울산대학교 연구논문집 제18권(자연과학·공학편) 제 1 호 pp. 25~32. 1987.
UOU Report Vol. 18. (Natural Science and Engineering) No. 1. pp. 25~32. 1986.
University of Ulsan.

The Database Design By the Mathematical Logic

Koh, Jac-Jin
Dept. of Computer Science
(Received September 30, 1986)

〈**Abstracts**〉

This paper presents the application of the mathematical logic to the design of databases.

The application is for the conventional databases and the deductive databases.

Mathematical logic provides a conceptual framework for the study of databases.

On the first part I concentrate on the interaction between logic and databases.

Logic can be used as a programming language, as a query language, to do deductive actions, to maintain the integrity of databases, to provide a formalism for handling negative information.

On the later part, I descrive the evolution of logic databases and the natural language query system.

數理論理에 의한 데이타베이스 設計

高 在 鎭
전 자 계 산 학 과
(1986. 9. 30 접수)

〈**요 약**〉

이 논문은 수리논리를 데이타베이스의 설계에 응용하는 것을 다룬다. 이 응용은 보통의 데이타베이스와 인연적 데이타베이스에 다 적용된다. 수리논리는 데이타베이스 연구를 위한 개념적인 틀을 제공한다. 처음 부분에서는 논리와 데이타베이스와의 상호관계에 대해서 중점을 둔다. 논리는 프로그래밍 언어로서, 질의어로서, 연연식 적용을 하기 위해서, 데이타베이스의 순수성을 보존하기 위해서, 부정 정보를 다루기 위한 형식을 제공하기 위해서, 사용되어 질수 있다. 뒷부분에서는 논리 데이타베이스의 형성과 자연어 질의 시스템에 대해서 기술한다.

I. Introduction

Mathematical logic provides a conceptual framework for database system.

This demonstrates how database concepts can be analyzed in terms of formal logic, and provides a characterization of the hypothetical worlds on which database system works.

This includes also the knowledge representation and how it relates to the process of deduction.

Another consideration of logic and database is how logic may be used to express constraints on a database integrity.

The fourth topic is that meaning assigned to

answers when negative questions are posed.

The last topic is how logic provides a unifying framework for query language.

In this paper we primarily consider relational type detabase.

Green [1969] used the logic for knowledge representation and manipulation.

His work was the basis of the study about a highly deductive manipulation of a small set of facts and an inferential mechanism.

Later they studied of database to handle large sets of facts, negative information and open and queries.

This has given rise to a deductive databases. Kuhns[1967] used logic for conventional databases.

Ⅱ. Databases and mathematical logic

A database is a collection of data that is stored permanently in a computer.

A *database system* is a *computer-based* record keeping system.

Therefore, a database can be thought as an *integrated and shared repository for stored data.*

By "integrated" we mean that the database is thought as a unification of several distinct data files.

By "shared" we mean that individual pieces of data in the database may be shared among several *different* users.

There are three types of database models.

They are relational model, network model and *hierarchical model.*

In this paper I will concentrate on the relational model.

1. Relational model

A domain is a finite set of values.

The Cartesian product of domains $D1, D2, \cdots, Dn$ is $D1\times D2\times\cdots\times Dn$ that is the set of all tuples $(x1, x2, \cdots, xn)$ such that for any i, $i=1, 2, \cdots, n$, $xi\in Di$.

a relation is any subset of the Carterian product of one or more domains.

A database instance is a finite set of finite relations. a finite relation is a relation whose extension is finite.

The extension of the relation is the totality of all tuples that can appear in a relation.

The arity of a relation $R\subseteq D1\times D2\times\cdots\times Dn$ is n.

A relation is a table of values.

The columns of this table have names.

These names are called attributes.

Values of the i-th attribute come from the domain Di.

A relation R with attributes $A1, A2, \cdots, An$ defines a relation scheme denoted as $R(A1, \cdots, An)$

Whereas the specific relation R is said to be an instance or extension of the relation scheme.

2. Mathematical logic

By a formal theory, we mean an artificial language in which the notions of "meaningful expression", axioms, and rules of inference are precisely described.

