pi_2) = \\
 &= \max(\{ \{ 1,2,3 \}, \{ 4 \}, \{ 1,2 \}, \{ 3,4 \} \}) = \\
 &= \{ \{ 1,2,3 \}, \{ 3,4 \} \} \neq 1 .
 \end{aligned}$$

Clearly, $\pi_1 \oplus \pi_2 \neq \pi_1 + \pi_2$.

Thus, the "sum" operations should always be taken in the context

of the lattice under consideration $(\Pi(S), ., +)$, or
 $(MC(S), \circ, \oplus)$.

From Example 4.2 we also see, as expected, that when we add two partitions with the sum operation of the maximal coverings, we do not, in general, obtain a partition but a (maximal) covering of S .

Lemma 4.1 The lattice $(MC(S), \circ, \oplus)$ of maximal coverings of S is distributive, whereas the lattice $(\Pi(S), ., +)$ of partitions of S is not.

Proof. The first part of the lemma is proved in Theorem 4.1. We show that the lattice of partitions is not distributive by the following counterexample:

Counterexample 4.1 Let $S = \{1, 2, 3\}$ and consider the following partitions of S :

$$\pi_1 = \{\{1, 2\}, \{3\}\},$$

$$\pi_2 = \{\{1\}, \{2, 3\}\},$$

$$\pi_3 = \{\{1, 3\}, \{2\}\}.$$

We first calculate $\pi_2 + \pi_3$.

The block B of $\pi_2 + \pi_3$ that contains the element 3 includes

the blocks : $\{ 2,3 \}$ of π_2 and $\{ 1,3 \}$ of π_3 . Therefore,

$$B \supset \{ 1,2,3 \} ;$$

it follows that $B = S$ and therefore

$$\pi_2 + \pi_3 = 1 .$$

Next, we obtain:

$$\pi_1 \cdot \pi_2 = \{ \{ 1 \} , \{ 2 \} , \{ 3 \} \} = 0 , \quad \text{and}$$

$$\pi_1 \cdot \pi_3 = \{ \{ 1 \} , \{ 2 \} , \{ 3 \} \} = 0 .$$

It follows that:

$$\pi_1 \cdot (\pi_2 + \pi_3) = \pi_1 \cdot 1 = \pi_1 \neq 0 , \quad \text{whereas}$$

$$(\pi_1 \cdot \pi_2) + (\pi_1 \cdot \pi_3) = 0 + 0 = 0 .$$

$$\text{So } \pi_1 \cdot (\pi_2 + \pi_3) \neq (\pi_1 \cdot \pi_2) + (\pi_1 \cdot \pi_3) ,$$

and therefore the lattice $(\Pi(S) , . , +)$ is not distributive.

However, the lattice $(MC(S) , o , \bullet)$ is distributive,

as it was shown in Theorem 4.1. We verify this result for the coverings $\pi_1, \pi_2, \pi_3 \in MC(S)$. We obtain the following:

$$\pi_1 \circ \pi_2 = \pi_1 \cdot \pi_2 = 0$$

$$\pi_1 \circ \pi_3 = \pi_1 \cdot \pi_3 = 0$$

$$\pi_2 \bullet \pi_3 = \max(\pi_2 \cup \pi_3) =$$

$$= \max(\{\{1\}, \{2,3\}, \{1,3\}, \{2\}\}) =$$

$$= \{\{2,3\}, \{1,3\}\},$$

$$\pi_1 \circ (\pi_2 \bullet \pi_3) = \{\{1,2\}, \{3\}\} \bullet \{\{2,3\}, \{1,3\}\} =$$

$$= \max(\{\{1\}, \{2\}, \{3\}\}) = \{\{1\}, \{2\}, \{3\}\} = 0.$$

Thus we obtain:

$$\pi_1 \circ (\pi_2 \bullet \pi_3) = 0 \quad \text{and}$$

$$(\pi_1 \circ \pi_2) \bullet (\pi_1 \circ \pi_3) = 0 \bullet 0 = \max(0) = 0.$$

$$\text{Therefore, } \pi_1 \circ (\pi_2 \bullet \pi_3) = (\pi_1 \circ \pi_2) \bullet (\pi_1 \circ \pi_3).$$

The lattice $(\Pi(S), \cdot, +)$ of partitions and the lattice $(MC(S), \circ, \bullet)$ of maximal coverings of S and their behaviour with respect to distributivity are shown in Figure 4.4.

Lemma 4.2 The structure $(\Pi(S), \circ, \bullet)$ is not a sublattice of the lattice $(MC(S), \circ, \bullet)$ of maximal coverings of S .

Proof. Clearly, $\Pi(S) \subset MC(S)$; however, $\Pi(S)$ is not closed under the operation " $\bullet$ " of the lattice $(MC(S), \circ, \bullet)$. As a counterexample consider the partitions $\pi_1, \pi_2 \in \Pi(S)$ of Example 4.1 :

$$\begin{aligned} \pi_1 \bullet \pi_2 &= \max(\pi_1 \cup \pi_2) = \\ &= \max(\{\{1,2\}, \{3\}, \{1\}, \{2,3\}\}) = \\ &= \{\{1,2\}, \{2,3\}\}. \end{aligned}$$

Clearly, $\pi_1 \bullet \pi_2 \notin \Pi(S)$, Q.E.D.

$$(\Pi(S), \cdot, +)$$

$$0 = \{1, 2, 3\}$$

$$\pi_1 \cdot (\pi_2 + \pi_3) = \pi_1 \cdot 1 = \pi_1$$

$$(\pi_1 \cdot \pi_2) + (\pi_2 \cdot \pi_3) = 0 + 0 = 0$$

The lattice $(\Pi(S), \cdot, +)$ is not distributive.

$$\pi_1 \oplus \pi_2 = \{12, 23\} \notin \Pi(S)$$

; therefore, $\Pi(S)$ is not a sublattice of $(MC(S), \circ, \bullet)$.

$$MC(S), \circ, \bullet$$

$$0 = \{1, 2, 3\}$$

$$\pi_1 \circ (\pi_2 \bullet \pi_3) = 0$$

$$(\pi_1 \bullet \pi_2) \bullet (\pi_2 \circ \pi_3) = 0 \bullet 0 = 0$$

The lattice $(MC(S), \circ, \bullet)$ is distributive.

