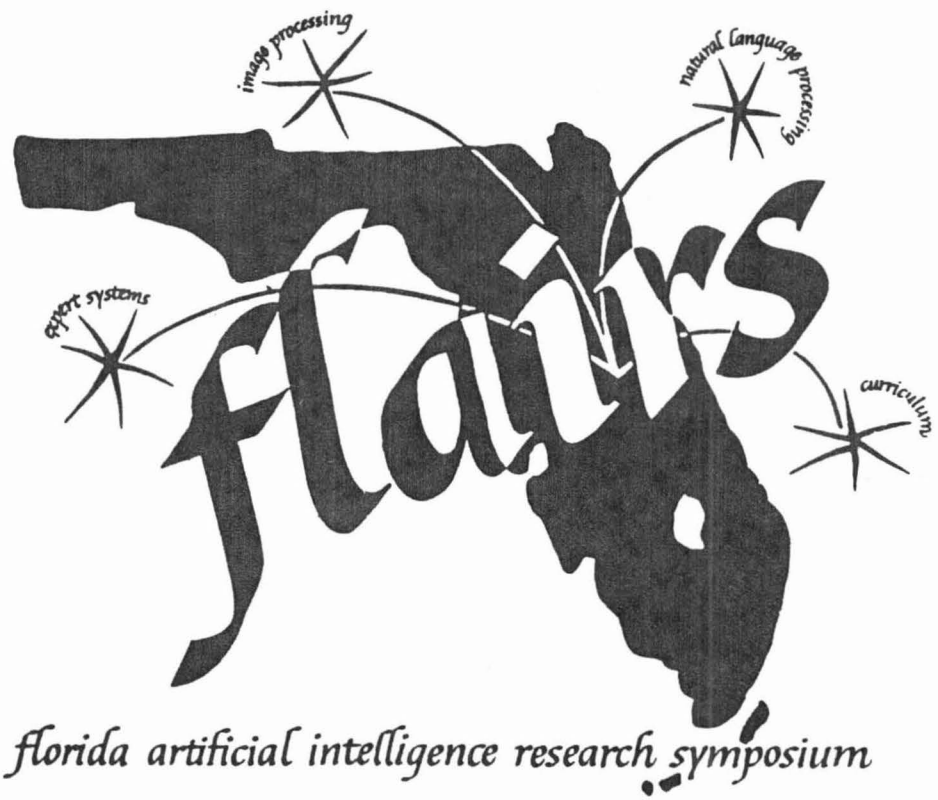
91-ML

Proceedings of the Fourth Florida Artificial Intelligence Research Symposium



Cocoa Beach, Florida
April 2-5, 1991

Mark B. Fishman
Editor

MACHINE LEARNING IN ACQUIRING CONSTRAINTS IN RELATIONAL DATABASE SYSTEMS

Wei Sun and Naphtali Rishe

School of Computer Science
Florida International University
Miami, Florida 33199
Internet: {weisun,rishen}@fiu.edu

1 ABSTRACT

We propose an approach to automatically acquire (dynamic) semantic integrity constraints from previously processed queries in relational database systems. By using these acquired constraints, query optimizer could be more intelligent in optimizing future queries. This automatic tool can also be an interactive tool for identifying and specifying static constraints that exist in applications. Previous results are extended by allowing θ-joins[20] in the specification of constraints and query qualifications instead of being restricted to only equi-joins.

2 INTRODUCTION

The use of inference rules in support of intelligent data processing plays a very important role in many areas of computer science. In database systems, *rules*, or *semantic integrity constraints*, are initially proposed for enforcing data integrity of database applications. It has been shown that these constraints can be used in database system to optimize users' queries so that query processing cost can be reduced. This is known as semantic query optimization, which is different from traditional query optimization strategies.

Semantic integrity constraints assert permissible or consistent database states (a database state is the stored data at a given time instance) for a database application. The semantics of data for a database application are captured or reflected by a set of semantic integrity constraints. For example, in a university, the regulation may require that all graduate students maintain a minimum grade-point-average (GPA) of 3.0; in a payroll database, the age of the youngest employee must be greater than 18 by law.

Semantic, or knowledge-based, query optimization has been proposed since early 80's [5, 8, 9, 6, 7, 2, 3, 4, 12] [15, 16, 17, 14, 13, 21]. The approach basically consists of transforming an original query submitted by an end-user into a semantically equivalent one (in the sense that they both yield the same answer) by exploiting the knowledge (the constraints) on the underlying application database. However, many problems remain unaddressed.