The language we are studying is called the object language, while the language in which we formulate and prove results about the object language is called the metalanguage.

As the object language we shall use a first order language such as that of the first order predicate calculus.

Primitive symbols of the first order predicate calculus are (1) parentheses, (2) variables, constants, functions, and predicate symbols, (3) the usual logical connectors ¬(not), &(and), ∨(or), → (implication), ↔ (equivalence), (4) quantifiers, ∀(for all), ∃(there exists).

A term is a constant, a variable, and if f is an n-ary function and $t1, \cdots, tn$ are terms, then $f(t1, \cdots, tn)$ is a term.

We assume that a term in the context of databases is either a constant or a variable,

If P is an n-ary predicate symbol and $t1, \cdots, tn$ are terms. then $P(t1, \cdots, tn)$ is an atomic formula.

An atomic formula or its negation is a literal.

An atomic formula is a well-formed formula.

If $W1$ and $W2$ are wffs*, then $\neg(W1)$, $(W1) \vee (W2)$, $(W1)\&(W2)$, $(W1) \rightarrow W2$ and $(W1) \leftrightarrow (W2)$ are wffs.

A wff is called a closed wff when it does not contain any free variable.

It contains only quantified variables and constants.

A statement form is an expression built up from the statemet letters $A, B, \cdots$ by appropriate application of the propositional connectives.

Each statement form determines a truth function.

There are two types of normal forms.

They are disjunctive normal form and conjunctive normal form.

A wff is in prenex normal form if all quantifiers appear in front of the formula.

For every wff we can construct an equivalent wff in prenex normal form.

A prenex formula is in Skolem normal form when all existential quantifiers are eliminated by replacing variables they quantify with arbitrary functions of all universally quantified variables that precede them in the formula.

These functions are called Skolem functions.

A Skolem function of 0 argument is called a Skolem constant.

A clause is a disjunction of literals, all of whose variables are implicitly universally quantified.

when a wff is in Skolem normal form, all the quantifiers may be eliminated since all variables are universally quantified.

$\neg A1 \vee \cdots \vee \neg Am \wedge B1 \vee \cdots \wedge Bn$

where Ai and Bj are positive literals,

is a clause.

It followes that

$A1\& \cdots \& Am \rightarrow B1 \vee \cdots \vee Bn$ is a clause.

Whenever n is equal to 0 or 1, the clause is said to be a Horn clause.

When $m=n=0$, the clause is called empty.

3. Databases viewed through logic

A relation can be defined intensionally as a set of general laws expressed as well-formed formulae in the first order predicate calculus.

The set of relational tuples is referred to as the extensional database (EDB) whereas the set of general laws is called the intensional data base (IDB).

The set of elementary facts that consist of the extensional data as an interpretation of a first order theory whose proper axioms are the general laws.

There are three assumptions that govern query evaluation and integrity constraint of data bases

(1) The closed world assumption (CWA)

facts not known to be true are assumed to be false.

(2) The unique name assumption

individuals with different names are different.

(3) The domain closure assumption

there are no other individuals than those in the database.

Queries involving negation are obtained by using the above hypotheses.

we consider as acceptable queries and integrity constraints only expressions that restrict their own reference domain.

A database can be considered from the viewpoint of logic in two different ways: an interpretation of a first-order theory and a first-order theory.

The interpretation view point and the theory viewpoint respectively formalize the concepts of conventional and deductive databases.

In the interpretation view point queries are considered to be formulas that are to be

* wff means well-formed formula

evaluated using the semantic definition of truth.

In the theory view point queries are considered to be theorems that are to be proved.

Ⅲ. Query languages

The reason for considering the language of logic as a basis for defining query languages is that a relational database (instance) can be viewed as an interpretation of a first-order language.

If we consider the formula $F(x1, \cdots, xr)$, where $x1, \cdots, xr$ are the free variables in F, as a query, the answer to this formula is the set of r-tuples $\langle e1, \cdots, er \rangle \subset D^r$ such that $F(e1, \cdots, er)$ is true.