Figure 4.4 The lattices of partitions and maximal coverings of S .

4.4 Interpretation of the structure of coverings in the knowledge context

We have seen how a covering of a data base state space S can represent knowledge about the data base.

A generalized view of the data base can be represented by a covering of S ; thus, we can compare the knowledge about the data base (information content) carried by two generalized views of the data base, via the refinement relation on the set of coverings of S . However, this relation is not a partial order unless we restrict ourselves to the set $MC(S)$ of maximal coverings of S . We overcome this problem by establishing the following theorem:

Theorem 4.2 For any two coverings of S , $\sigma_1, \sigma_2 \in C(S)$,
 σ_1 refines σ_2 iff $\max(\sigma_1)$ refines $\max(\sigma_2)$.

Proof. Suppose σ_1 refines σ_2 . Let B_1 be a maximal element of σ_1 . Then there exists $B_2 \in \sigma_2$ such that $B_1 \subseteq B_2$. Let B'_2 be a maximal element of σ_2 such that $B_2 \subseteq B'_2$; it follows that $B_1 \subseteq B'_2$; therefore, $\forall B_1 \in \max(\sigma_1)$ there exists $B'_2 \in \max(\sigma_2)$ such that $B_1 \subseteq B'_2$; thus,
 $\max(\sigma_1) \leq \max(\sigma_2)$.

Suppose now that $\max(\sigma_1) \leq \max(\sigma_2)$. Let $B_1 \in \sigma_1$; then there exists $B'_1 \in \max(\sigma_1)$ such that $B_1 \subseteq B'_1$. Also there exists $B_2 \in \max(\sigma_2)$ such that $B'_1 \subseteq B_2$. As $\max(\sigma_2) \subseteq \sigma_2$ we see

that $\forall B_1 \in \sigma_1$ there exists $B_2 \in \sigma_2$ such that $B_1 \subset B_2$.
Therefore, σ_1 refines σ_2 , Q.E.D.

The refinement partial order established in the lattice $MC(S)$ of maximal coverings of S provides a natural hierarchy of knowledge (information content) about the data base, contained in the generalized data base views.

Definition 4.6 If the coverings σ_1, σ_2 of a data base state space S represent the generalized data base views v_1 and v_2 , and $\max(\sigma_1) \leq \max(\sigma_2)$ (or, equivalently, if $\sigma_1 \leq \sigma_2$), we say that v_1 is at least as informative as v_2 , or we say that v_1 contains at least as much knowledge as v_2 .

Suppose σ_1, σ_2 are two coverings of S representing two generalized data base views v_1 and v_2 . We may assume that they are maximal coverings of S . The covering σ_1 represents the knowledge of the data base state when we only know the state of the view v_1 . Similarly, σ_2 represents the knowledge of the data base state when we only know the state of view v_2 . Intuitively, the product $\sigma_1 \circ \sigma_2$ represents our knowledge of the data base state when we know the states of both views, v_1 and v_2 . Therefore, the product " $\circ$ " "adds" (increases) the knowledge. The sum $\sigma_1 \bullet \sigma_2$ represents our knowledge of the data base state when we know the state of v_1 or v_2 . Therefore, the sum " $\bullet$ " "adds" (increases) ignorance of the data base state.

Example 4.3 Let us consider the two partitions π_{01} and π_{02} of the data base state space of Figure 4.1; their product and their sum, $(\bullet)$, are shown pictorially on Figure 4.5. We have :

$$\pi_{01} = \{ \{ 1,2 \}, \{ 3,4 \} \} , \quad \pi_{02} = \{ \{ 1,3 \}, \{ 2,4 \} \}$$

$$\pi_{01} \cdot \pi_{02} = \{ \{ 1 \}, \{ 2 \}, \{ 3 \}, \{ 4 \} \} = 0$$

$$\begin{aligned} \pi_{01} \bullet \pi_{02} &= \max(\pi_{01} \cup \pi_{02}) = \\ &= \{ \{ 1,2 \}, \{ 3,4 \}, \{ 1,3 \}, \{ 2,4 \} \} . \end{aligned}$$

The results correspond to intuition; in the case of the product knowing whether each employee (John and Peter) is present "adds" to (increases) our knowledge about the data base state; and as in fact $\pi_{01} \circ \pi_{02} = 0$, we have complete knowledge of the data base state; in the case of the sum knowing about the presence of John or Peter "adds" (increases) our ignorance of the data base state.

In the following sections we discuss some central data base concepts, namely, view complement, incomplete information, protection of information and independence, using the lattice of maximal coverings of the data base state space.

Figure 4.5 Addition of knowledge $\pi_{01} \cdot \pi_{02}$ and addition of ignorance $\pi_{01} \odot \pi_{02}$.

4.5 View complements

An important concept in data bases is the view complement.

Two views are complementary if together they provide complete knowledge of the data base. Consider for example the view Q_1 in Figure 4.1. The view Q_2 is a complement of Q_1 as the two views together allow us to know the data base. Let us recall from CHAPTER 3 that addition of knowledge (information content) of two views is represented by partition product; by a similar argument, addition of information content of generalized views is represented by product of maximal coverings.