- Previous researchers assume equi-joins only [21]. We extend to θ-joins[20], that is, all comparators $\{<, \leq, >, \geq, =, \neq\}$ are allowed. An efficient algorithm is proposed to decide the implication of one join specification by another one and/or the equivalence between two join specifications[19]. Deciding the above problems is essential in reasoning using constraints [16, 21], deductive database systems and distributed database systems [18]. The proposed implication algorithm is efficient and generalizes earlier ones by allowing both θ-joins and/or selections.
- Constraints among data under the current database state, so called *dynamic constraints*, are ignored and not distinguished from the *static constraints* (which are true under any consistent database states). A

Proceedings of the 4th Florida Artificial Intelligence Research Symposium, 1991, by the Florida AI Research Society

c. 1991 FLAIRS 0-9620-1733-7/91/300-0145

larger class of queries can be optimized using dynamic constraints, together with static constraints, which capture the up-to-date view of the underlying data[21].

- Semantic integrity constraints need to be specified explicitly by users and/or data base administrators (DBAs) as assumed in literature. However, recent database applications tend to become larger and larger. As a result, it is usually difficult to identify all relevant integrity constraints. The proposed automatic knowledge acquisition is likely to ease the identification of constraints. More specifically, we will identify dynamically semantic integrity constraints through comparing and analyzing the results of queries previously processed. After identifying a potential constraint, the DBA/user can judge whether this constraint is always valid (as intended by the application) or not. If it is, then the constraint will be enhanced to be a static one, and not subject to invalidation in the presence of data base updates. Thus, our knowledge acquisition tool can be used as an interactive tool for static knowledge acquisition. This is very important if the database application is extremely large and complicated.

Knowledge acquisition has been known as machine learning in AI area. Meta_Dentral[11] learns molecular fragmentation rules by examining previous attempts to analyse molecule structure; and RX_Project[1] discovers rules about drug usage and patient's condition. Automatic knowledge acquisition in database systems has rarely been addressed. A group of senior DBMS researchers recently pointed out [13] "*Knowledge acquisition was a central and difficult problem in rule based applications. The sentiment of the workshop was that this task is extremely difficult.*"

Let all *statically semantic integrity constraints* for a database be denoted by S^s. S^s is usually specified explicitly by the DBAs or the users as intended by the application. By employing the knowledge acquisition tool, certain static constraints may be acquired after confirmation from users and/or DBAs. Let all *dynamically semantic integrity constraints* for a specific database state x be denoted by $S^d(x)$. Obviously, $S^d(x)$ may vary with the database state x; but S^s does not.

We assume the relational data model, and relations in the *first normal form* without null values. We also assume that a query $Q[T : q]$ or Q has a target (output) consisting of an attribute list T and a qualification q in conjunctive form, where x is a consistent database state, and q is: $p_1 \wedge p_2 \wedge ... \wedge p_n$, or equivalently written as a set $\{p_1, p_2, ..., p_n\}$. Each $p_k, 1 \leq k \leq n$, can be of either form $(R_i.X \text{ op } R_j.Y)$ or $(R.X \text{ op } c)$, where $\text{op} \in \{>, \geq, <, \leq, =, \neq\}$ and $c \in X$. The former is a $\theta-join$ [20] between relations R_i and R_j on attributes X and Y, respectively. The latter is a *restriction (a selection)* on the attribute R.X. All *join predicates* and *restrictions* in a set of predicates q are denoted as the set $JP(q)$ and $Res(q)$, respectively. Negated restrictions are not allowed, because negated restrictions can always be put in non-negated forms, namely:

$$\neg(R.X \ [>, \geq, =, \neq, <, \leq] \ c) \quad \equiv \quad (R.X \ [\leq, <, \neq, =, \geq, >] \ c)$$

Alternatively, Q[T: q] can be interpreted by the relational algebra as $\Pi_T UR(q)$, where $\Pi_T R$ is the projection of the relation R on the target attribute(s) T. UR(JP(q)), called *the underlying relation* of the qualification q, is the relation obtained by performing all joins specified in q (i.e., from JP(q)); and UR(q), called the *restricted underlying relation* of the qualification q, is the relation obtained by applying all restrictions in q (i.e., Res(q)) to the relation UR(JP(q)). When $JP(q) = \emptyset$, i.e., no join specified, relation UR(JP(q)) is the base relation upon which the restrictions of q are defined. Scheme(q) denotes the scheme for relation UR(q). Obviously, both UR(q) and UR(JP(q)) are defined on the same scheme, and $UR(q) \subseteq UR(JP(q))$.