A semantic characterization of formulas that can be considered to be reasonable queries led to the notion of definite formulas and safe formulas.

Such formulas are domain independent since they self-restrict the range of the variables that they contain.

Using many-sorted logic as a basis for defining query languages while considering elements of domains as primitive objects was exploited by Pirotte [1978].

Each sort is assigned to a data base domain, and the well formedness of formulas is checked with regard to sort requirements.

Logic supports powerful extensions to basic query languages in two direction: natural languages and programming languages. Querying databases using natural language has been a subject of active research.

When restricted to finite relations, the *relational calculus and the relational algebra* are both complete in the sense that for a given database they can express all relations whose extension is definable over the set of all domains of that database.

Many authors proposed that the relational calculus or relational algebra be embedded in a host language [Aho and Ullman 1979].

Such an embedding will allow a simple expression to define an operator, for example, transitive closure, from the primitive constructs of the host language. In some cases, all computable functions can be expressed: this is the ultimate notion of query language completeness.

Logic offers an alternative way to provide extensions such as those that motivate the embeding of the query language(logic) in a host programming language.

This extends the representation and manipulation capabilities of the database system itself.

This is precisely the idea of the deductive database system, where the database system is a theory, usually first order, with nonunit axioms.

Ⅳ. Integrity Contraints

Database consistency is enforced by integrity constraints, which are assertions that database instances (states) are compelled to obey.

There are two types of constraints: state constraints and transition constraints.

In the state constraints, there are two types of constraints: type constraints and dependency constraints.

Query language can be used to express integrity constraints.

In the integrity checking of deductive databases we use the general laws in databases.

A database can be considered to be an interpretation of a first order theory, where the nonlogical axioms are general laws perceived on the world.

The state laws deal with information in a single state of the world.

The transition laws deal with world evolution.

The state laws are interpreted as integrity rules by restricting valid states of the database to models of a first order theory which has

these laws, expressed by logical formulas, as proper axioms.

The transition laws constrain the consecutive states of the the database.

The database constituted by the set of elementany information is viewed as an interpretation of a first order theory whose proper axioms are the general laws. whenever a conflict between general laws and elementary informations is detected, the elementary informations are to be suspected first.

A formal system can be provided whose objects are the transactions. A syntax, semantics, and proof theory for reasoning about objects are needed.

A transaction is expressible in a programming language, including expressions used to define sets of data upon which the transaction acts.

The data definition language, viewed through its logic perspective, includes the integrity constraints as axioms of a first-order theory.

The transaction language then is embedded in regular programs, supported by a formal system, first-order dynamic logic.

Regular first-order dynamic logic is extended to a system called modal dynamic logic to reason about such programs.

Ⅴ. Dependencies

Data dependencies are special cases of integrity constraints that express structural properties of relations and permit relations to be decomposed, and retain certain properties.

An equivalence exists between some dependencies and a fragment of the propositional logic.

Functional dependencies and multivalued dependencies can be associated with propositional logic statements.

Functional dependencies behave precisely the same as a certain well-studied subset of propositional logic.

R: a database relation. If $A1, \cdots, Am$, $B1, \cdots, Br$ are among the column names of R, $A1, \cdots, Am$ determine $B1, \cdots, Br$ if whenever two tuples of R agree in columns $A1, \cdots, Am$, then they also agree in columns $B1, \cdots, Br$.

$A1, \cdots, Am \rightarrow B1, \cdots, Br$

we call each such statement a dependency statement with each column name A, we associate a distinct propositional variable A.

If the dependency statement is $A1, \cdots, Am \rightarrow B1, \cdots, Br$, the corresponding propositional statement is

$A1 \wedge \cdots \wedge Am \Rightarrow B1 \wedge \cdots \wedge Br$

t: a truth assignment

It is a mapping that assigns to each propositional variable either the value 0(false) or 1(true).

$A1 \wedge \cdots \wedge Am \Rightarrow B1 \wedge \cdots \wedge Br$ has truth value 0 if each of $A1, \cdots, Am$ has the truth value 1 and at least one of $B1, \cdots, Br$ has truth value 0, otherwise it has truth value 1.