Definition 4.7 . Two partitions $\pi_1, \pi_2 \in \Pi(S)$ of the data base state space are complementary iff $\pi_1 \cdot \pi_2 = 0$.

Example 4.4 Referring to Figure 4.1, we observe that $\pi_{01} \cdot \pi_{02} = 0$ and $\pi_{01} \cdot \pi_{03} = 0$. Therefore, π_{01}, π_{02} are complementary; also, π_{01}, π_{03} are complementary . We also say that the views Q_1, Q_2 are complementary; similarly, the views Q_1, Q_3 are complementary.

Definition 4.8 Two maximal coverings σ, τ of S are complementary iff $\sigma \circ \tau = 0$.

Lemma 4.3 For any coverings σ, τ of S ,
 $\max(\sigma) \circ \max(\tau) = 0$ iff $\sigma \circ \tau = 0$.

Proof. It is easy to see that

$$\begin{aligned} \sigma \circ \tau &= \max(\{A \cap B \mid A \in \sigma, B \in \tau\}) = \\ &= \max(\{A \cap B \mid A \in \max(\sigma), B \in \max(\tau)\}) = \\ &= \max(\sigma) \circ \max(\tau) . \end{aligned}$$

Thus $\max(\sigma) \circ \max(\tau) = \sigma \circ \tau$, Q.E.D.

The concept of view complement has been used as a means to translate updates on views back to the underlying data base [7] . The view complement provides the semantics missing from the view in order to correctly translate view updates. It is shown in [7] that the less informative the complement , the more view updates we can translate; hence the need to use the largest possible complement. However, in the case of partitions, only the trivial partitions, 0 and 1 , have a supremum complement. For example, in Figure 4.1; π_{02} is a maximal complement of π_{01} ; also π_{03} is a complement of π_{01} but it is not comparable to π_{02} ; therefore, there exists a maximal complement of π_{01} that is refined by π_{02} ; in fact it is the partition:

$$\pi = \{ \{ 2, 3 \}, \{ 1, 4 \} \} .$$

Clearly, π_{01} has no supremum complement. However, every maximal covering has a supremum complementary maximal covering, as is shown in the following theorem :

Theorem 4.3 Every maximal covering of S has a supremum complementary maximal covering.

Proof. Let $\sigma \in MC(S)$. Clearly :

$$\begin{aligned} \sigma \circ 0 &= \max(\{ A \cap \{ s \} / A \in \sigma, s \in S \}) = \\ &= \max(\{ \{ s \} / s \in S \}) = \{ \{ s \} / s \in S \} = 0 . \end{aligned}$$

Therefore, 0 is a maximal covering complementary to σ .

Let K denote the set of all maximal coverings of S that are complementary to σ . As 0 belongs to K , the latter is not empty. Define τ as follows:

$$\tau = \sum_{q \in K} q ,$$

where Σ denotes summation in the sense of $\bullet$.

Clearly, $\tau \in MC(S)$.

By the distributivity of the lattice $(MC(S), \circ, \bullet)$, we obtain :

$$\sigma \circ \tau = \sigma \circ \left(\bigcup_{q \in K} q \right) = \bigcup_{q \in K} (\sigma \circ q)$$

As every $q \in K$ is complementary to σ , $\sigma \circ q = 0$, and therefore:

$$\sigma \circ \tau = 0$$

It follows that τ is a maximal covering of S , complementary to σ . Suppose now that ϕ is a maximal covering of S , complementary to σ . Then $\phi \in K$. We shall show that $\phi \leq \tau$. Let $B \in \phi$.

As $\tau = \bigcup_{q \in K} q$ and $\phi \in K$, there exists $B' \in \tau$

such that $B \subset B'$. It follows that $\phi \leq \tau$. Therefore, τ is a maximal covering of S which is the supremum complement of σ , Q.E.D.

We conclude this section with an example of supremum maximal complement.

Example 4.5 Consider a data base consisting of a single relation

$R(X : D_1 ; Y : D_2)$

and suppose that $D_1 = D_2 = \{ 0,1 \}$, and that the

functional dependency $X \rightarrow Y$ holds.

In Figure 4.6, we see the set S of nine states of this data base partitioned into four blocks by a view σ derived from projection on the attribute X . It is easy to see that the supremum maximal covering of S , complementary to σ , is the following:

$$\tau = \{ \{ a_1, a_1, a_3, a_4 \} / a_1 \in B_1, a_2 \in B_2, a_3 \in B_3, a_4 \in B_4 \}$$

An example of a maximal covering τ' of S which is complementary to σ , but is not the supremum complement, is the following:

Consider the following four mappings f_1, f_2, f_3, f_4 from D_1 into D_2 :

$$f_1 = \{ (0, 0), (1, 0) \},$$

$$f_2 = \{ (0, 1), (1, 1) \},$$

$$f_3 = \{ (0, 0), (1, 1) \},$$

$$f_4 = \{ (0, 1), (1, 0) \}.$$

$$\sigma = \{ B_1, B_2, B_3, B_4 \}$$

$$\tau = \{ \{ a_1, a_2, a_3, a_4 \} \mid a_1 \in B_1, a_2 \in B_2, a_3 \in B_3, a_4 \in B_4 \}$$

Figure 4.6 Example of a supremum maximal covering τ , complementary to a view σ

Define now the covering τ' of S as follows :

$$\tau' = \{ B_i' = \bigcup_{j=1}^4 \{ b \in B_j \mid b \subset f_i \} \mid i = 1, \dots, 4 \} ;$$

in other words, for $i = 1, \dots, 4$, the block B_i' of τ' is defined as follows: for $j = 1, \dots, 4$, a member b of B_j is in B_i' iff $b \subset f_i$. Thus we obtain :

$$B_1' = \{ \emptyset, \{ (0,0) \}, \{ (1,0) \}, \{ (0,0), (1,0) \} \} ,$$

$$B_2' = \{ \emptyset, \{ (0,1) \}, \{ (1,1) \}, \{ (0,1), (1,1) \} \} ,$$

$$B_3' = \{ \emptyset, \{ (0,0) \}, \{ (1,1) \}, \{ (0,0), (1,1) \} \} ,$$

$$B_4' = \{ \emptyset, \{ (0,1) \}, \{ (1,0) \}, \{ (0,1), (1,0) \} \} .$$

Clearly, $\tau' \neq \tau$ and τ is a maximal covering of S . Also τ' refines τ and therefore τ' is more informative than τ .

