MINISTRY OF EDUCATION AND TRAINING

VIETNAM ACADEMY OF SCIENCE AND TECHNOLOGY

GRADUATE UNIVERSITY OF SIENCE AND TECHNOLOGY

.....***.....**

NGUYEN THI VAN

EXTENDING THE NEW TYPE OF DEPENDENCIES CALLED APPROXIMATE GENERALIZED POSITIVE BOOLEAN DEPENDENCIES IN RELATIONAL DATABASES

SUMMARY OF DISSERTATION ON INFORMATION SYSTEM
Major code: 9 48 01 04

Ha Noi – 2023

The dissertation is completed at: Graduate University of Science and Technology, Vietnam Academy Science and Technology

Supervisors:

Supervisor : Assoc. Prof., D.Sc. Nguyen Xuan Huy

Referee **1**: …….
Referee **2**: …….
Referee **3**: …….

The dissertation will be examined by Examination Board of Graduate University of Science and Technology, Vietnam Academy of Science and Technology at ……. hour……, date…… month……year 2023

The dissertation can be found at:

1. Graduate University of Science and Technology Library
2. National Library Vietnam

INTRODUCTION

1. The ugency of the thesis

Research situation abroad::

- In 1970, Codd introduced the relational database model and the concept of functional dep3endency to reflect the semantics of real-world data.

- In 1983 J. Demetrovics and O. Gyepesi proposed dual dependence, strong dependence, weak dependence.

- From 1977 to 2003, R. Fagin and Zaniolo and several other groups of authors proposed multivalued dependence, multivalued dependence expands the value set to not only accept two values {0, 1}, which includes *k* real values in the interval [0, 1].

- In 1981 - 1985, the research groups of Berman, Blok and Sagiv, Delobel developed the concept of functional dependence to the concept of positive Boolean dependence, including data constraints that are described through positive Boolean formulas, but still keeping the equality comparison.

- In 1995 Jyrki Kivinen and colleagues proposed approximate functional dependence. The groups of Hultala Y. and colleagues, Ronald S. K. and Janes J. L. have further developed several algorithms for this type of approximate functional dependence.

- In 2004, Ilyas and his colleagues studied soft functional dependence, which is a functional dependence in which the value of X determines the value of Y with a given uncertainty.

- In 2007 Bohannon and his colleagues proposed conditional functional dependencies to clean data.

- In 2011, the research group Song S. and Chen L. proposed differential dependency to solve some problems such as ensuring integrity and query optimization.

- Currently, some new development directions on data dependence are being researched by groups such as comparative dependence (Song S., Chen L., and Yu P.S - 2013); analysis of constraints according to pattern structure (Baixeries J., Kaytoue M., and Napoli A. - 2015) extended approximate functional dependence.

- In 2016, authors Loredana Caruccio, Vincenzo Deufemia, Giuseppe Polese summarized 35 types of extended dependencies of research groups around the world called the group of relaxed functional dependencies.

Researches in our country:

- In Vietnam, some domestic research groups have expanded the types of dependencies to create tighter ties in the database: such as authors Dam Gia Manh; Vu Ngoc Loan, Bui Duc Minh, Nguyen Hoang Son, Luong Nguyen Hoang Hoa Le Xuan Vinh ; Truong Thi Thu Ha... investigated general positive Boolean classes, weak dependence, differential dependence, dual dependence, ... from many different angles.

- Nguyen Xuan Huy and Le Thi Thanh expanded positive Boolean dependence into general positive Boolean dependence, multivalued positive Boolean dependence, and positive Boolean dependence by tuple group.

Based on survey and analysis, the researcher obtained some of the following basic characteristics:

(1) Data dependencies can be described through logical clauses that reflect the correlation between attributes in the database.

(2)All data dependencies in a database are based on real-world perception, which attempts to represent the semantics of real-world data.

(3) Most of the results focus on basic concepts, typical properties, applications and important basic algorithms of database theory. Some modern, more in-depth research has recently appeared on combinatorial database theory such as closed sets, keys, anti-keys, relational schema transformation, families of minimal sets of attributes. Calculating and extending functional dependencies or finding equivalent descriptions of functional dependencies are also introduced.

The thesis continues to research and expand general positive Boolean dependence to

obtain a new form of dependence: approximate positive Boolean dependence, general approximate positive Boolean dependence and approximate weak dependence with the purpose of creating a general model for the above dependency classes by showing the correlation between types of dependencies, proposing a unified dependency class that covers the data dependency classes currently of interest in research and development. declare.

2. Objectives of the thesis

(1) Extending the new types of dependencies called Approximate Generalized Dependencies, Approximate Positive Boolean Dependencies , Approximate Generalized Positive Boolean Dependencies, and Approximate weak Dependencies

(2) Building the relationships between types of logical dependencies and relaxed positive Boolean dependencies: generalized positive Boolean dependencies, approximate positive Boolean dependencies.

(3) The thesis proposes a new type of logical dependencies that are broader than known dependencies. With this layer of logical dependence, the thesis has obtained the following results:

- Proposing a process for solving the derivation problem according to three formal approaches: direct proof according to Vuong Hao's algorithm, direct proof according to standard association and counterfactual proof according to fusion solution and new results on the application tautology proof methods.
- Build an algorithm to find closure of a set of attributes for a class of logical dependencies.
- Build an algorithm to find keys for a class of logical dependencies.

3. Research object and scope

- *The research object* of the thesis is the concepts and properties of logical dependencies: Positive Boolean dependence, general positive Boolean dependence, weak dependence, approximately positive Boolean dependence, approximately total positive Boolean dependence general… Relationships between logical dependencies. Classic problems in the theory of dependencies.

- *The scope of research* of the thesis is variations of functional dependencies, general positive Boolean dependencies, relationships between logical dependencies and methods of solving some classic problems in logical dependencies..

4. Research Methods

Methods of inference, interpretation, and formalization from research results to present concepts of basic logical dependencies have been studied sequentially: stating definitions, characteristics, problems, ...proven and used; Analyze, synthesize, and prove to give expected results.

5. Content of research

(1) Research variations of Boolean dependencies to obtain more general dependency classes and expand applications in database management and exploitation.

(2) Research comparisons between sets in the database. Proposing a quantitative Lambda function for attributes and applying the concept of measure in comparing sets of relations, obtaining new types of dependencies: approximate positive Boolean dependency, approximate general positive Boolean dependency, and approximate positive Boolean dependency. belonging to weak approximation.

(3) Propose the concept of approximate positive Boolean dependency, general approximate positive Boolean dependency in the relational data model, state and prove theorems, properties of approximate positive Boolean dependency and dependency General approximate positive Boolean, confirms the relationship between approximately positive Boolean dependence and general positive Boolean dependence, between general approximate positive Boolean dependence and general positive Boolean dependence.

(4) Proposing the concept of approximate weak dependence, stating and proving theorems, properties of Approximate Weak Dependency showing the relationship between approximate weak dependence and general positive Boolean dependence.

6. Scientific and practical significance

Scientific significance

Scientifically, the thesis has developed and expanded a number of data dependencies and built a class of logical dependencies with common properties and characteristics of dependent classes.

Practical significance

The results of the thesis are used as a basis to build a general model of different layers of dependencies in data and knowledge mining using AI and deep learning tools according to an overview of the relationship. between logical dependencies in the database. From there, it is possible to choose a theory of dependencies that is more suitable for designing a specific database that meets the frequently changing needs of practice, capable of supporting diverse applications. means to meet the needs of collecting, organizing, and managing databases.

7. Layout of the thesis

Chapter 1. Presents the background knowledge related to the thesis, specifically: Presents an overview of functional dependencies, concepts of relationships, properties, sets, equivalence theorems... and focuses on the presentation content on relaxed functional dependencies, approximate functional dependencies, and general positive boolean dependencies.

Chapter 2. Presenting the results of the researcher's own research on approximately positive Boolean dependence, approximately general positive Boolean dependence, and approximately weak dependence.

Chapter 3. Presents the derivation problem, closure algorithm, algorithm for finding keys in a relational schema, method to convert any logical formula to standard form, algorithms to prove constants and some Related results of NCS.

Conclude. Summary of achieved results, remaining points and future research directions.

CHAPTER 1. TYPES OF DATA DEPENDENCY IN DATABASES

1.1. Preliminary

The results of this chapter include:

(1) Presents the most important and quintessential issues related to the concepts of functional dependence, general positive Boolean dependence, approximate functional dependence, and relaxed functional dependence.

