Intensional Answers in Object-Oriented Database Systems

Yang Hee Kim

ABSTRACT

When processing a query in a conventional database system, a set of facts or tuples are usually returned as an answer. This also applies to object-oriented database where a set of objects is returned. Deductive database systems, however, provide the opportunity to obtain the answer of a query as a set of formulas, thereby reduce the costs to process the query, and represent its "intensional answers" in a more compact way independently of the database state. In this paper, by introducing rules into the object-oriented database systems and integrating the intensional query processing of deductive database systems into the object-oriented database systems, we make it possible not only to answer incomplete queries which are not able to be answered in conventional object-oriented database systems, but also to express the answer-set abstractly as the names of classes, which provides us better understanding of the answer.

Keyword : 객체지향 데이터베이스(Object-Oriented Databases), 클래스 계층(Class Hierarchy), 내포적 질의문 처리(Intensional Query Processing), 내포적 답(Intensional Answers), SLD-분석(SLD-resolution)

1 Introduction

Conventionally, a query to an object-oriented database (OODB) system is answered only by the set of objects that satisfy a given query [1]. These objects may belong to different classes within a class hierarchy or they may be different complex objects somehow related to each other.

Compared to an extensional answer, an intensional answer is a set of rules which characterizes the conditions that an object must satisfy in order to belong to an answer to a certain query. This formula representation of the output of a query is independent of particular circumstances in the database. Not only does an intensional answer represent the answer to the given query in a more compact way, but it can be computed much faster than extensional answers. Most of the time intensional answers can be computed only using the rules without accessing the database. For a detailed description of the intensional query processing, we refer to [13].

In this paper, we introduce rules into OODB systems and apply the intensional query processing (IQP) techniques to OODB systems. All the query languages in OODB systems known to us are not able to incomplete queries. An incomplete query is a query on the attribute which belongs to a subclass but not the base class. In this paper, we make it possible to answer incomplete queries by representing OODB schema in terms of rules. Conventionally, the answer-set of a query in OODB is represented as a set of objects. But, the presence of semantics in OODB schema and IQP...
methodologies enable us to express the answer-set abstractly as names of classes. In this paper, we present an algorithm to obtain abstract representation of a given answer-set. It provides us better understanding of the answer.

Rules which represent OODB schema consists of structural rules and subclassing rules. The structural rules in turn consist of "IS_A"-relationships representing the class hierarchy. The subclassing rules represent characteristic properties of subclasses. We then transform all the rules to non-recursive Horn clauses and get an intensional answer by using SLD-resolution.

We provide some sample queries to show the advantages of an intensional answer to give query in an OODB system over conventional answers.

We organize this paper as follows. In section 2, we discuss several researches which has been done in the IQP. In section 3, we review the definition of intensional answers and methods for deriving intensional answers. In section 4, we look at the "IS_A"-relationship in OODB systems and the rules in OODB systems which are necessary in order to use the IQP. In section 5, we formalize methods to derive intensional answers for a class hierarchy model and give a detailed example. In section 6, we give conclusions and some remarks for possible extensions of the method in this paper.

2. Researches on Intensional Answers

For last several years, a lot of works have been done in this area. While each adopts different approaches, all have the common goal, that is to answer queries more abstractly. But research which integrates this area and OODB was started few years ago.

A deductive database is composed of extensional predicates (facts) and intensional predicates (rules). Cholvy and Demolombe [4] provided answers to queries that are independent of a particular set of facts, i.e., answers that are derived only from the rules. They also developed a method for generating answers using resolution.

Imielinski [6] tried to incorporate intensional predicates as a part of the answers and Pascual and Cholvy [9] restricted types of rules in intentional databases (IDB) to Horn clauses which are most widely adopted and studied in deductive database systems.

Motro [7] discussed a method that applies database constraints to generate intensional answers and Motro and Yuan informally discuss a simple query language incorporating intensional queries in [7].