DEP: a set of dependency statements

α: a single dependency statement

α is a consequence of DEP if α holds for every relation that obeys each dependency statement in DEP

ex. DEP: $AB \rightarrow C$, $AC \rightarrow D$

α: $AB \rightarrow D$

DEP: a set of implicational statements of propositional logic

α: a single implicational statement

α is a logical consequence of DEP if α has truth value 1 for every truth assignment that gives truth value 1 to each implicational stetement in DEP

ex. DEP: $A \wedge B \Rightarrow C$, $A \wedge C \Rightarrow D$

α: $A \wedge B \Rightarrow D$

Ⅵ. Incomplete information

Null values are a special case of the incomplete information. null values mean that attribute is applicable but values are unknown or value

does not exist.

An unknown value can be represented readily in a database, but problems arise with respect to its manipulation and interpretation in a query language.

A model-theoretic approach is to define a three-valued logic, based on the truth values true, false, undefined. Lipski [1979] defines information which can be extracted from a database in the presence of unknown values.

Data are said to be indefinite if they are of the form $P(a) \vee Q(b)$, and it is unknown whether $P(a)$ is true, $Q(b)$ is true, or both are true.

There are two types of null values: value at present unknown and property inapplicable.

There are two different interpretations of the query language. The external one, which refers the queries directly to the real world modeled in an incomplete way by the system and the internal one, under which the queries refer to the system's information about this world.

A mathematical model of a database with incomplete information is the information system which stores information concerning properties of some objects.

The information may be incomplete in that it may be known whether or not an object has a property.

A knowledge base is incomplete when it does not have all the information necessary to answer some question of interest to the system.

Ⅶ. Deductive databases

A deductive database is a database in which new facts may be derived from facts that were explicitly introduced. indefinite data are data in which one knows, say

$P(a) \vee P(b)$ is true, but one does not know if $P(a)$ is true, $P(b)$ is true, or both are true.

A deductive database is defined formally to be a database to consist of a finite set of constants, say $\{c1, \cdots, cn\}$ and a set of first-order clauses without function symbols.

Functions are excluded in order to have finite and explicit answers to queries.

The general form of clauses that will represent facts and deductive laws is

$P1 \& P2 \cdots \& Pk \rightarrow R1 \vee \cdots \vee Rq$

equivalent to

$\neg P1 \vee \cdots\cdots \vee \neg Pk \vee R1 \vee \cdots\cdots \vee Rq$

whenever any variable that occurs in the right-hand side of a clause also occurs in the left-hand side, the clauses is said to be range restricted.

(Type 1 : $k=0$, $q=1$)

$\rightarrow P(t1, \cdots, tm)$

(a) if the ti are constants, $Ci1, \cdot, Cim$

$\rightarrow P(Ci1, \cdots, Cim)$

which represents an assertion or a fact in the database. The set of all such assertions for the predicate letter P corresponds to a "table" in a relational database.

(b) When some, or all of the ti are variables, the clause corresponds to a general statement in the database

$\rightarrow$ancestor (adam, x)

adam is an ancestor of all individuals in the database.

(Type2 : $k=1$, $q=0$)

$P(t1, \cdots, tm) \rightarrow$

(a) When all of the ti are constants

$P(Ci1, \cdots, Cim) \rightarrow$

which stands for a negative fact.

(b) some of the ti are variables

An integrity constraint or "value does not exist" meaning for null values.

(Type3 : $k>1$, $q=0$)

$P1 \& \cdots \& Pk \rightarrow$

integrity constraints

"no individual can be both a father and a mother of another individual"

means

Father (x, y) & mother (x, y) ‣

(Type4 : $k \geq 1$, $q=1$)

$P1\&P2\cdots\&Pk\rightarrow R1$

an integrity constraint or

a definition of the predicate R, in terms of the predicate P1, $\cdots$, Pk. Such a definition is a deductive law.