(2) Build dependent classes in the database based on two basic characteristics: derivation formula and value comparison.

(3) Survey and specification of different variants of generalized positive Boolean dependence. Find a specification for general relaxed dependence and show that relaxed dependence in general and relaxed functional dependence in particular are just special cases of general positive Boolean dependence.

1.2. Some concepts and conventions

Concepts of attributes, value domain, tuples and relationships:

Let $U = \{a_1, a_2, \ldots, a_n\}$, $n \geq 1$, *U is called the set of attributes.* Each element $a \in U$ *is the set of value domains* d_x. The symbol $\mathfrak{D}$ is the union of value domains d_x of the properties in U , $\mathfrak{D} = \bigcup_{a \in U} d_x$.

Relationship r with attribute set U, denoted $r(U)$, is a set of mappings t: $U \to \mathfrak{D}$ such that each attribute $a \in U$ then $t.a \in d_x$, in there $t.a$ is the image of the attribute a through mapping t. Each mapping t is called a tuple in the relation r.

- Value assignment e for variable x: $x := e$
- Notation $x := (e)\ ?\ a : b$,
- The value of attribute a is written as d_a.

The sets in relationship r are denoted $t, u, v, \ldots$ or with the index t_i, u_j, v_k. The value of attributes a in the set t is $t.a$, The value of the sub-practice of the X attributes in the t set is $t.X = \{t.a \mid a \in X\}$.

Union of the two sets are written $X \cup Y$; intersection is written $X \cap Y$; subtraction is written X-Y.

Union of the two the logical formula is denoted $X \wedge Y$ or XY or $X \cdot Y$; disjunctive: $X \vee Y$ or $X+Y$, denoted ' instead of negative $\neg$. Partition of the sets M (into sub-sets of each other) $X_1, X_2, \ldots, X_k$ denoted $M = X_1 \mid X_2 \mid \ldots \mid X_k$ with meaning $M = X_1 \cup X_2 \cup \ldots \cup X_k$ and $X_i \cap X_j = \varnothing$, $1 \leq i, j \leq k$, $i \neq j$.

1.2. Function dependent

For attributes U. A functional dependence on U is the expression

$$f: X \to Y\ ;\ X, Y \subseteq U$$

For relations r and a functional dependence f: $X \to Y$ on U. We say the relationship r is the function dependence f and written $r(f)$, If the two sets arbitrary in r are the same on X, they are the same on Y,

$$R(X \to Y) \Leftrightarrow (\forall u,v \in R)\colon (u.X = v.X) \Rightarrow (u.Y = v.Y)$$

Denoted $X \nrightarrow Y$ That is, the set of properties Y does not depend on the function of the X property.

For functional dependence F on U. We say the relationship r is the function dependence F,

$$R(F) \Leftrightarrow (\forall f \in F)\colon R(f)$$

Closure of the attribute

For functional dependence F on U and a sub-set of X properties in U. *Closure of property X*, denoted X^+, is the attribute $X^+ = \{A \in U \mid X \to A \in F^+$. *Some properties of closure*

For relationship schema a = (U,F). *Khi đó* $\forall\ X, Y \subseteq U$ *has following propercities:*

(1) *Reflexivity*: $X \subseteq X^+$

(2) *Uniformity*: $X \subseteq Y \Rightarrow X^+ \subseteq Y^+$

(3) *Tensilevity*: $(X^+)^+ = X^+$

Relationship schema

Relationship schema is one pair (U, F), U is a finite set of properties, F is the set of constraints.

Key of relationship schema

For relationship schema $a = (U, F)$. Set properties of $K \subseteq U$ are called *key* of the relationship schema a if

(i) $K^+ = U$

(ii) $\forall A \in K: (K - \{A\})^+ \neq U$

If K satisfy condition (i) (hoặc (i')) then K is called a super key.

For relationship schema $a = (U, F)$. We denoted U_K is the key attribute of a and U_0 is the set of non-key attributes of a. Implies that $U_K | U_o$ is a subdulation of U.

Theorem 1.1: *Equivalent theorem for function dependence*

For the set of function dependent F and a functional dependence f on U, The following three types of *consequence* are equivalent:

- $F \models f$ (*logic consequence*)
- $F \vdash f$ (*relation consequence*)
- $F \vdash_2 f$(*two tuple relation consequence*)

For relationship schema (U, F), F is a set of function dependence on the attribute U.

Closure of the function dependent, denoted F^+, is all of the function dependent on U logic consequenced from F. $F^+ = \{f \mid F \models f\}$

1.3. Relaxed functional dependencies

Relaxed functional dependencies are formula in a general form: $f: X(\lambda) \rightarrow Y(\gamma)$; $X, Y \subseteq U$ with relaxed conditions λ and γ as follows:

Relationship r satisfied Relaxed functional dependencies $f: X(\lambda) \rightarrow Y(\gamma)$; $X, Y \subseteq U$ if $T_r \subseteq T_f$.

Relax the comparisons on several properties: The relation r satisfies the relaxed functional dependence by value comparison $r(X\ (\rho) \rightarrow Y)$ if and only if any two tuples are in r The difference does not exceed the threshold ρ on X then the difference between those two sets is not more than a threshold ρ on Y.

General definition of *generalized relaxed functional* dependence:

Let U be a set of attributes, X and Y are two sets of attributes in U. Relaxed functional dependencies has the form:

$$f: X(\gamma) \rightarrow Y(\vartheta), X, Y \subseteq U$$

We say that functional dependence relaxes f satisfies in the relation r if:

$$f: X(\gamma) \rightarrow Y(\vartheta), X, Y \subseteq U$$

With every of sets $u, v \in r$, if neologism $\gamma(u.X, v.X)$ deduced neologism $\vartheta(u.X, v.Y)$.

Relaxed functional dependencies are functional dependencies with accompanying conditions to mitigate the conditions of formal functional dependence. Mitigating conditions are expressed through prepositions γ and ϑ.

1.4. Positive Boolean dependency

1.4.1. Boolean formula

Definition 1.1

Let $U = \{x_1,...,x_n\}$ is a finite set of variables *Boole*, $\mathcal{B}$ is a set of values *Boole*, $\mathcal{B} = \{0,1\}$.

Definition 1.2

Each vector has 0/1 elements, $v = (v_1,...,v_n)$ in space $\mathcal{B}^n = \mathcal{B}\times\mathcal{B}\times...\times\mathcal{B}$ is called a value assignment.

Then for each boolean formula $f \in L(U)$ we have $f(v) = f(v_1,...,v_n)$ is the value of the

formula f for the value assignment v, and $f(v)$ is calculated as follows:

1. Replace all variables x_i in f equals value v_i respectively, i=1,2,…, n to obtain a logical proposition b.

2. The value of $f(v)$ is the value of b.

For each subset $X \subseteq U$, We write conventions of association *logic* ($\wedge$) of variables in X is the symbol sequence of X. X also represents the following objects:

- *a set of attributes in U,*
- *a set of logical variables in U,*
- *A Boolean formula is formed by logical association of variables in X.*

We call formula f: $Z \to V$ is:

- *Consequence formula if Z and V have union form, i.e. f*: $\wedge Z \to \wedge V$
- *Strong consequence formula* if Z has the recruitment form and V has the union form, i.e: f: $\vee Z \to \wedge V$
- *Weak consequence formula* if Z has the assembly form and V has the disjunctive form, i.e: f: $\wedge Z \to \vee V$
- *Duality consequence formula* if Z and V both have a selective form, that is f: $\vee Z \to \vee V$

Two special value assignments are unit value assignment, e = □1,1,...,1) and zero value assignment , z = (0,0,...,0).

For each finite set of boolean formulas, $F = \{f_1, f_2,...,f_m\}$ in $L(U)$, F is a form formula $F = f_1 \wedge f_2 \wedge ... \wedge f_m$. Then for each value assignment v, The truth value of formula F is calculated as:

$$F(v) = f_1(v) \wedge f_2(v) \wedge ... \wedge f_m(v)$$

1.4.2. Value table and truth table

The concepts of value table and truth table are defined as follows:

- For each formula f on U, the value table of f. The truth table of f, denoted by T_f, is the set of value assignments v such that $f(v)$ a receives value 1, $T_f = \{v \in \mathscr{B}^n \mid f(v) = 1\}$

truth table T_F of the finite set of formulas F on U, is the intersection of the truth tables of each member formula in F

$$T_F = \bigcap_{f \in F} T_f$$

We have, $v \in T_F$ if and only if $\forall f \in F$: $f(v) = 1$.