Intuitively, this is so because each block of τ' contains relations that satisfy the functional dependency $X \rightarrow Y$. This does not happen in τ .

4.6 Incomplete information

Incomplete information can be considered at two levels. First, according to the so - called " open world assumption " [25] , the information stored in a data base is simply what we know about the real world. So, at any particular moment, the data base information is simply a " subset " of the real world information. Secondly, supposing that the data base contains complete information about the real world, according to the " closed world assumption " [25] , we may want to " hide " part of that information from a particular user (protection of information). What the user gets then is incomplete information about the data base.

4.6.1 Incomplete information under the "closed world assumption". Protection of information.

The case where we want to protect (hide) information can be described in terms of allowable and forbidden queries. Let us see an example.

• Example 4.6 Suppose that the data base consists of a single relation :

$R(\text{PROJECT} : \{ X \} ; \text{BUDGET} : \{ 100K , 1000K \} ; \text{STATUS} : \{ C , NC \})$

In other words, we may have only one project, named X , whose budget can be either 100K, or 1000K and which can either be classified, (C), or not classified, (NC), as shown in Figure 4.7.

The information to be protected (hidden) is the budget of classified projects. Let us consider a partition π^* of the data base state space S in which all classified states form a block and each non - classified state forms a singleton block. The data base view Q^* that corresponds to π^* lumps together the states 2 and 3 so that in view Q^* the information about the budget of classified projects is merely the one given by the definition of the domain of BUDGET, i.e., that the possible budget values are 100K and 1000K. The information in Q^* , called the information " secrecy level ", is the maximum amount of information we are willing to disclose to

π^* refines π_a ; Q_a is an allowable query

π^* does not refine π_b ; Q_b is not an allowable query

Figure 4.7 Protection of information at secrecy level π^*

the general user who is interrogating the data base via a set A of allowable queries. We wish to establish the set A of allowable queries. It is natural to require that any query - be it a partition or a covering of S - that is refined by Q^* , be in A . On the other hand, the fact that the user can take the product $\circ$ of allowable queries, implies that the set A of allowable queries must be closed under product. The following theorem shows that the set of queries which are refined by Q^* is indeed closed under product.

Theorem 4.4 The set $A' = \{ \sigma \in C(S) / \pi^* \leq \sigma \}$ of all coverings of S which are refined by a given partition π^* of S is closed under product.

Proof. Let $D = \{ \sigma_j \in C(S) \text{ such that } \pi^* \leq \sigma_j / j \in J \}$ and let

$$\tau = \prod_{\sigma \in D} \sigma$$

where Π denotes covering product $\circ$.

Let $B \in \pi^*$; as $\pi \leq \sigma_j$, $\forall j \in J$, there exists $B_j \in \sigma_j$ such that $B \subset B_j \forall j \in J$; it follows that

$$B \subset \bigcap_{j \in J} B_j$$

It follows that $\pi^* \leq \tau$ and therefore $\tau \in D$, Q.E.D.

Clearly, $A' \subseteq A$. The following lemma shows that actually the set A' of queries that are refined by Q^* , is equal to the set A of all allowable queries.

Lemma 4.4 For any partition $\pi^* \in \Pi(S)$ and any covering $\sigma \in C(S)$, if π^* does not refine σ , then $\sigma \circ \pi^*$ strictly refines π^* .

Proof. By our hypothesis, there exists $B \in \pi^*$ such that B is not a subset of any block of σ . Let $s \in B$ and let B' be a maximal block of σ that contains s . Clearly, $B \cap B'$ is a block of $\sigma \circ \pi^*$ and it is strictly a subset of B . On the other hand, every block of $\sigma \circ \pi^*$ is a subset of a block of π^* , and thus $\sigma \circ \pi^*$ strictly refines π^* , Q.E.D.

In conclusion, in order to protect the information at secrecy level π^* , we should allow only the set of queries each of which is refined by π^* .

Referring back to Example 4.6 (Figure 4.7) we see that hiding the budget of classified projects means that the information content of view Q^* , represented by the partition π^* , must be at least equal to the "cumulative" information content of the set A of allowable queries.

Consider e.g., the following queries:

Q_a : Is X classified ?

Q_b : What is the budget of X ?

In Figure 4.7 we show the data base state space S , consisting of four states and the partitions π^* , π_a , π_b , representing the view Q^* and the queries Q_a , Q_b , respectively. Let us see whether Q_a and Q_b are allowable queries.

Query Q_a is allowable, $Q_a \in A$, because $\pi^* \leq \pi_a$; thus, by answering query Q_a we do not reveal any secret specified by Q^* .

Query Q_b , however, is not allowable, $Q_b \notin A$, as $\pi^* \not\leq \pi_b$; indeed the block $\{2,3\}$ of π^* is not a subset of any block of π_b .

It should be noted that if the user knows only the answer to query Q_b , the secret specified in Q^* is not yet revealed, since the user knows only the budget of the current state. However, when he combines with this information the answer to some other

appropriate allowable query, Q_a for example, then he will know the secret information. In fact, in this particular case he will know completely the data base because $\pi_a \circ \pi_b = 0$.