Definition 1 *A* **semantic integrity constraint** *(briefly, a constraint) is a logical implication $JC \wedge A \Longrightarrow B$, where JC is a conjunction of a set of $\theta-joins$, and A and B are conjunctions of restrictions. A and B must be evaluable on $UR(JC)$. The interpretation is: if we perform the join operations specified by JC to form $UR(JC)$, then when-*

ever a tuple in $UR(JC)$ satisfies the restriction(s) A, it must satisfy the restriction(s) B. JP, B and A are called the **join precondition**, *the* **right-hand side (RHS)** *and the* **left-hand side (LHS)** *of the constraint, respectively. Join predicates are not allowed to appear on the right-hand side of a constraint by this interpretation. JC and/or A can be null. If $JC = \emptyset$, the underlying relation is the relation on which restrictions A or B are defined; if $A = \emptyset$, then the evaluation of A on the underlying relation becomes unconditionally true (i.e., the RHS should always been enforced). []*

The following are three sample constraints to be enforced in a university database which consists of three relations **student(s), student_course(sc),** and **course(c).** We note that it is possible to specify non equi-joins, although only equi-joins are used.

Domain assertion: "The lowest student GPA is 1.8":
$\Longrightarrow (s.GPA \geq 1.8)$

Restrictions within a relation: "The lowest GPA for a graduate student is 3.0."
$s.Status =$ "*Grad*" $\implies s.GPA \geq 3.0$

Restrictions across relations: Course at $\geq$ 500 level are only for graduate students.
$s.ss\# = sc.ss\# \wedge sc.c\# = c.c\# \wedge$
$c.Level \geq 500 \implies s.Status =$ "*Grad*"

3 The Implication of Joins

In this section, we discuss how different sets of joins may be equivalent, or one implies the other. For example, $(R_1.X = R_2.Y \wedge R_2.Y = R_3.Z)$ is equivalent to $(R_1.X = R_3.Z \wedge R_2.Y = R_3.Z)$, because they specify the same join operation although they are in different forms. $R.X < S.Y$ is implied by $\{R.X \leq A.B, A.B \leq S.Y, R.X \leq E.F, E.F \leq S.Y, A.B \neq E.F\}$.

The equivalence of two sets of join specifications A and B can be decided by testing the implication of A by B and the implication of B by A. Therefore, it is sufficient for us to focus on the implication problem. Deciding the implication of two join predicate sets is important in knowledge acquisition (to be shown later) and reasoning using these constraints (for semantic query optimization and knowledge base maintenance [8, 16, 21]), as well as in deductive database systems [10, 20] and distributed database systems [18]. For example, in order to apply a constraint to deduce its right hand side as a consequence, one has to test whether the join precondition of the constraint is satisfied (by the joins specified in the query qualification). This is exactly the implication problem.

Klug[10] and Ullman[20] proposed an algorithm to solve the above implication problem (say, whether T is implied by S) in the context of deductive database systems. The time complexity of the algorithm was shown to be $O(n^3)$, where n is the number of join predicates in S. The essential part of the algorithm is to compute the *closure* S^+ (similarly defined as the closure of a set of fcunctional dependencies through a set of axioms – the Amstrong Axioms. And this set of axioms is proved to be *sound* and *complete*.) With the introduction of *database domain knowledge*, we generalize the above result by allowing that each inequality can be either a θ-join and/or a selection while maintaining the same time complexity[19].

4 Automatically Acquiring Constraints

In this section, we identify several situations in which dynamic constraints on the current database state can be extracted.

Proposition 1 *If q_i and q_j yield the same set of qualified tuples in the underlying relation, and the two join specifications $JP(q_i)$ and $JP(q_j)$ are equivalent, then the constraints $JC \wedge Res(q_i) \implies Res(q_j)$, and $JC \wedge Res(q_j) \implies Res(q_i)$ can be asserted for the database state x, that is, $S^d(x) = S^d(x) \cup \{JC \wedge Res(q_j) \implies Res(q_i),\ JC \wedge Res(q_i) \implies Res(q_j)\}$, where $JC = JP(q_i)$ (or equivalently $JC = JP(q_j)$).*

Proof: Let t be a tuple in $UR(q_i)$, that is, t satisfies both JP(q_i) and Res(q_i). Since $UR(q_i(x)) = UR(q_j(x))$, $Res(q_j)$ must evaluate to be true on t. Hence, $JC \wedge Res(q_i) \Longrightarrow Res(q_j)$ can be asserted for the database state

x by *Definition 1*. In the same way, we can prove that $JC \wedge Res(q_j) \Longrightarrow Res(q_i)$ can be asserted for the database state x. []

In most situations, comparing the qualified tuples of the underlying relations is neither efficient nor direct. Thus, it is important to relate the outputs of queries.