1.5. Generalized positive Boolean dependence

- *Let* $U = \{x_1, ..., x_n\}$ is a finite set of Boolean variables that receive values in the logical value set $\mathfrak{B} = \{0, 1\}$.

- Convention for each value domain V_x of attribute x in U contains at least two elements. For each domain value V_x, consider mapping α_x: $V_x^2 \to$ satisfy the following axioms:

$\forall a, b \in V_x$

A1) Reflexivity $\alpha_x(a, a) = 1$

A2) Symmetry $\alpha_x(a, b) = \alpha_x(b, a)$

A3) Partiality $\exists c \in d_x$: $\alpha_x(a, c) = 0$.

α_x is the partial relation, satisfying the properties of reflection and symmetry on the value domain V_x.

Equal relationship $=_x$ is defined: $\forall a, b \in V_x$: $=_x(a, b) = 1$, if and only if $a = b$, *i*s a special case of value comparison and is implicit in the case of not explicitly defining the value comparison for attribute x.

- The relation r on the attribute set U satisfies the general positive boolean dependency f and is written $r(f)$ ($r(F)$) if $T_r \subseteq T_f$ ($T_r \subseteq T_F$).

- *Every positive Boolean formula f in P(U) with given value comparisons is called a generalized positive Boolean dependency, the resulting scheme in this case is called a scheme with generalized positive Boolean dependency.*

1.6. Classification of generalized positive Boolean dependency classes

Survey and specification of different variants of generalized positive Boolean dependence. Find a specification for general relaxed dependence and show that relaxed

dependence in general and relaxed functional dependence in particular are just special cases of general positive Boolean dependence.

***1.6.1. IE Class* (*Implication Fomula & Equal Comparison*)**

Class IE is a class of classic functional dependency diagrams, built on the basis of inference operations and equality comparisons..

Cho tập các thuộc tính $U = \{x_1, x_2, \ldots, x_n\}$, $n \geq 1$. Conjecture $X, Y \subseteq U$. A dependency of class IE is an expression of the form f: $X \rightarrow Y$.

Based on the logical expressions X and Y, we distinguish the following types of functional dependencies:

IE-1: The logical expression has the form $X \rightarrow Y$, we have the traditional functional dependency diagram.

IE-2: Logical expressions of the form $\vee X \rightarrow Y$ give us a strong functional dependence diagram

IE-3: Logical expressions of the form $X \rightarrow \vee Y$, We have a weak functional dependency diagram.

IE-4: Logical expressions of the form $\vee X \rightarrow \vee Y$, We have a dual functional dependency diagram.

The following table summarizes the specifications for subclasses IE1-4 of the IE class

Type of dependency	Name	Characteristic
$X \rightarrow Y$	functional dependency	Tightly constraint
$\vee X \rightarrow Y$	strong functional dependence	Relax the left side
$X \rightarrow \vee Y$	weak functional dependence	Relax the right side
$\vee X \rightarrow \vee Y$	dual function dependence	Relax
$X(\delta) \rightarrow Y$	Approximate dependence	Relax

Table 2. 2. Specification table of IE1-4 subclasses

Thus, the functional dependencies IE-1, IE-2, IE-3, IE-4 are relaxed functional dependencies and they are special cases of general positive Boolean dependence.

***1.6.2. LA Class* (*Logic Fomula & Alpha Comparison*)**

Based on the relaxation conditions, we divide the LA class into the following subclasses:

LA-1: Class LA-1 is a class of dependencies built on the basis of inference operations and alpha comparisons.

Given a set of attributes $U = \{x_1, x_2, \ldots, x_n\}$, $n \geq 1$. Conjecture $X, Y \subseteq U$. A dependency of class LA-1 is an expression of the form f: $\alpha_X \rightarrow \alpha_Y$ with comparisons α.

The r relationship is satisfactory LA-1: $\alpha_X \rightarrow \alpha_Y$ and written $r(X(\alpha) \rightarrow Y(\alpha))$, if for any two sets $u, v \in r$, satisfy the constraints α_X on set X, then u and v also satisfies the constraints specified by the difference function α_Y on set Y:

$$r(X(\alpha) \rightarrow Y(\alpha)) \overset{def}{\Longleftrightarrow} \forall u,v \in r: \alpha(u.X, v.X) \Rightarrow \alpha(u.Y, v.Y)$$

In class LA-1, we have the following representative dependencies:

- f: $\alpha_X \rightarrow \alpha_Y$, we have different dependencies, with α_X và α_Y are other wrong functions defined:

For relationship r on the set of attributes U, $a \in U$ and difference unit m_a. Function diffrent wrong α_a on attributes a Binding specifications of difference unit m_a: With two values ß$_x$, $y \in d_a$, We define $\alpha_a(x, y) = 1$ if only if $m_a(x, y)$ satisfy conditions for α_a In the form of comparative expressions with comparisons $=$, $\neq$, $<$, $\leq$, $>$, and $\geq$.

Cho tập thuộc tính $X \subseteq U$. Hàm sai khác ϕ_X trên tập thuộc tính X là hội logic của các hàm sai khác trên mọi thuộc tính $a \in X$: $\phi_X = \bigwedge_{a \in X} \phi_a$

Diffrent wrong Dependencies on the is the specific case of the loosening function dependence, in this section indicates that the other is different from the class LA-1.

- If the logic expression is formal f: $X(\delta) \rightarrow Y$, $0 \leq \delta \leq 1$ We depend on the loosening according to the force.

Lớp LA-2: LA-2 layer is the dependent layer built on the basis of logic boole operations and equal treatment. The representative of La-2 class is dependent on Boole Duong.

Lớp LA-3: The LA-3 layer is the dependent layer built on the basis of logic boole and alpha treatment. The representative of the LA-3 class is a general positive boole dependence, which is also a dependent class that depends on the logic dependent in the database that has been researched by domestic and foreign authors groups.

1.7. Conclusion Chapter 1

In chapter 1, the thesis presented the basis of data dependence in the database of relations and analysis to see that the comparison of sets according to the equation is not enough for the database to reflect properties. Diverse of the semantics of data in reality.

Chapter 1 also presents studies related to the research orientations of the thesis: research and development, expanding the dependent layer of boole is approximately. Can summarize the important highlights of chapter 1 as follows:

(1) With equality comparisons, we can develop all kinds of functional data and some variants of this form such as loose function dependence, approximate function dependence.

(2) The equivalent theorem for classic function allows solving member problems on the basis of logical failure $X \rightarrow Y$.

(3) The technique of building the value table of relationships according to equation comparisons. The technique of checking the relationship of the relationship with the function depends on equilateral comparison.

(4) Survey and specify different variants of general positive boole dependence. Find a specification for a general relaxation dependence and point out that the loosening dependence in general and the loosening function in particular are only specific cases of the general boole dependence..

CHAPTER 2. CLASSES APPROXIMATION DEPENDENCY IN THE DATABASE

2.1. Preliminary

In this chapter, the thesis proposes the new tyoe of dependency

Approximate Positive Boolean Dependencies, Approximate Generalized Positive Boolean Dependencies, Approximate Weak dependencies. The chapter content will include the following specific results:

(1) Proposing how to build a general positive booling scheme from a approximation schema is predesigned.

(2) Proposing the Lambda function on the domain of the attribute and proving the theorem of mathematical guarantee for the application of the Lambda function to develop the concept of Approximate Positive Boolean Dependencies, Approximate Generalized Positive Boolean Dependencies.

(3) Proposing the properties and theorem to build a new approach to the approximate boole dependence and dependence on the positive boole with approximation according to the value rather than the number of sets. Specifically, proving the theorem of the necessary and sufficient conditions for an agent relationship Approximate Positive Boolean Dependencies.

(4) Proposing general concept of dependence on approximation on sets of numerical and non -digital properties, not necessarily in the form of functions but according to an arbitrary boole expression with more general rule than equation than equal to equality..

The results in this chapter are published in CT1-CT6, in "*Published science works*" of PhD.