4.6.2 Incomplete information under the "open world assumption"

Let us consider now the "open world assumption", i.e., incomplete information about the real world. We represent below a data base B with incomplete information about the real world W and a query Q against W .

According to the open world assumption, any superset of the information stored in B is a possible instance of W . Therefore, B can be represented as a view of a data base with data base state space W . In general, B will be a generalized view represented by a covering of W . The query Q is actually intended to be

a query against W , but the answer we obtain is not exactly what we wanted because it comes, in general, from the information stored in B . In other words, the answer to the query is actually Q / B , represented by a covering of W .

We do not intend to go into detailed discussion here. Our intention in this section was to suggest that the lattice of maximal coverings presented in section 4.3 could serve as a formal basis for the study of incomplete information.

We conclude this chapter by presenting in the next section a definition of independence using coverings.

4.7 Independence

The concept of independence has received considerable attention in the field of data bases [27,12,6]. Yet it has been discussed exclusively in the context of the relational model. In this section we present a model - independent definition of independence, based only on information - theoretic considerations.

Referring back to Figure 4.1, we see intuitively that the views Q_1 and Q_2 are independent because knowing whether "Peter" is in the data base tells us nothing about "John". Let us try to explain why this is so, by examining the partitions π_{01} and π_{02} that represent Q_1 and Q_2 (Figure 4.1(b)). We observe that the block $\{1,2\}$ of π_{01} intersects both blocks of π_{02} and this is true also for the other block $\{3,4\}$ of π_{01} . This implies that the coverings σ_{21}, σ_{12} that represent the knowledge Q_2 / Q_1 and Q_1 / Q_2 are both equal to 1. Indeed we obtain :

$$\sigma_{21} = \{ \cup \{ B \in \pi_{02} / B \cap A \neq \emptyset \} / A \in \pi_{01} \} = \{ S \} = 1,$$

$$\sigma_{12} = \{ \cup \{ B \in \pi_{01} / B \cap A \neq \emptyset \} / A \in \pi_{02} \} = \{ S \} = 1.$$

This means that given Q_1 , we know nothing about Q_2 and given Q_2 , we know nothing about Q_1 . In other words, no matter what answer to the one query, all answers to the other query are possible, i.e., our knowledge of the latter is not modified. This remark motivates

the following definition of independence :

Definition 4.9 Two coverings σ_1, σ_2 of S are independent, denoted $\sigma_1 \text{ indep } \sigma_2$, iff $\forall B_1 \in \sigma_1$ and $\forall B_2 \in \sigma_2$, $B_1 \cap B_2 \neq \emptyset$.

In other words, two coverings of a set are independent iff each block of one intersects all the blocks of the other.

Intuitively, two data base views are independent if knowledge of either one tells us nothing about the other. In formal terms, intuition suggests that two data base views Q_1, Q_2 , represented by the coverings σ_1, σ_2 , are "independent" iff $\sigma_{21} = \sigma_{12} = 1$. The question that arises then is the following: Is the formal definition of independence (Definition 4.9), equivalent to the "intuitive definition" ?

In the following we show that , in general, the two definitions are not equivalent and we establish conditions under which the two definitions become equivalent.

Theorem 4.5 For any coverings σ_1, σ_2 of S , $\sigma_1 \text{ indep } \sigma_2$ implies $\sigma_{21} = \sigma_{12} = 1$, but the converse, in general, does not hold.

Proof. Suppose $\sigma_1 \text{ indep } \sigma_2$. It follows that

$\forall B_1 \in \sigma_1, \forall B_2 \in \sigma_2, B_1 \cap B_2 \neq \emptyset$. Therefore,

$$\sigma_{21} = \{ u \{ B_2 \in \sigma_2 / B_2 \cap B_1 \neq \emptyset \} / B_1 \in \sigma_1 \} =$$

$$= \{ u \{ B_2 \in \sigma_2 \} / B_1 \in \sigma_1 \} = \{ S / B_1 \in \sigma_1 \} = \{ S \} = 1.$$

Similarly, we can show that $\sigma_{12} = 1$. It follows that :

$$\sigma_1 \text{ indep } \sigma_2 \quad \text{implies} \quad \sigma_{21} = \sigma_{12} = 1.$$

We show next that the converse does not hold, in general, by a counterexample (Figure 4.8). Let us consider the data base state space S and its coverings σ_1, σ_2 , defined as follows:

$$S = \{ x, y, z, w \},$$

$$\sigma_1 = \{ B_{11}, B_{12}, B_{13}, B_{14} \},$$

$$\sigma_2 = \{ B_{21}, B_{22}, B_{23} \},$$

where

$$B_{11} = \{ x, y \},$$

$$B_{12} = \{ y, w \},$$

$$B_{13} = \{ z, w \},$$

Counterexample

$\sigma_1 = \{B_{11}, B_{12}, B_{13}, B_{14}\}$, $\sigma_2 = \{B_{21}, B_{22}, B_{23}\}$, where:

$B_{11} = \{x, y\}$, $B_{12} = \{y, z\}$, $B_{13} = \{z, w\}$, $B_{14} = \{x, w\}$,

$B_{21} = \{x, z\}$, $B_{22} = \{y, w\}$, $B_{23} = \{x, w\}$.

$\sigma_{21} = \sigma_{12} = 1$, but $B_{11} \cap B_{22} = \emptyset$; therefore, σ_1, σ_2 are not independent.