Proposition 2 *If* $Ans(Q_j[T : q_j(x)]) \subseteq Ans(Q_i[T : q_i(x)])$, *and* $T \stackrel{FD}{\Longrightarrow} Attr\_Set(q_i)$, *then* $JC \wedge Res(q_j) \Longrightarrow Res(q_i)$ *can be asserted for the database state x, where* $JC = JP(q_i) \cup JP(q_j)$, *and* $Attr\_Set(q_i)$ *is the set of attribute names involved in* $Res(q_i)$, *where* $Ans(Q_y[T : q_j(x)])$ *is the answer to* Q_y *and* $X \stackrel{FD}{\Longrightarrow} Y$ *is a functional dependency.*

Proof: Let t be a tuple in UR(JC) such that $Res(q_j)$ is evaluated to be true. We now show that Res(q_i) also evaluates true on t. Let $t_0 = \Pi_{Scheme(q_j)}t$ and $t_1 = \Pi_{Scheme(q_i)}t$. Obviously, $t_0 \in UR(q_j)$, $\Pi_T t_0 = t' \in Ans(Q_j)$, and $t_1 \in UR(JP(q_i))$. Since $\Pi_T t_0 = t' \in Ans(Q_j)$ and $Ans(Q_j) \subseteq Ans(Q_i)$, there exists a $t_2 \in UR(JP(q_i))$ where $Res(q_i)$ is true on t_2, and $\Pi_T t_2 = t' \in Ans(Q_i)$. By $t_1, t_2 \in UR(JP(q_i))$, $\Pi_T t_1 = \Pi_T t_2 = t'$ and $T \stackrel{FD}{\Longrightarrow} Attr\_Set(q_i)$, we have $\Pi_{Attr\_Set(q_i)}t_1 = \Pi_{Attr\_Set(q_i)}t_2$, i.e., the attribute values on t_1 and t_2 on which Res(q_i) is evaluated are the same. Therefore, $Res(q_i)$ should evaluate true on t_1, and hence on t because $t_1 = \Pi_{Scheme(q_i)}t$. []

The following three statements are direct, we state them without proofs.

Corollary 1 *Let* $Ans(Q_i[T : q_i(x)]) = Ans(Q_j[T : q_j(x)])$, $x \in D$. *If* $T \stackrel{FD}{\Longrightarrow} Attr\_Set(q_i)$, *then* $JC \wedge Res(q_j) \Longrightarrow Res(q_i)$ *can be asserted for the database state x, where* $JC = JP(q_i) \cup JP(q_j)$. *If* $T \stackrel{FD}{\Longrightarrow} Attr\_Set(q_j)$, *then* $JC \wedge Res(q_i) \Longrightarrow Res(q_j)$ *can be asserted for the database state x.*

Proposition 3 *If* $Ans(Q_i[T : q_i(x)]) \cap Ans(Q_j[T : q_j(x)]) = \emptyset$, *then both* $JC \wedge Res(q_i) \Longrightarrow \neg Res(q_j)$ *and* $JC \wedge Res(q_j) \Longrightarrow \neg Res(q_i)$ *can be asserted for the database state x, where* $JC = JP(q_i) \cup JP(q_j)$.

Corollary 2 *If* $Ans(Q[T : q(x)]) = \emptyset$, *then* $JP(q) \Longrightarrow \neg Res(q)$ *can be asserted for the database state x.*

Prop 3 is different from *Prop 2* and *Cor 1* in the way that no functional dependency is required. Certain propositions need to compare the results of different queries in order to make a deduction. In realistic situations, only the answers of the queries submitted by a user within a user session are stored. It is reasonable to expect that certain queries submitted by a user within a user session are more likely to be related, thus may satisfy the hypothesis of the above results. For example, if a user is interested in finding all "brilliant" students, he may use different criteria to retrieve them. These criteria or qualifications are likely to satisfy the conditions in *Prop 2* or *Cor 3*. Steady increase of disk capacity and decrease in hardware cost make mass storage for storing query results more affordable. In addition, comparing query results may be performed in the background, possibly in parallel by using a separate processor.

5 Conclusion

An automatic machine learning scheme is proposed to acquire dynamic semantic integrity constraints, which will likely ease the identification of constraints in large database applications and facilitate a larger degree of semantic query optimization. We extended the previous results by allowing general θ-joins instead being restricted to equi-joins. An efficient algorithm to decide the implication of two join specifications is proposed.

Due to space limit, we are unable to show how we can automatically acquire the three sample constraints in *Section 2* by applying the above propositions to a history of users' queries on the university database. Similar examples can be found in [21].