2.2. Building Lambda function and Measure

2.2.1. Lambda function

Let U be an attribute set. For each attribute A in U, we define a mapping λ_A from V_A to the real number space as follows:

$\lambda_A: V_A \to \mathbb{R}$

L1) $\forall a,b \in V_A: a = b \Rightarrow \lambda_A(a) = \lambda_A(b)$

L2) $\exists a, b \in V_A: \lambda_A(a) \neq \lambda_A(b)$

λ_A is called *quantitative function for the attribute A.*

Example 2.1

- Let attribute *A* be the type string, we define $\forall s \in V_A: \lambda_A(s) = \text{len}(s)$. We have,

λ_A("Internet") = 8; λ_A("e-mail") = 6; λ_A("") = 0 (the quantification of empty string is 0).

- Let attribute *B* be the salary coefficient, we define $\forall a \in V_B: \lambda_B(a) = a$. Then

$$\lambda_B(2.3) = 2.3; \lambda_B(8.0) = 8.0$$

- Let attribute *C* be the level of satisfaction rank (of customers for an airline company, for example) with the values V_C = {very satisfied, satisfied, normal, not satisfied}, we define

λ_C(very satisfied) = 4; λ_C(satisfied) = 3;

λ_C(normal) = 2; λ_C(not satisfied) = 1.

Theorem 2.1

Let λ_A be given. Then the function α_A is defined by the following formula (*)

$$\forall a, b \in V_A: \alpha_A(a,b) \overset{def}{\Longleftrightarrow} (\lambda_A(a) = \lambda_A(b)) \quad (*)$$

satisfies properties A1, A2 and A3 in the definition of α.

2.2.2. Measure

The measure d on a set V is a mapping $d: V \times V \to \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of non-negative real numbers, satisfying the following properties :

$\forall a, b, c \in V$

- *Non-negativity*: $d(a,b) \geq 0$, $d(a,a) = 0$

- *Symmetry*: $d(a,b) = d(b,a)$
- *Triangle*: $d(a,b) + d(b,c) \geq d(a,c)$.

Example 2.2

On line (one-dimensional space), the measure between two points x and y, representing the distance from x to y, is calculated by $d(x,y) = \|x-y\|$. In n-dimensional space, the measure between two points $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n})$ and $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_n})$ is defined by the *Euclidian distance* as follows:

$$e(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

Let $p = (U, F)$ be a schema with Lambda functions defined for each attribute. For each tuple $\boldsymbol{u} = (\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{u_n}) \in r$, we define $\lambda(u)$ as a vector

$$\lambda(\boldsymbol{u}) = (\lambda_1(\boldsymbol{u_1}), \lambda_2(\boldsymbol{u_2}), \dots, \lambda_n(\boldsymbol{u_n}))$$

Theorem 2.2

Given a relation r on the attribute set U and Lambda functions on each attribute A in U. Let d be an arbitrary measure in an n-dimensional vector space. Then the following d function is a measure*:

$$\forall u, v \in r: d^*(u, v) = d(\lambda(u), \lambda(v))$$

We call d^* the *inductive measure* from the d measure.

2.3. Approximate Generalized functional dependencies

In this section, the concept will be expand Approximate functional dependencies on numerical properties, then the concept will then be expanded approximate functional Dependencies on arbitrary or non -numerical properties.

the concept Approximate Generalized functional dependencies is defined as follows:

Give two sub -subtenses of attributes X and Y in the volume of U and two thresholds ε_1, ε_2 are two non -negative real numbers. Depends on the approximation function of the volume of Y attribute according to the X attribute X is the expression:

$$X \varepsilon_1 \rightarrow \varepsilon_2 Y$$

We say that the intake of the Y attribute depends on the approximation of the X properties and the relationship r is the approximate function dependence $X \varepsilon_1 \rightarrow \varepsilon_2 Y$ if with every set of u and v in relations r, two sets u and v There is not too much deviation ε_1 on the episode of properties X, these two sets are not exaggerated ε_2 on the attribute Y.

where d is the measurement function on the sets of the relationship.

$$\forall u, v \in r(U): d(u.X, v.X) \leq \varepsilon_1 \Rightarrow d(u.Y - v.Y) \leq \varepsilon_2$$

2.4. Building approximate relationship schema through measurement

Relationship schema is approximately a pair p = (U, F, d), where U is the attribute, F is the set of approximate functional dependencies according to the measurement d.

For each measurement d on D we link a value $\varepsilon \geq 0$ And called the threshold of d on D. If D exists two values a and b to satisfy propercities $d(a,b) > \varepsilon$ then ε called *threshold is not trivial*; gainsay, if with every pair of values $a, b \in D$, by assumption $d(a,b) \leq \varepsilon$ then ε *called trivial threshold*

2.5. Weak Dependency

A formula of the form $f: \wedge X \rightarrow \vee Y$ is called *weakly derived formula*. A *weak dependency* (WD) on U is a weakly derived formula $f: \wedge X \rightarrow \vee Y$.

We say the relation r satisfies WD f and write $r(\wedge X \rightarrow \vee Y)$ if for every pair of tuples u and v in r that are equal on X then they must be equal on some attribute B of Y.

$$r(\wedge X \rightarrow \vee Y) \Leftrightarrow (\forall u,v \in r): (u.X = v.X) \Rightarrow \exists B \in Y: (u.B = v.B)$$

Thus, an WD is a special case of the positive Boolean dependency. We also have:

The relation *r satisfies* the weak dependency *f* (the set of WDs *F*) if and only if $T_r \subseteq T_f$ ($T_r \subseteq T_F$).

Theorem 2.3

Each positive Boolean formula is equivalent to a set of weakly derived formulas (in the sense that they all have the same truth-table).

2.6. Approximate Weak Dependency

The concept of approximate functional dependencies was proposed by Jyrky Kivinen et al in 1995, where the term *approximation* is understood as if at least ε tuples are removed from the relation *r* then the rest will satisfy the given functional dependency. In this paper, a general concept of approximate dependence is proposed on a set of numerical and non-numerical attributes, according to an arbitrary positive Boolean formulas with more general comparisons.

Given an attribute set *U* with a measure *d* and a weak derivation formula on *U*, *f*: $\wedge X \rightarrow \vee Y$. Let ε_1, ε_2 be two non-negative real numbers. An *approximate weak dependency* (AWD) of attribute set *Y* on attribute set *X* is an expression of the form:

$$X \; \varepsilon_1 \rightarrow \varepsilon_2 \; Y$$

We say that the relation *r*(*U*) satisfies the *approximate weak dependency* $X \; \varepsilon_1 \rightarrow \varepsilon_2 \; Y$ if

$\forall u,v \in r$: $d(u.X,v.X) \leq \varepsilon_1 \Rightarrow \exists B \in Y$: $d(u.B, v.B) \leq \varepsilon_2$

Example 4.3

On the promotional day, a supermarket selling item *H* has a policy to give each customer a discount of 10 units of the assumed currency *T* from 1 PM onwards. Information at the cashier is given in the form of the relation *r* as follows:

ID	Time	Quantity	Price	Total
1	10	5	10	50
2	11	8	10	80
3	12	4	10	40
4	13	5	10	40
5	14	8	10	70
6	15	4	10	30

Table 4.1. Relation r

Consider the following dependencies:

- Functional Dependency:
  h: Quantity, Price $\rightarrow$ *Total*
- Weak Dependency :
  w: Quantity, Price $\rightarrow$ *Total*
- Approximate Weak Dependency :
  f: Quantity, Price (0) $\rightarrow$ (10) *Total*

We can see that two transactions 1 and 4 buy the same quantity but at two different times, then the Total has two different values, so the relation *r* is not satisfied functional dependency *h*.

Similarly, relation *r* is not satisfied approximate weak dependency *w*.

However, relation *r* satisfies the approximate weak dependency *f*: two customers buying the same quantity at different times will pay different sums: Total and Total – 10T.

We call p = (U, $\varepsilon_1, \varepsilon_2$, λ, d, F) a relation schema with approximate weak dependencies (RSAWD), where

• U is a set of n attributes, each of which is equipped with a quantitative Lambda function λ.

• d is an arbitrary measure.

• F is a set of weak consequence formulas $\wedge X \rightarrow \vee Y$, $X, Y \subseteq U$.

• $\varepsilon_1 \geq 0, \varepsilon_2 \geq 0$ are approximate thresholds.