Yoon and Song [13] [14] used only Horn-clauses for intensional databases and use a SLD-resolution which takes advantages of Horn-clauses. They introduced the notions of extended term-restricted rules, relevant literals and relevant clauses to avoid generating certain meaningless intensional answers. Also, Yoon and Song [15] introduced a partially automated method for generating intensional answers to represent answer-set abstractly for a query by extending current data mining techniques.

3. Definition of Intensional Answers

3.1 Definition

Cholvy and Demolombe [4] give a formal definition of query answers. Define $T$ as the database theory consisting of a set of facts and rules and let $Q(X)$ be a query where $X$ is a tuple of free variables. Then, the intensional answer $\text{ANS}(Q)$ to a certain query $Q(X)$ is defined as follows:

$$\text{ANS}(Q) = \{ \text{ans}(X) : \; T \vdash \forall X (\text{ans}(X) \rightarrow Q(X)) \}$$

where $\text{ans}(X)$ is a literal.

However, we want to restrict the answers within a defined domain of interest. Here are some extension on the intensional answer set.

So, let $DP = (P_1, \ldots, P_n)$ be a set of predicate symbols either of the IDB of the extensional database (EDB). And let $L(DP)$ be the first order language whose predicate symbols are $P_1, \ldots, P_n$. Then define an intensional answer $\text{ANS}(Q, DP)$ to the query $Q(X)$ by:

$$\text{ANS}(Q, DP) = \{ \text{ans}(X) : \; \text{ans}(X) \in L(DP) \text{ and } \; T \vdash \forall X (\text{ans}(X) \rightarrow Q(X)) \text{ and } \; (\text{ans}(X) \text{ is not the negation of a tautology) and } \; (\text{each ans}(X) \text{ is not redundant}) \}$$

3.2 Method for deriving answers

In the previous section, we defined an intensional answer. We note that:

$$T \vdash \forall X (\text{ans}(X) \rightarrow Q(X))$$

$$\vdash T \cup \{ \neg (\forall X (\text{ans}(X) \rightarrow Q(X))) \} \text{ is inconsistent}$$

Let $S$ be a set of clauses that represent the standard clausal form of $T$ axioms [3]. Then

$$\exists X (\text{ans}(X) \land \neg (Q(X)))$$
leads the standard form: \(\text{ans}(X_b) \land \neg(Q(X_b))\) where \(X_b\) is a tuple of Skolem constants. So, the above formula is equivalent to

\[ S \cup (\text{ans}(X_b), \neg(Q(X_b))) \text{ is inconsistent.} \]

Therefore, answer formulas \(\text{ans}(X)\) are such that resolution on \(S \cup (\text{ans}(X_b), \neg(Q(X_b)))\) leads to the empty clause.

However, initially we do not know what \(\text{ans}(X_b)\) are. So, for the resolution processing, we will start with \(S \cup (\neg(Q(X_b)))\). After resolving \(S \cup (\neg(Q(X_b)))\) will result in a resolvent \(R(X_b)\) and then resolving \(R(X_b)\) together with \(\text{ans}(X_b)\) will result in the empty clause. That means that \(\text{ans}(X_b)\) must equal to \(\neg R(X_b)\).

4. Class Hierarchy and Rules for abstract expression

4.1 Class Hierarchy

We look at a class hierarchy, representing "IS.A"-relationships between the different classes. We assume the following query syntax in our context:

- SELECT <attribute of target record type>
- FROM <object variables>
- WHERE <predicate>

The class hierarchy represented in the following Figure will be our object-oriented example database.

A typical incomplete query for the class hierarchy in (Figure 1) is the following:

\[
\text{SELECT VEHICLE.id}
\text{WHERE speed} > 50
\]

According to query languages in OODB systems known to us, the above query is incorrect. But by integrating intensional query processing into OODB system, we can answer such a query.