Figure 4.8 $\sigma_{21} = \sigma_{12} = 1$ does not imply independence in $MC(S)$.

$$B_{14} = \{x, z\},$$

$$B_{21} = \{y, z\},$$

$$B_{22} = \{z, w\},$$

$$B_{23} = \{w, x\}.$$

It follows that :

$$\sigma_{21} = \{ \cup \{ B_2 \in \sigma_2 / B_2 \cap B_1 \neq \emptyset \} / B_1 \in \sigma_1 \} =$$

$$= \{ \cup \{ B_{21}, B_{23} \}, \cup \{ B_{21}, B_{22}, B_{23} \} \} =$$

$$= \{ \{x, y, z, w\} \} = \{S\} = 1,$$

$$\sigma_{12} = \{ \cup \{ B_1 \in \sigma_1 / B_1 \cap B_2 \neq \emptyset \} / B_2 \in \sigma_2 \} =$$

$$= \{ \cup \{ B_{11}, B_{12}, B_{13}, B_{14} \}, \cup \{ B_{12}, B_{13}, B_{14} \} \} =$$

$$= \{ \{x, y, z, w\} \} = \{S\} = 1.$$

Thus, $\sigma_{21} = \sigma_{12} = 1$.

However, $B_{11} \in \sigma_1$ and $B_{22} \in \sigma_2$ are such that, $B_{11} \cap B_{22} = \emptyset$.

Therefore, σ_1 and σ_2 are not independent, Q.E.D.

Theorem 4.5 suggests that not all coverings are "good" for knowledge representation, in the sense that they may not agree with our intuitive notion of independence. It should be noted also that even if we restrict ourselves to the set $MC(S)$ of maximal coverings of S , the converse of the above theorem does not hold, in general. This is clear, as the coverings σ_1 and σ_2 of the above counterexample are in fact maximal coverings of S .

We will show in the sequel, however, that there exists a finer class of coverings, for all members of which the converse part of Theorem 4.5 does hold.

Definition 4.10 A covering σ of a set S is called a reduced covering iff $\forall B \in \sigma, \sigma - \{B\}$ is not a covering of S . We denote by $RC(S)$ the set of reduced coverings of S .

In the above counterexample (Figure 4.8), the coverings σ_1 and σ_2 are not reduced, whereas the following coverings are reduced:

$$\sigma_1 = \{B_{11}, B_{13}\}, \quad \sigma_1 = \{B_{12}, B_{14}\},$$

$$\sigma_2 = \{B_{22}\}.$$

Lemma 4.5 $RC(S) \subset MC(S)$. In other words, every reduced covering of a set S is also a maximal covering of S , but not all maximal coverings of a set S are reduced.

Proof Let σ be a reduced covering of S . Suppose σ is not a maximal covering of S . Then there exist $B, B' \in \sigma$ such that $B \subset B'$. But then $\sigma - \{B\}$ is a covering of S , contradicting the fact that σ is reduced. Therefore, σ is a maximal covering of S .

We show that not all maximal coverings of S are reduced by a counterexample (Figure 4.9). Consider the data base state space S and its coverings σ_1, σ_2 , defined as follows:

$$S = \{1, 2, 3, 4\},$$

$$\sigma_1 = \{ \{1, 2\}, \{2, 3\}, \{3, 4\} \},$$

$$\sigma_2 = \{ \{1, 2\}, \{3, 4\} \}.$$

Clearly, σ_1 is a maximal covering of S that is not reduced; indeed, $\sigma_2 = \sigma_1 - \{ \{2, 3\} \}$ is a covering of S (and, in fact, it is a reduced one), Q.E.D.

Theorem 4.6 . $\forall \sigma_1, \sigma_2 \in RC(S)$, σ_1, σ_2 are independent iff $\sigma_{21} = \sigma_{12} = 1$. In other words, for any reduced coverings σ_1, σ_2 of a set S , σ_1 indep σ_2 iff $\sigma_{21} = \sigma_{12} = 1$.

Proof. Let $\sigma_1, \sigma_2 \in RC(S)$ be independent. Then by Theorem 4.5, it follows that $\sigma_{21} = \sigma_{12} = 1$.

Data base state space A maximal, but not reduced covering of S A reduced covering of S

Figure 4.9 Counterexample : Not all maximal coverings are reduced , $RC(S) \subsetneq MC(S)$

Now, let $\sigma_1, \sigma_2 \in RC(S)$ be such that $\sigma_{21} = \sigma_{12} = 1$.

Suppose σ_1, σ_2 are not independent. Then there exist $B_1 \in \sigma_1$ and $B_2 \in \sigma_2$ such that $B_1 \cap B_2 = \emptyset$. As σ_1 is reduced,

there exists $x \in B_1$ such that $x \notin \cup(\sigma_1 - \{B_1\})$;

otherwise $\sigma_1 - \{B_1\}$ would be a covering of S and so σ_1 would not be a reduced covering of S .

We have: $\sigma_{12} = \{ \cup\{B_i \in \sigma_1 / B_i \cap B_j \neq \emptyset\} / B_j \in \sigma_2 \}$.

To each $B_j \in \sigma_2$ there corresponds a block $B'_j \in \sigma_{12}$, where

$$B'_j = \cup\{B_i \in \sigma_1 / B_i \cap B_j \neq \emptyset\}.$$

So, corresponding to $B_2 \in \sigma_2$, let us calculate $B'_2 \in \sigma_{12}$.

As $B_1 \cap B_2 = \emptyset$, we obtain:

$$B'_2 = \cup\{B_i \in \sigma_1 / B_i \cap B_2 \neq \emptyset\} =$$

$$= \cup\{B_i \in (\sigma_1 - \{B_1\}) / B_i \cap B_2 \neq \emptyset\} \subset$$

$$\subset \cup\{B_i \in (\sigma_1 - \{B_1\})\}.$$

Thus, we obtain: $B'_2 \subset \cup(\sigma_1 - \{B_1\})$.

As σ_1 is a reduced covering of S , the set $\cup(\sigma_1 - \{B_1\})$ is a proper subset of S ; it follows that the block B'_2 of

σ_{12} is different from S ; therefore, $\sigma_{12} \neq 1$,

contradicting our hypothesis. It follows that the coverings

σ_1, σ_2 are independent, Q.E.D.