Given a relation r on a RSAWD $p = (U, \varepsilon_1, \varepsilon_2, \lambda, d, F)$. For each pair of tuples $u = (u_1, u_2, \dots, u_n)$, $v = (v_1, v_2, \dots, v_n)$ in r, we define:

$$t(u, v) = (t_1, t_2, \dots, t_n)$$

$$t_i = \left(d\left(\lambda(u_i), \lambda(v_i)\right) \leq \varepsilon\right)? 1 : 0 \; ; \; 1 \leq i \leq n$$

$$\varepsilon = min\,(\varepsilon_1, \varepsilon_2)$$

$$T_r = \{t(u,v) \mid u, v \in r\}$$

and call T_r the value table of the relation r according to the schema p.

Theorem 2.4

Let $p = (U, \varepsilon_1, \varepsilon_2, \lambda, d, F)$ be a RSAWD and f be an AWD. Let r be a relation on U. Then,

(*i*) Relation r satisfies f if and only if $T_r \subseteq T_f$.

(*ii*) Relation r satisfies the set of AWDs F if and only if $T_r \subseteq T_F$

2.7. Proposal Approximate Generalized dependencies

- Continuing to expand the concept of Boole Duong Sang dependency dependence on the approximate boole that can help us manage more complex databases. Especially allows expanding data search capabilities.

Database and rough database research directions also allow us to search for the queries of the above type. However, the dependence between attributes in the form of general logic accompanied by measurement reflects the approximation is also an open issue, worthy of research.

The concept of an approximate dependency dependence is as follows:

Give U set includes n attributes, một công thức Boole f dương trên U, và các hàm λ_i trên mỗi thuộc tính i trong U. For the threshold ε_i are non -negative real numbers per attribute i in U. We call f is a Approximate Generalized dependencies to the threshold $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$.

We say that Approximate Generalized dependencies f satisfies r to the threshold ε if with all sets $u, v \in r$: $f(t(u,v)) = 1$, in that $t(u,v)$ is determined as follows:

$$t(u, v) = (t_1, t_2, \dots, t_n)$$

$$t_i = (|\lambda_i(u_i) - \lambda_i(v_i)| \leq \varepsilon_i)? \; 1 : 0$$

The above conditions have the following meaning:

Relationship r satisfies Approximate Generalized dependencies if each pair of sets has the measurement of each attribute is below the stipulating the formula f.

Definition 2.6

We call the four $p = (U, \lambda, \varepsilon, F)$ is a relationship scheme with an Approximate Generalized dependencies, in which

- U is the set n atributes,
- F is set Approximate Generalized dependencies on U,
- $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ are quantitative functions for the inner properties U,
- $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ are the approximation threshold for internal properties U,
- $1 \leq i \leq n$

Give relations r on the relationship with an Approximate Generalized dependencies, $p = (U, \lambda, \varepsilon, F)$. We denote, $T_r = \{t(u,v): u, v \in r\}$

and call T_r is the truth of relations r according to the schema p.

Theorem 2.5

Cho lược đồ quan hệ với phụ thuộc Boole dương xấp xỉ $p = (U, \lambda, \varepsilon, F)$ và một phụ thuộc

Boole dương xấp xỉ $f \in F$. Cho quan hệ *r* trên *U*. Khi đó,

(i) Relationship *r* satisfies Approximate Generalized dependencies i *f* if only if $T_r \subseteq T_f$:

$$r(f) \Leftrightarrow T_r \subseteq T_f$$

(ii) Relationship *r* satisfies Approximate Generalized dependencies if *F* if only if $T_r \subseteq T_F$:

$$r(F) \Leftrightarrow T_r \subseteq T_F$$

2.8. Approximate generalized positive Boolean dependencies

Definition 2.7

Let *U* be an attribute set and *f* be a positive Boolean formula *f* over *U*. Let ε_1, ε_2 be two non-negative thresholds. Approximate generalized positive Boolean dependency (AGPBD) is a set of approximate weak dependencies *S* equivalent to *f* with the threshold ε_1, ε_2 .

We say that the relation *r*(*U*) satisfies an AGPBD *f* if *r* satisfies every AWD in *S*.

Since *f* equivalent to *S*, then $T_f = T_S = \cap\{T_g \mid g \in S\}$. We have the following result as a corollary of theorem 4.1.

Corollary 2.1

Let p =($U, \varepsilon_1, \varepsilon_2, \lambda, d, F$) *be a RSAWD and f be an AWD. Let r be a relation on U. Then,*

(*i*) *Relation r satisfies AGPBD f if and only if* $T_r \subseteq T_f$.

(*ii*) *Relation r satisfies the set of AGPBDs F if and only if* $T_r \subseteq T_F$)

2.9. Conclusion Chapter 2

In chapter 2, the thesis presented the results of building the new Lambda transformation function, based on the Lambda function was built and a combination of Approximate functional Dependencies and Approximate Generalized dependencies Newly included: The approximate boolean dependencies, Approximate Generalized Positive Boolean Dependencies, Approximate Weak dependencies and prove the equivalent between the Positive Boolean Dependencies and pointed out and prove the relationship between the Positive Boolean Dependencies. Approximate Positive Boolean Dependencies and Approximate Generalized Positive Boolean Dependencies.

The classification and proposal of a common model for data dependencies is one of the issues that are being concerned by large data researchers. With an arbitrary measurement, through the Lambda functions on the domain of the attributes, we can assess the approximation of the ministries in the relationship including digital and non -digital properties such as costumes, skills acting, ... to apply in evaluation and classification of objects as well as searching approximately the objects in the database

CHAPTER 3. RELATIONSHIP SCHEME PROCESSING ALGORITHMS

3.1. Preliminary

In chapter 3, focus on presenting the following contents:

(1) Proposing the process of solving the problem with three approaches is to direct the Vuong Hao algorithm, direct proof of the assembly and proof of the anti -award and new results on application. Methods to prove tautology to solve the problem of the membership of the class depends on the logic dependent and compact the guiding laws in the knowledge base

(2) Building the algorithm to find the closed bag of a set of attributes for the class depends on the logic.

(3) Buildingthe key algorithm for class depends on logic. The key in a relationship is understood as a small set of properties that can be determined no more than one set in the database. The key is mainly applied in database query algorithms.

These results are one of the basic contributions of PhD, Presented in CT2, CT4 - *Published science works*

3.2. Buiding method to transfer a logical formula to the conjunctive normal form

To be able to expand the concept of a Approximate Positive Boolean Dependencies, the following arguments are related to the transfer of any logic formula. Conjunctive normal form (CNF).

Definition 3.1

Give the Boole variable $U = \{a_1, a_2, ..., a_n\}$. Logic formula Ψ on U There is a standard form if Ψ is represented in the form conjunctive of which each: $\Psi = g_1 g_2 \ldots gm$

In which each g_i , $1 \le i \le m$ is a disjunctive of the ingredients in U.

For example, for sets of the variable $U = \{a, b, c, d\}$. Fomula Ψ following belongs conjunctive normal form:

$$\Psi = (a+b+c')(b+d)a'(b+c).$$

Theorem 3.3

Every logic formulas are equivalent to a formula CNF.

There are two methods to transfer a logic formula to the form of CNF as follows:

3.2.1. Logic method

Transfer logic formula Ψ to CNF format by applying the following laws in the proposition logic:

Logic rule

With all formulas X, Y, Z on the Boole U. we have:

(R1) Commutative rule: $X + Y \equiv Y + X$; $XY \equiv YX$.

(R2) Associative rule: $X+(Y+Z) \equiv (X+Y)+Z$; $X(YZ) \equiv (XY)Z$.

(R3) Idempotent rule: $X+X \equiv X$; $XX \equiv X$.

(R4) Neutral rule: $X+0 \equiv X$; $X+1 \equiv 1$; $X0 \equiv 0$; $X1 \equiv$ X.

(R5) $XX' \equiv 0$; $X+X' \equiv 1$.

(R6) Negative of negation: $X'' \equiv X$.

(R7) $X \to Y \equiv X'+Y$.

(R8) $(X \to Y)' \equiv XY'$.

(R9) de Morgan rule: $(XY)' \equiv X' + Y'$; $(X+Y)' \equiv X'Y'$.

(R10) Distributive rule: $X + YZ \equiv (X+Y)(X+Z)$; $(X+Y)Z \equiv XZ + YZ$;

(R11) Swallow rule: $X + XY \equiv X$; $X(X+Y) \equiv X$.

Repeat the application of the rules of reduction, the law of de Morgan, the law of distribution in the logic, until obtained the logic formula has the form of CNF.