In a conventional OODB we get back a set of vehicle ids where the speed of the vehicle is greater than 50 miles per hour. According to our sample database in (Figure 1), we get back the following set:

\{HSCI, HSC2, ..., NSC1, NSC2, ..., HSS1, HSS2, ...\}

All these objects belong to different subclasses of the base class object. In order to find all the applying objects, the database must provide a technique to search through all the subclasses of vehicle. But this is not the common "State of the Art" in OODB systems right now. There do not exist good query languages for OODB systems which are simple to use the advantages of an object-oriented system.

By using semantics in OODB schema and intensional query processing methodologies, the answer set of a query is given by not a set of objects but names of classes to which answer object belong. Since these abstract representation of the answer set is more concise, it provides us better understanding of the answer set.

In our example there are the following intensional answers:

- intensional-answer 1 = all high_speed_cars
- intensional-answer 2 = all normal_speed_cars
- intensional-answer 3 = all high_speed_ships

In the next section of this paper, we will look at a way to automatically access the desired subclasses without being aware of the exact structure of the class hierarchy.

3.2 Rules

Introducing the notion of rules we can distinguish between different kinds of rules:

- integrity constraints
- "usual" rules
- structural rules
- subclassing rules

First, an object-oriented database may have integrity constraints expressed as rules. These are not unique for
object-oriented systems, but can be found in any database system. So, we are not interested in them here.

The second sort of rules are the "usual" rules that any deductive database can contain. Also these rules can be used the same way in an object-oriented database system with rules as in deductive database systems. The rules don't depend on dealing with objects or with tuples. So the occurrence of this type of rules is also not something which is special or requires a different treatment in object-oriented database than it requires in deductive databases.

The more important rules we have to be concerned about are the rules that has to be established in order to complete an intensional query successfully within an object-oriented database system. These rules come out of the class hierarchy of the objects. The system maintains automatically a logical representation of the schema information. The rules we are interested in are the rules whose information is stored in the database schema. The rules concerning the structural information of the subclassing schema (the "IS_A"-relationship) are the subclassing rules.

The structural rules and subclassing rules of our given example database would be the followings:

- structural rules
  - "IS_A" - rules
    - IS_A (automobile, vehicle)
    - IS_A (watervehicle, vehicle)
    - IS_A (sportscar, automobile)
    - IS_A (family_car, automobile)
    - IS_A (ship, watervehicle)
    - IS_A (high_speed_car, sportscar)
    - IS_A (normal_speed_car, sportscar)
    - IS_A (high_speed_ship, ship)
    - IS_A (normal_speed_ship, ship)

- subclassing rules
  - high_speed_car (X) ← sportscar (X) ∧ speed (X, Y) ∧ greater (Y, 200)
  - normal_speed_car (X) ← sportscar (X) ∧ speed (X, Y) ∧ greater (Y, 100) ∧ less (X, 200)
  - sportscar (X) ← automobile (X) ∧ no_doors (X, Y) ∧ equal (Y, 2)
  - family_car (X) ← automobile (X) ∧ no_doors (X, Y) ∧ equal (Y, 4)
  - high_speed_ship (X) ← ship (X) ∧ speed (X, Y) ∧ greater (Y, 50)
  - normal_speed_ship (X) ← ship (X) ∧ speed (X, Y) ∧ less_eq (Y, 50)

5. Formalization of Intensional Query Processing in Object-Oriented Database Systems

In this section, we outline an algorithm deriving intensional answers for a class hierarchy model consisting of non-recursive Horn clauses. By limiting non-recursive clauses the algorithm can be terminated, and by Horn clauses efficient algorithm can be used.

5.1 Processing comparison literals

To compute intensional answers efficiently, subclassing rules should be represented in a proper form. Since testing satisfiability in first-order logic formula is undecidable, adopting first-order logic formula for managing subclassing rules is not desirable. Therefore, we need a subset of first order logic expressions which is powerful enough for expressing subclassing rules and in which the satisfiability problem can be processed efficiently.