(Type5 : $k=0$, $q>1$)

$\rightarrow R1\vee R2\cdots\vee Rq$

The indefinite assertion

(Type6 : $k\geq 1$, $q>1$)

$P1\&P2\cdots\&Pk\rightarrow R1\vee R2\cdots\vee Rq$

an integaity constraint or

the definition of indefinite data.

"Each individual has at most two parents"

means

Parent$(x,y)\rightarrow$mother$(x,y)\vee$Father(x,y)

The empty clause, where $k=0$, $q=0$, should not be part of database. We shall call a clause definite if its right-handside consists of exactly one atom.

all the types of clauses, except ground facts, are treated as integrity constraints in conventional databases.

In a deductive database some of them may be treated as deductive laws.

There are two classes of databases: definite detabases in which no clauses of either Type 5 or Type 6 apppear and indefinite databases in which such clauses do appear.

Ⅷ. Logic databases

Introducing functions into DB Horn clauses takes the deductive database field closer to the field of logic programming.

Horn clauses augmented with negation as failures led to the PROLOG language.

A PROLOG program is quite similar to a definite deductive database up to functions.

A logic database system wauld be obtained by combining query facilities and functions for integrity and maintenance of deduced facts with an efficient access method to a large number of facts.

A logic database language could be continuously extended, by providing extensions to negation as failures, incorporating metalanguage capabilities.

Ⅸ. Natural language query system

The natural language query system is a user oriented database query system.

There is a mismatch between the user's conceptualization of the data and its actual structure.

The use of natural language instead of an artificial query language is the only means to solve the mismatch.

Actual dialogs and end user experience are used to show the resulting increase in end user orientation.

The natural language query system accepts the user's natural language-English, French, or Korean, whatever the case may be.

The system translates the natural language to formal DBMS commands.

(Five types of information)

(1) The meaning of individual English words

(2) The syntactic structure of English

(3) Advice from the DBA

(4) The information contained within the database itself

(5) the user himself

The meaning of English words is represented within the system by actual program code.

The dictionary entry for one of these words contains its syntactic category and a pointer to this procedural definition.

The words with fixed built-in definitions are the common English words such as "is", "are", "not", "greater", ect.

The meaning of these words is independent of the contents of the database.

Other words are defined within the advice from the DBA or within the database itself.

The syntactic structure of English is used to

assemble the meanings of individual words into an appropriate semantic representation.

This syntax structure is represented as an augmented transition network.

An ATN is basically a set of finite state machines that can activate each other much like a subroutine call.

The transition from state to state in each machine is based on the syntactic category of the current word being analyzed.

The syntax is defined by constructing a finite state machine to parse each syntactic structure.

X. Conclusion

Logic is a powerful tool for the study of database.

It can be used as a query language, a programming language, to prove the correctness of programs, perform deductive searches, maintain the integrity of a database, provide a formalism for handling negation, generalize the concept of semantic network, and represent and manipulate data structures.

It also applies to query evaluation, database design through dependenices, representation and manipulation of deduced facts, and incomplete information.

The field "logic and databases" constitute the core of the work in the knowledge bases of the fifth generation computer.

There are many research areas that remain to be investigated.

In the near future, logic databases may be made practical and efficient.

References

(1) AHO, A.V., AND ULLMAN, J.D. 1979. Universality of data retrieval language. In proceedings of the 6th ACM symposium on Principles of Programming Languages (San Antonio, Tex., Jan. 29—31)., ACM, New-York. pp.110—120.

(2) GREEN, C. 1969. Theorem Proving by resolution as a basis for question-answering systems. In machine Intelligence 4, B. meltzer and D. michie, Eds. Elsevier north-Holland, New York, pp.183—205.

(3) KUHNS, J.L 1967. answering questions by computers-a logical study. Rand memo RM 5428 PR, Rand corp., Santa monica, calif.

(4) LIPSKI, W. 1979. On semantic issues connected with incomplete iinformation system. ACM Trans. Database syst. 4, 3 (Sept.), 262—296.

(5) PIROTTE, A. 1978. High level database query languages. In Logic and Databases, H. gallaire and J. minker Eds. Plenum, New York pp.409—436.