3.2.2. Table method

Transfer logic formula Ψ on the utility set of u -type CNF according to the following table method.

1. Create the value table T_Ψ According to variables in U and logic formula Ψ

2. From each row $t = (v_1,\ldots,v_n)$ satisfy $\Psi(t) = 0$ *in* value table T_Ψ create a base disjunctive $g_t = (z_1 + \ldots + z_n)$, with

$z_i = \begin{cases} x_i \text{ nếu } v_i = 0 \\ x_i' \text{ nếu } v_i = 1 \end{cases} \quad 1 \le i \le n$

3. Return results $M = g_1 \ldots g_k$ is conjunctive of the base disjunctive.

Kết quả thu được sau bước 3 chính là dạng chuẩn hội của công thức Ψ.

The result obtained after step 3 is conjunctive normal form

Two methods can produce two different types of CNF, although the results are equivalent. Both methods are NPC classes and have a complexity of calculation O(2^n), where n is the number of logic variables.

Proposition 3.1

Each boole formula is equivalent to a set of weak formulas.

Consequences 3.1

Each boolean formula equivalent with weak formulas.

Each set boolean formula equivalent with weak formulas.

Consequences 3.1

Each set positive boolean formula equivalent with a set weak formulas.

3.3. Building method to prove the tautology

Definition 3.2

For the set sign the Boolean *U*. A Boolean formula *f* on U is called tautology if $f(x) = 1$ with every value assignment *x*.

3.3.1. Direct proof method with CNF

According to this method, to prove the formula *f* is the correct constant we follow the following two phases:

Phase 1. Transfer *f to* CNF. $f \equiv u_1 u_2 \ldots u_k$

Phase 2. KConclude: *f is* tautology if only if $u_i = 1$, $1 \le i \le k$.

3.3.2. Wang Hao method

Wang Hao method use to prove tautology $T \to P$ follow the following steps.

Step 1 (CNF → DNF).

Bring the left side *T* to CNF.

Bring the right side *P* to DNF is logic formula have CNF.

Bring the formula $T \to P$ to CNF → DNF we can applying rule R1 – R11 To change the two sides *T* and *P* The following standard form: $t_1\ t_2 \ldots t_u \to p_1 + p_2 + \ldots + p_v$

The left side is a CNF, the right is a DNF.

Step 2 (Eliminate negation). Transfer of negative signs. If *x*' appears on the left side *x*' to the right side *x* and vice versa.

Step 3 (detached). If the current line has one of the following two forms:

- Form 1: $t_1 \ldots (a + b) \ldots t_u \to p_1 + \cdots + p_v$

Then replace with 2 lines:

$$\begin{cases} t_1 \ldots a \ldots t_u \to p_1 + \cdots + p_v \\ t_1 \ldots b \ldots t_u \to p_1 + \cdots + p_v \end{cases}$$

- Form 2: $t_1 \ldots t_u \to p_1 + \ldots + a \cdot b + \ldots + p_v$

Then replace with 2 lines:

$$\begin{cases} t_1 \ldots t_u \to p_1 + \ldots + a + \ldots + p_v \\ t_1 \ldots t_u \to p_1 + \ldots + b + \ldots + p_v \end{cases}$$

Step 4 (Conclude). One row is proved when and only when one variable appears in both sides.

3.3.3. Robinson method

In logic, the solution is used to solve the problem $f \to g$. The solving is also known as the method of proof by reduction and absurdum.

Let *Q* form CNF. Robinson method Perform the process of strategy *Q* as follows: Replace each pair of element $(x + B)$, $(x' + C)$ in *Q* by element $B + C$, *B* và *C* are logical formulas. The process is repeated until there is no existence in *Q* element have form belove or not No more strategy. If $Q = \varnothing$ Then we say that it is successful, that mean $Q \to false$, On the contrary, we say that the solving is unsuccessful. When successfully solved, because $Q \to false$ So we

conclude that *Q* is *false*.

Algorithm 3.1

```
1    Algorithm  Unif(H)
2    Input      CNF(H)
3    Output     unif(W)
4    Begin
5      W := H;
6      while there are terms (x + B) and
7          (x' + C)  in W do
8          delete (x + B) in W;
9          delete (x' + C) in W;
10         (B + C) to W
11     endwhile
12     return W;
13   End Unif
```

Theorem 3.1

The following proof of tautology belongs to the NPC class:

(i) Prove the Wang Hao method.

(ii) Proof of CNF method.

(iii) Proof of CNF method.

The assignment of the equation and truth of the relationship

Definition 3.3

Give relationships *r* with *n* attribute. Convention that each domain d_i of attributes *i*, $1 \le i \le n$, contains at least two elements. With each pair of sets

$$u = (u_1, u_2, \dots, u_n) \text{ and}$$
$$v = (v_1, v_2, \dots, v_n)$$

in *r* we build one vector Boolean *t*(*u*, *v*) as follows:

$$t(u, v) = (t_1, t_2, \dots, t_n)$$
$$t_1 = (u_i = v_i)\ ?\ 1 : 0$$

With each relationship r on episode n, the attribute of u, we determine the sets vector Boole

$$T_r = \{t(u, v): u, v\ r\} \quad and\ call\ T_r\ is\ the\ truth\ table\ of\ the$$

relationship r.

According to the definition we see:

- *If r is an empty relationship* $T_r = \varnothing$.
- *If the relationship r contains at least one set of u, it is because* $\alpha(u,u) = e$ *the* $e \in T_r$.

Definition 3.4

For U is an empty attribute. Each Boolean formula on *U is a positive Boolean Dependencies*

For relationship r on U. We say that the relationship r satisfied positive Boolean Dependencies f and denote *r*(*f*) if with all set *u*, *v* in *r*:

$$r(f) \Leftrightarrow \forall u, v \in r: f(t(u,v)) = 1.$$

For relationship r on U. We say that the relationship r satisfied set positive Boolean Dependencies F and denote *r*(*F*) if *r* satisfied all dependency in*f*: $r(F) \Leftrightarrow \forall f \in F: r(f)$

For *U* and sets Boolean formula on *U*. We call *p* = (*U*, *F*) is relationship schema with set *positive Boolean Dependencies F.*

Theorem 3.2

For relationship schema p = (U, F) and a positive Boolean Dependencies $f \in F$. *For relationship r on*. Then,

(i) Relationship r satisfied positive Boolean Dependencies f if only if $T_r \subseteq T_f$: $r(f) \Leftrightarrow T_r \subseteq T_f$

(ii) Relationship r satisfied set positive Boolean Dependencies s F if only if $T_r \subseteq T_F$: $r(F) \Leftrightarrow T_r \subseteq T_F$

Following positive Boolean Dependencies with Boolean formula respectively as follows:

Dependencies	Denote	Positive Boolean formula
Function Dependencies	$X \rightarrow Y$	$\wedge X \rightarrow \wedge Y$
Strong Dependencies	$X\,(s) \rightarrow Y$	$\vee X \rightarrow \wedge Y$
Weak Dependencies	$X\,(w) \rightarrow Y$	$\wedge X \rightarrow \vee Y$
Duality Dependencies	$X\,(d) \rightarrow Y$	$\vee X \rightarrow \vee Y$
Multivalent Dependencies	$X \twoheadrightarrow Y$	$\wedge X \rightarrow (\wedge Y) \vee (\wedge(U \backslash X \backslash Y))$

Table 2. 3. Positive Boolean Dependencies

3.4. Build a derivation algorithm in the relational schema

- In this section and the following sections, we will state and prove necessary and sufficient conditions for a logical formula to be expressed as an association of derived formulas..

- The meaning of this result is: logical dependencies can be used to describe diverse types of data constraints in practice. Functional dependencies and their variants, for example, relaxed functional dependencies and strong, weak, and extreme dependencies are expressed through derivation formulas. $X \rightarrow Y$.

3.4.1. Consequence in relation diagrams with functional dependencies

Each dependency between sets of attributes in a relational database is described through an expression f. The expressions describing the dependence can be analytic For example: Two withdrawals from the same ATM card at two locations over 100 km apart cannot be less than 30 minutes apart..

Given attribute set U, dependencies f and relation r on U. We say that the relation r satisfies the dependency f, and written $r(f)$ if: $\forall u, v \in r$: $\gamma(f,u,v) \rightarrow \lambda(f,u,v)$, in that γ and λ are the predicates defined above f and set attribute u, v.

Relationship schema is a set $p = (U, \Sigma)$, in there U is an attribute set, Σ is the set of above dependencies U.