Let us note that in fact we have proved a stonger version of the converse of Theorem 4.6 ; namely we have proved the following :

If $\sigma_{21} = 1$ or $\sigma_{12} = 1$, then σ_1 indep σ_2 .

Finally, let us note that reduced coverings are non - redundant, in the sense that they contain just enough blocks to cover the data base state space. It is precisely this feature that makes reduced coverings behave well with respect to our notion of independence.

4.8 Conclusion

An analysis of some information - theoretic aspects of data bases has been carried out.

We extended the data base mapping to a more general carrier of information, the relation on two state spaces. We introduced the state space covering to represent the information content of a relational information carrier. We have explained how a covering represents knowledge and we used coverings to represent incomplete information. We presented an algebra that allows us to "add" the knowledge or the ignorance contained in two coverings. Having established this framework of analysis we applied our theory to some information - theoretic aspects of data bases related to generalized mappings, such as view complement, incomplete information, protection of information and independence.

The basic mathematical tool throughout this analysis is the covering of a data base state space.

The advantage of the information content representation by coverings is that both complete and incomplete information can be treated in a uniform way.

We believe that the presented approach, further elaborated, can provide a basis for information - theoretic considerations in data bases.

CHAPTER 5

CONCLUDING DISCUSSION

The purpose of this chapter is to summarize and relate the salient features of the foregoing chapters, to comment on the significance and impact of the results, and to provide suggestions for further research.

5.1 Summary and conclusions

In this thesis we have exposed a model-independent approach to data bases, using only update operations.

We summarize the basic ingredients of this approach.

Given are :

A set of values T that we have called the data base "type".

A set U of unary operations on T , i.e., a set of mappings from T into T .

We have seen that a partition π of T can be defined by a set of operations on T that we termed a definition set for π . In general a partition π has more than one definition set.

We chose to associate with each partition π of T , a well-defined definition set, namely the union of all its definition sets, which is also a definition set of π , that we termed a "saturated" set of updates.

We discovered the existence of a double semilattice isomorphism that links the structures of :

The set I_T of all saturated sets of operations on T .

The set $\Pi(T)$ of all partitions of T .

The set $F_T/\equiv$ of all equivalence classes of mappings on T

(two mappings being equivalent iff they induce equal partitions on T).

Consequently, we have "replaced" equivalence classes of data base mappings by saturated sets of updates and we have established the structure of view information content to be either one of the isomorphic semilattices I_T , $\Pi(T)$.

We have then carried out an important application of our approach, to lossless decompositions; namely, we have derived a model-independent criterion for a decomposition to be lossless, based only on properties of saturated sets of updates. We have seen that our criterion applies to any kind of decomposition and, consequently, it accounts for the known results on lossless decompositions that have appeared in the literature.

On the other hand, we have discussed the structure of a particular class of saturated sets of operations on T which can replace the family of all integrity constraints that can be defined on a given data base type T .

The model-independent approach is then extended to include generalized mappings. Extending the concept of a data base mapping, we studied a more general information carrier, the data base relation, i.e., a binary relation between two state spaces.

In much the same way as the induced state space partition represents the information content of a data base mapping, we have introduced the induced state space covering- a "generalized partition"- to represent the information content of a relation.

We have explained how a covering of a state space represents knowledge of the data base or part thereof, and we have used coverings to represent incomplete information.

We have presented an algebra that allows us to "add" the knowledge or the ignorance contained in two coverings.

Then we have applied our theory to some information-theoretic aspects of data bases related to generalized mappings, such as view complement, incomplete information, protection of information and independence.

The basic mathematical tool throughout this analysis is the covering of a data base state space.

The advantage of the information content representation by coverings is that both complete and incomplete information can be treated in a uniform way.

In conclusion, we have provided a model-independent approach to data bases, where integrity constraints, data base transformations, view information content, lossless decomposition are all treated in terms of updates only; and we have seen that it is possible to study some important data base problems without reference to a particular model; finally, we have extended our approach to represent the information content of generalized mappings in model-independent terms through the state space coverings; and we have seen that it is possible to treat some important problems such as incomplete information, protection of information, view complement and independence in a model-independent framework through state space coverings.

5.2 Suggestions for further research

There are some important aspects of our approach that, we believe, can be further investigated.

In what follows, we describe briefly these aspects and we give suggestions and possible directions for further research.

The first aspect concerns incomplete information.

In CHAPTER 4 we extended our model-independent approach so that it accounts for incomplete information as well. We used coverings instead of partitions in order to capture the notion of incomplete information.

Coverings seem to be well adapted to this end, as we were able to define all important notions, such as views, independence, protection of information etc.

However, the concept of a covering has no real status in a data base context.

Therefore, we think that state space coverings should be replaced by some extended notion of a saturated definition set of updates, in much the same way as it was done for state space partitions.

Should the above idea succeed, the resulting approach would treat in a simple and uniform manner both complete and incomplete information in model-independent terms using only data base update operations.

The second aspect for further investigation concerns the transition from our model-independent approach to a specific data model approach.

Although it is often convenient to treat problems such as lossless decomposition from a model-independent point of view, it is nevertheless necessary to translate the results into specific data models.

What we need here is a way of going from names for data to operations on data and vice versa.

The situation is sketched roughly in the following diagram:

Using the fundamental isomorphism theorem, we have been able to convert partitions into operations by associating to each partition a set of operations. On the other hand, we have already seen how attribute names really correspond to partitions.

We could probably use these facts to create a direct association between attribute names and operations. After all, such an association is directly involved in the normalization process.