We are very grateful to Dr. Clement Yu for his contribution to the early work of this research.

References

[1] Blum, R.: Discovery, confirmation, and incorporation of causal relationships from a large time-oriented clinical database: the RX project. *Computers and Biomedical Research*, Vol. 15, pp. 164-187, 1982.

[2] Chakravarthy, U., Fishman, H., and Minker, J.: Semantic query optimization in expert systems and database systems. *Expert Database System.* ed. by L. Kerschberg, pp. 659-674. Benjamin/Cummings Publishing Company, INC. Norfolk, Virginia, 1986.

[3] Chakravarthy, U. S., Minker, J. K., and Grant, J. H.: Semantic query optimization: additional constraints and control strategies. *Expert Database System.* ed. by L. Kerschberg, pp. 345-379. Benjamin/Cummings Publishing Company, INC. Norfolk, Virginia, 1987.

[4] Chakravarthy, U., Grant, J., and Miner, J.: Logic-based approach to semantic query optimization, *ACM Transactions on Database Systems*, Vol. 14, No. 2, pp. 162-207, June, 1990.

[5] Hammer, M. and Zdonik, S.B.: Knowledge-based query processing. *Proc. of Sixth International Conference on Very Large Databases.* pp. 137-147, Los Angeles, CA, September, 1980.

[6] Jarke, M., Clifford, J., and Vassiliou, Y.: An optimizing Prolog front-end to a relational query system. *Proc. of ACM SIGMOD.* pp. 296-306. Boston, June, 1984.

[7] Jarke, M.: External semantic query simplification: a graph-theoretic approach and its implementation in Prolog. *Expert Database System.* ed. by L. Kerschberg. pp. 675-692. The Benjamin/Cummings Publishing Company, INC. Norfolk, Virginia, 1986.

[8] King, J. J.: Quist: a system for semantic query optimization in relational databases. *Proc. of the Seventh Very Large Databases Conference.* pp. 510-517, Los Angeles, CA, September, 1981.

[9] King, J. J.: *Query Optimization by Semantic Reasoning.* UMI Research Press, Ann Arbor, MI, 1984.

[10] Klug, A.: On conjunctive queries containing inequalities. *J. ACM.* Vol. 35, pp. 146-160, January, 1988.

[11] Lindsay, R., Buchanan, B., Feigenbaum, and E., Lederberg, J.: Applications of artificial intelligence for organic chemistry: The DENTRAL Project, MacGraw-Hill, New York, 1980.

[12] Malley, C. V. and Zdonik, S. B.: A knowledge-based approach to query optimization. *Expert Database System.* ed. by L. Kerschberg. pp. 329-343. The Benjamin/Cummings Publishing Company, INC. Norfolk, Virginia, 1987.

[13] McLeod, D.: 1988 VLDB panel on future directions in DBMS research: a brief, informal summary. *SIGMOD Record.* Vol. 18, pp. 27-30, March, 1989.

[14] Qian, X. and Smith, D.: Integrity constraint reformulation for efficient validation. *Proc. 13th VLDB.* pp. 417-425. Brighton, England, Sept., 1987.

[15] Shenoy, S. T. and Ozsoyoglu, Z. M.: A system for semantic query optimization. *Proc. of the ACM SIGMOD.* pp. 181-195. San Francisco, May 27-29, 1987.

[16] Shenoy, S. T. and Ozsoyoglu, Z. M.: Design and implementation of a semantic query optimizer. *IEEE Transaction on Knowledge and Data Engineering.* Vol. 1, No. 3, pp. 344-361. September, 1989.

[17] Siegel, M.: Automatic rule derivation for semantic query optimization. *Proc. of Second International Conf. on Expert Database Systems.* ed. by L. Kerschberg. pp. 371-386. The George Mason Foundation, Norfolk, Virginia, 1988.

[18] Sun, X., Kamel, N., and Ni, L.: "Solving Implication Problems in Database Applications", Proc. of ACM SIGMOD, May 1989, pp. 185-192.

[19] Sun, W. and Weiss, M.: An Efficient Algorithm for Implication Testing Involving Arithmetic Inequalities, Technical Report, School of Computer Science, Florida International University, September, 1990.

[20] Ullman, J. D.: *Principles of Database and Knowledge-based Systems.* Vols. I and II. Computer Science Press, College Park, Maryland, 1989.

[21] Yu, C. and Sun, W.: Automatic knowledge acquisition and maintenance for semantic query optimization. *IEEE Transactions on Knowledge and Data Engineering.* Vol. 1, No. 3, pp. 362-375. September, 1989.