All internal relationships and dependencies are understood as dependencies built on a set of attributes U given.

Each relation r above relationship schema p called one expression of p. All expressions r above relationship schema p have to satisfies dependencies Σ. We say, relationships *r satisfy the set of dependencies* Σ, and write $r(\Sigma)$, if r satisfies all internal dependencies Σ,

$$r(\Sigma) \overset{def}{\Leftrightarrow} \forall f \in \Sigma: r(f)$$

Traditional Function Dependencies and variations of Traditional FD is described as follows

Function Dependencies	$\forall$u, v$\in$R: Eq(u.X, v.X) $\Rightarrow$ Eq(u.Y, v.Y) Eq(u.X, v.X) $\overset{def}{\Leftrightarrow} \forall A \in X$: u.A = v.A
Strong Dependencies	$\forall$u, v$\in$R: Eq1(u.X, v.X) $\Rightarrow$ Eq(u.Y, v.Y) Eq1(u.X, v.X) $\overset{def}{\Leftrightarrow} \exists A \in X$: u.A = v.A
Weak Dependencies	$\forall$u, v$\in$R: Eq(u.X, v.X) $\Rightarrow$ Eq1(u.Y, v.Y)
Multivalent Dependencies	$\forall$u, v$\in$R: Eq1(u.X, v.X) $\Rightarrow$ Eq1(u.Y, v.Y)

Bảng 3. 1. Specific description of types FD, SD, WD, MD

3.4.2. Problems related to data dependence

The concept of data dependence is studied to best solve the following problems.

Updated problem: Given relation r on the diagram $p = (U, \Sigma)$ and two sets t and t' on U. The update problem involves three operations: add, delete, and edit as follows:

Assumption: $r(\Sigma)$: relation r satisfies the dependency set Σ.

Request: $r'(\Sigma)$: After the update (add, delete or update) result relationship r' still satisfy the dependency set Σ.

Specifically, if r previously satisfied the dependency set Σ then after performing the operations of adding, deleting, and editing to obtain the relationship r' then r' still have to satisfy Σ. Ta nói Σ is invariant to update operations.

We see that the edit operation is equivalent to a sequential sequence of two delete and add operations, so we only need to require Σ to be invariant for the two delete and add operations.. On the other hand, according to the definition of satisfaction, if $r(\Sigma)$ then with all relationships $r' \subseteq r$, we have $r'(\Sigma)$. From here it can be deduced Σ is invariant to addition operations.

To add tuple t to relation r we must check the following condition.

$$\forall u \in r, \forall f \in \Sigma: \gamma(f,u,t) \rightarrow \lambda(f,u,t)$$

This procedure requires great computational complexity because it depends on the sizes of the sets Σ and r.

This difficulty is initially resolved by the concept of standardization.

Standardization problem: Replace the given relation with a smaller set of relations that is most convenient for updating. With traditional functional dependencies, the database designer separates each relation into component relations of standard form "*the best*" allows testing on only one key K of r.

If $\exists u \in r$: $u.K = t.K$ then do not add t; opposite: add t in r.

Search problem: Given relation r on the diagram $p = (U,\Sigma)$ and a set of t on U. Search requests can be very diverse.

To speed up the search, we must apply the concept of a key as a small enough subset of attributes that allows a set to be uniquely identified in a relation. The locking concept is built on the basis of the membership problem below.

Membership problem: Let *relationship schema* $p = (U,\Sigma)$ and the relation r on p, we have $r(\Sigma)$, That mean, r satisfies all internal dependencies Σ. In addition to Σ, r can satisfy other dependencies as well. For example, if $U = ABC$, $\Sigma = \{A \rightarrow B, B \rightarrow C\}$, then every relationship r satisfies Σ, and r also satisfies the dependency $A \rightarrow C$.

The membership problem is stated as follows:

Hypothesis: Given a relational database schema $p = (U, \Sigma)$ and a dependency g on U. Conclusion: Is $g \in \Sigma+$?

Solving the membership problem allows us to solve the key problem as well. Under what conditions can we solve the membership problem using logic tools, specifically, if we consider the expressions that describe the dependencies as logical expressions built on variables representing attributes in U, then in what circumstances is $\Sigma \vdash$ g equivalent to $\Sigma \models g$?

This problem is entirely resolved by Armstrong with the traditional PTC as follows:

Given a Relational Database Schema $p = (U, \Sigma)$ and a dependency g on U. We say that the dependency g can be derived from relations with at most two tuples from the set of dependencies Σ, and we write $\Sigma \vdash_2 g$ if $\forall r \in REL_2(U)$: $r(\Sigma) \Rightarrow r(g)$.

Equivalent Theorem: For a Relational Database Schema $p = (U, \Sigma)$ and a functional dependency g on U, the following three propositions are equivalent:

- $\Sigma \models$ g (logical consequences)
- $\Sigma \vdash$ g (consequences through relations)
- $\Sigma \vdash_2$ g (consequences through relations with at most two tuples).

Armstrong also proposed three axioms as the basis for logical inference in PTH traditional $\forall X, Y, Z \subseteq U$:

- Reflexivity Axiom: If Y ⊆ X, then X→Y.
- Augmentation Axiom: If X→Y, then XZ→YZ.
- Transitivity Axiom: If X→Y and Y→Z, then X→Z.

According to the equivalence theorem, to prove Σ ├g, we can apply Armstrong's axioms, and to disprove Σ ├g, we only need to construct a relation r with 2 sets such that r(Σ) holds, but r does not satisfy g.

Related works to functional dependencies and positive Boolean dependencies before 1992 only concerned the equivalence relation *Eq* defined as follows.

$$\forall X \subseteq U, \forall u, v \in r(U):$$
$$Eq(u.X, v.X) \overset{def}{\Longleftrightarrow} \forall A \in X: u.A = v.A$$

In 1992, the research team led by Nguyễn Xuân Huy expanded the concept of equivalence and introduced a general framework for positive Boolean dependencies, preserving the equivalence theorem as a foundation for inference mechanisms and related problems such as search, membership, key determination, which are essential issues in database management..

3.4.3. Consequences algorithm

General Concept

Among the class of positive Boolean dependencies, the main form of inference formulas is the class of functional dependencies first formalized by Codd in 1970 .

We define $\forall v \in \mathcal{B}^n$, $Set(v) = \{A \in U \mid v.A = 1\}$, then there exists a bijective $\mathcal{B}^n$ and 2^U.

If we consider X as a set of logical variables, for each assignment v, we have $X(v)=1$ if and only if $Set(v) \supseteq X$.

Let f: $X \to Y$ be a Boolean function on U, for each assignment v, we have $f(v) = 1$ if and only if $Set(v) \supseteq X \Rightarrow Set(v) \supseteq Y$.

The notation I(U) represents the set of Boolean functions on the set of variables U.

We call a set of implications a closure system, denoted as F, where $F = \{f_1,\ldots,f_m\}$. For each Boolean function f: $X \to Y$, we know that $f(e) = X(e) \to Y(e) = 1 \to 1 = 1$, so the Boolean functions are positive Boolean formulas, which means $I(U) \subseteq P(U)$.

Intersection Consequences

We also know that every logic formula can be represented in the form of standard conjunction (intersection). In other words, every truth table T ⊆ Ɓn corresponds to a logic formula in standard conjunction form. The problem of representing a logic formula using a set of operations and given constants does not have a general solution. The following sections are related to this problem.

Problem

Determine the necessary and sufficient conditions for representing a logic formula in the form of Intersection Consequences

Lemma 3.1

(Lemma on the Closure Property of the & Operator in T_f). *For every CTSD f on U, T_f contains single-unit e assignment, z zero-assignment, and is closed under the & operator.*

For every HSD F on U, because the truth table T_F of F is the intersection of the truth tables of its member formulas, we have the following consequences.

Consequent 3.1

For every CTSD F on U, T_F contains single-unit e assignment, z zero-assignment, and is closed under the & operator

The following consequence establishes the necessary and sufficient conditions for a truth table $T \subseteq \mathcal{B}^n$ to be a truth table of a *Intersection Consequences*.