REFERENCES

- [1] ABITEBOUL S., SPYRATOS M., SPYRATOS N., " Information-theoretic aspects of data bases " , INRIA Reasearch Report , to appear.
- [2] ANSI / X3 / SPARC, the ANSI / X3 / SPARC framework, D. Tsichritzis and A. Klug eds., AFIPS Press, 1978.
- [3] ARMSTRONG W.W., " Dependency structures of data base relationships " , Proc. IFIP, North Holland, pp. 580 - 583, 1974.
- [4] ARMSTRONG W.W., " On the generation of dependency structures of relational data bases " , Publication Number 272, Département d' informatique et de recherche operationnelle , Université de Montréal , June 14 , 1977.
- [5] BANCILHON F., " On the completeness of query languages for relational data bases " , Proc. Seventh Symposium on Mathematical Foundations of Computer Science , Springer - Verlag , 1978.
- [6] BANCILHON F., SPYRATOS N., " Independent Components of data bases " , Proceedings VLDB 7 , Cannes , France , September 1981.

- [7] BANCILHON F., SPYRATOS N., " Update semantics of relational views " , ACM Trans. Data Base Systems , December 1981.

- [8] BEERI C., BERNSTEIN P., GOODMAN N., " A sophisticate' s introduction to data base normalization theory " , Proc. 4th Int. Conf. on Very Large Data Bases , West Berlin , pp. 113 - 124 , 1978.

- [9] BEERI C., MENDELZON A.O., SAGIV Y. and ULLMAN J.D., " Equivalence of relational data base schemes " , SIAM J. on Computing , 10 , pp. 352 - 370 , 1981.

- [10] BEERI C. and RISSANEN J., " Faithful Representation of Relational Data Base Schemes " , IBM Research Report RJ 2722 , 1980.

- [11] BERNSTEIN P.A., " Synthesizing third normal form relations from functional dependencies " , ACM Trans. Database Systems , 1 , pp. 277 - 298 , 1976.

- [12] BISKUP J., DAYAL U. and BERNSTEIN P.A., " Synthesizing independent data base schemas " , ACM / SIGMOD International Symposium on Management of Data , pp. 143 - 152 , 1979.

- [13] CHAMBERLAIN D.D., GRAY J.N. and TRAIGER D.D.,
" Views, authorization and locking in a relational
data base system " , Proceedings of AFIPS NCC,
Vol. 44 , 1975.
- [14] CHIN F.Y., " Security in statistical databases
for queries with small counts " , ACM Trans. on
Database Systems 3:1 , pp. 92 - 104 , 1978.
- [15] CODD E.F., " A relational model for large shared
data banks " , Comm. ACM 13:6 , pp. 377 - 387 ,
1970.
- [16] CODD E.F., " Further normalization of the data
base relational model " , Data Base Systems
(R. Rustin, ed.) , Prentice - Hall , Englewood
Cliffs , N.J. , pp. 33 - 64 , 1972.
- [17] CODD E.F., " Relational completeness of data
base sublanguages " , Data Base Systems (R. Rustin
ed.) , Prentice - Hall , Englewood Cliffs ,
N.J. , pp. 65 - 98 , 1972.
- [18] DAYAL U. and BERNSTEIN P.A., " On the updatability
of relational views " , Proc. of the 4th International
Conference on VLDB , West Berlin , 1978.

- [19] DELOBEL C., " Normalization and hierarchical dependencies in the relational data model " ,
TODS 3 : 3 , pp. 201 - 203 , 1978.

- [20] DEMILLO R.A., DOBKIN D.P., JONES A.K. and LIPTON R.J.,
" Foundations of Secure Computation " , Academic
Press , New York , 1978.

- [21] DENNING D.E., DENNING P.J. and SCHWARTZ M.D.,
" Securing data bases under linear queries " ,
IFIP Congress , North Holland , pp. 395 - 398 ,
1977.

- [22] FAGIN R., " Multivalued dependencies and a new
normal form for relational databases " , TODS 2 : 3 ,
pp. 262 - 278 , 1977.

- [23] FAGIN R., " Normal forms and relational database
operators " , Proceedings ACM - SIGMOD ,
International Conference on Management of Data ,
pp. 153 - 160 , 1979.

- [24] HARTMANIS J. and STEARNS R.E., " Algebraic
structure theory of sequential machines " ,
Prentice - Hall , Englewood Cliffs , N.J. , 1966.

- [25] IMIEILINSKI T. and LIPSKI W., " On representing incomplete information in relational databases " , Proceedings VLDB-7 , Cannes , France , 1981.
- [26] NICOLAS J.M., " Mutual dependencies and some results on indecomposable relations " , Proceedings ACM International Conference on Very Large Data Bases , pp. 360 - 367 , 1978. .
- [27] RISSANEN J., " Independent components of relations " , TODS 2 : 4 , pp. 317 - 325 , 1977.
- [28] SCHLORER J., " Confidentiality of statistical records : a threat monitoring scheme for on - line dialogue " , Methods of Information in Medicine 15 : 1 , pp. 36 - 42 , 1976.
- [29] SPYRATOS M., " A homomorphism theorem for data base mappings " , Information Processing Letters , North Holland , Accepted for publication , To appear in the upcoming issue , 1982.
- [30] SPYRATOS N., " An operational approach to data bases " , Proceedings on Principles of Database Systems , Los Angeles , March 1982.

- [31] ULLMANN J., " Principles of data base systems " ,
Computer Science Press , 1980.
- [32] YANNAKAKIS M. and PAPADIMITRIOU C.H., " Algebraic
Dependencies " , Proceedings of the 21st annual
IEEE Symposium on foundations of Computer Science ,
pp. 328 - 332 , 1980.
- [33] ZANIOLO C., " Relational views in a database
system support for queries " , Proceedings IEEE
COMPSAC ; 1977.

END

131108H84

FIN