Consequent 3.2

Table $T \subseteq \mathcal{B}^n$ is truth table of *HSD if and only if T contains single-unit e assignment, z zero-assignment, and is closed under the & operator.*

3.5. Building an algorithm to Find Closure with General Positive Boolean Dependencies

Let U be the set of attributes and Ψ be the set of logical dependencies on U. Let $X \subseteq U$. We define the closure X^+ of X as the set of attributes as follows:

$$X^+ = \{a \in U \mid X \models a \in \Psi^+\}$$

The closure of the attribute set X is the complete set of attributes that depend on X.

Algorithm for Finding the Closure of an Attribute Set

Algorithm Idea: To find the closure X^+ of X, we initialize $X^+ = X$, then apply the membership algorithm to check for each attribute $a \in U - X$. If $X \models a \in \Psi^+$, we add a to X^+. With Ψ already converted to standard form, the condition $X \models a \in \Psi^+$ is equivalent to CNF(Ψ $(X \models a)') = \Psi Xa'$, which is equivalent to the condition Unif(ΨXa') = ∅."

Algorithm 3.2 (Algorithm for Finding the Closure in class of logical dependencies)

```
1    Algorithm  LClosure(X, Ψ)
2    Input      LNorm(Ψ); X ⊆ U
3    Output   X+ = {a ∈ U | X⊨ a ∈  Ψ+}
4    Begin
5       Y :=  X; V := U - X;
6           for each attribute a in V do
7                if  Unif(ΨXa') = ∅ then
8                    add a to Y;
9                endif;
10          endfor;
11        return Y;
12   End  LClosure
```

Proposition 3.1

The problem of finding closure in the class of logical dependencies belongs to the NPC class

3.6. Building an algorithm to find keys with a general positive Boolean dependency.

A key is a minimal subset of attributes that uniquely determines a tuple in the relation.

For *F* being a set of logical dependencies on U, the set $K \subseteq U$ is called a key if it satisfies:

- $K^+ = U$,
- $\forall a \in K: (K - a)^+ \neq U$.

If K satisfies only the first condition, then K is called a *superkey*.

The following algorithm performs the steps to find a key on the attribute set *U* and the given set of logical dependencies F.

Algorithm for Finding Keys

Starting from an arbitrary superkey *K*, iterate through each attribute a in K. If $(K - a)^+ = U$ then remove *a* from *K*.

Algorithm 3.3 (Finding Key *K* in class of logical dependencies)

```
1    Algorithm  LKey(U,  Lnorm(Ψ))
2    Input     U,  Lnorm(Ψ)
3    Output  K ⊆ U:
4    K+ = U
5    ∀a ∈ K: (K − a)+ ≠ U
6    Begin
7                K ≔ U;
8                for each attribute a in U
9    do
10                 if Unif(Ψ(K − a)a') = ∅
11   then
12                       delete a from K;
13                endif
14               endfor
15               return K;
16   End LKey
```

Proposition 3.2

The problem of finding keys in the class of logical dependencies belongs to the NPC class.

3.7. Conclusion chapter 3

In Chapter 3, the thesis presents the proposed results:

Approaches to consequences problems in the class of logical dependencies using three methods: Vuong Hao's method, indirectly solving method by union, direct solving method.

Algorithms for finding the closure of attribute sets in relational schemas with functional dependencies, finding the closure of attribute sets in relational schemas with general positive Boolean dependencies, finding keys with functional dependencies, and finding keys with general positive Boolean dependencies have been developed."

CONCLUSION AND FUTURE RESEARCH

I. The achieved results of the thesis

Some of the new contributions of the thesis focus on three groups of results:

(1) Building lambda transformation functions, based on the constructed lambda function, combining approximate functional dependencies and general positive Boolean dependencies. The thesis proposes new data dependencies, including *generalized approximate functional dependencies, approximate positive Boolean dependencies, generalized approximate positive Boolean dependencies, and weak approximate dependencies*.

(2) Establishing relationships between different types of logical dependencies: Demonstrating the equivalence between general positive Boolean dependencies and relaxed functional dependencies, the equivalence between general positive Boolean dependencies and approximate positive Boolean dependencies, and the equivalence between general positive Boolean dependencies and generalized approximate positive Boolean dependencies.

(3) Constructing several algorithms to solve the following characteristic problems:

- Using the Vuong Hao method, CNF, and union for solving consequences problems.
- Developing and evaluating computational complexity for algorithms to find the closure of an attribute set for functional dependencies and general positive Boolean dependencies.
- Constructing and evaluating computational complexity for key-finding algorithms for functional dependencies and general positive Boolean dependencies.

The results of this thesis can provide database designers and administrators with an overall understanding of the relationships between various logical dependencies, thus enabling them to choose the appropriate type of logical dependencies based on practical needs

II. The main contributions of the thesis

(1) Proposing new forms of dependencies, including generalized approximate functional dependencies, approximate positive Boolean dependencies, generalized approximate positive Boolean dependencies, and weak approximate dependencies.

(2) Establishing relationships between different types of logical dependencies, such as the relationship between general positive Boolean dependencies and relaxed functional dependencies, general positive Boolean dependencies and approximate positive Boolean dependencies, and general positive Boolean dependencies and generalized approximate positive Boolean dependencies.

(3) Proposing a process for solving consequences problems using three formal approaches. Developing and evaluating algorithms for finding closure and finding keys of an attribute set for the class of logical dependencies."

III. Future Developments of the Thesis:

Currently, there are various types of logical dependencies proposed to serve database design. To create a comprehensive model for these types of logical dependencies, we must determine the relationships between them. To achieve this goal, the thesis has demonstrated the equivalence between some types of logical dependencies, such as general positive Boolean dependencies, general conditional dependencies, general weak dependencies, and approximate dependencies, and has built the class of logical dependencies. If we continue to expand this class by incorporating other types of logical dependencies, the generality of the class of logical dependencies will become more reliable. This is also the next development direction for NCS

when continuing to pursue this topic:

Further investigate new types of dependencies, such as soft functional dependencies, matching dependencies, conditional dependencies, pattern-based dependencies, sequential dependencies, and any other dependencies proposed in the future. Establish relationships between these new types of dependencies and the dependencies mentioned in the thesis.

Incorporate new dependencies discovered by research groups into the class of logical dependencies built in the thesis.

Explore additional classic problems and establish standards for the class of logical dependencies.

Develop application software to solve practical problems for the class of logical dependencies and solve problems within this class using the union approach.

PUBLISHED SCIENCE WORKS

[CT1] Nguyen Xuan Huy, Truong Thi Thu Ha, Nguyen Thi Van (2016), "*The relationship between functional dependency and positive generalization of Boolean dependency in relational databases"*, *Proceedings of the 19th National Conference: Selected Issues in Information Technology and Communication*., Science and Technology Publishing House, ISBN: 978-604-67-0781-3, Ha Noi, page.361-365.

[CT2] Truong Thi Thu Ha, Nguyen Thi Van, Nguyen Xuan Huy (2016), "*Algorithm for determining closure and key using the normalization approach in the class of logic dependencies."*, *Journal on Information Technologies & Communications, ICT Research is a scientific publication of the Journal of Information and Communication* ISSN 1859-3526, kỳ 3, Vol V-2, No 16(36), page.50-57.

[CT3] Nguyen Thi Van, Truong Thi Thu Ha, Nguyen Xuan Huy (2017), "*Dependencies in Databases from a Logical Approach"*, *Proceedings of the 20th National Conference: Selected Issues in Information Technology and Communication*, Qui Nhon, 23-24/11/2017, Science and Technology Publishing House, ISBN: 978-604-67-1009-7, Ha Noi, page.260-265.

[CT4] Nguyen Xuan Huy, Nguyen Thi Van (2018), "*The Logic Inference Problem and Its Applications in Knowledge Bases", Proceedings of the 21th National Conference: Selected Issues in Information Technology and Communication*, Science and Technology Publishing House ISBN: 978-604-67-1104-9, Thanh Hoa, page.27-31.

[CT5] Nguyen Xuan Huy, Nguyen Thi Van, Truong Thi Thu Ha (2020), "*The relationship between partial dependency and positive generalization of Boolean dependency"*, *Journal on Information Technologies & Communications, ICT Research is a scientific publication of the Journal of Information and Communication*, ISSN 1859-3526, Vol 1, page.44-58.

[CT6] Nguyen Xuan Huy, Nguyen Thi Van (2022), "Lambda functions and approximate generalized positive Boolean dependencies", *Journal on Information Technologies & Communications*, *ICT Research is a scientific publication of the Journal of Information and Communication*, ISSN 1859-3526, Vol 2022, No 2, page.112-118.