Subclassing rules can be represented with the "simple predicates" of Eswaran et al. [7]. The BNF of simple predicate abbreviated by SP is as follows:

\[
<SP> ::= <SP> \land <SP> | <SP> \lor <SP> | \neg <SP> \\
| <predicates>
\]

\[
<predicates> ::= \\
| <comparison operator> (<variable name>, <constant>) \\
| <comparison operator> (<variable name>, <variable name>) \\
| <comparison operator> (<variable name>, <variable name> + <constant>)
\]

\[
<comparison operator> ::= equal | not_equal | greater \\
| greater_eq | less | less_eq
\]

Rosencrantz and Hunt showed that the satisfiability problem of the set of simple predicate is NP-hard [11]. But they showed that conjunctive not_equal free predicates (simple predicates that do not contain not_eq and \lor) can be solved in polynomial time. We can represent a large class of subclassing rules with conjunctive not_equal free predicates. The following algorithm changes a conjunctive not_equal free predicate to a weighted directed graph [11], [7].

<table>
<thead>
<tr>
<th>Input</th>
<th>A conjunctive not_equal free predicate P.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>A weighted directed graph.</td>
</tr>
<tr>
<td></td>
<td>( w_1 ) and ( w_2 ) stand for variables and ( c ) stands for a constant.</td>
</tr>
</tbody>
</table>
The next algorithm will test comparison literals using a weighted directed graph and return a truth constant or a simplified predicate.

\[ \text{Algorithm 1} \]

There are two restrictions in the above algorithm. The one is that each variable should be integer valued. The other is that predicates can not have \text{not_equal} operators. Fortunately, many of subclassing rules involve integer valued domains such as engine size, price, number of doors, etc. And in this paper, we will deal with \text{not_equal} free predicates.

The next algorithm will change a weighted directed graph \( G \) with no multiple edges to a conjunctive \text{less_equal} predicate (\text{not_equal} free predicate that contains only \text{less_equal}).

\[ \text{Algorithm 2} \]

We can use Floyd's all shortest path algorithm to see if the graph has a negative weight cycles. In algorithm 3, the step 1 can be processed in a linear time, if part of the step 2 (Floyd's all shortest path algorithm) takes \( O(k^3) \) and else part of the step 3 can be processed in a linear time where \( k \) is a number of node in \( G \).

\[ \text{Algorithm 3} \]

5.2 Strategies for Intensional Answers

In Section 3.2, we proposed structural rules and subclassing rules for class hierarchy. Before we get the intensional answers, first of all some rule transformations should be done in order to get unique intensional literals, secondly recursion in subclassing rules should be removed, and "IS_A"-rules to the first order logic should be changed.

Unique intensional literals are literals that are either extensionally or intensionally defined but not both. So if we have literal \( p \) which is both EDB-defined and IDB-defined, then rename the extensional literal \( p' \) and introduce a new rule \( p \leftarrow p' \) in the IDB. In doing so, we can handle complete queries as well as incomplete queries for the intensional query processing.
Since structural rules and subclassing rules are conjunction in IDB, we can remove recursion in subclassing rules.

Finally we can change "IS_A"-rules to the first-order logic since the semantic of "IS_A" is implication. For example, $IS_A(X, Y)$ can be changed $Y \dashv \vdash X$. Now, IDB corresponds to a set of non-recursive Horn clauses.

The following algorithm will compute intensional answers from a set of non-recursive Horn clauses consisting of $EDB \cup IDB$ and a query $Q(X)$.

<table>
<thead>
<tr>
<th>Algorithm 4</th>
</tr>
</thead>
</table>

Frist we rename the extensional literal, add new rules in the IDB, remove recursion, and change "IS_A"-rules to the first order logic:

<table>
<thead>
<tr>
<th>EDB and IDB schema looks as follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• EDB</td>
</tr>
<tr>
<td>vehicle (id, weight, color)</td>
</tr>
<tr>
<td>automobile (id, weight, color, size, gas_mileage, no_doors)</td>
</tr>
<tr>
<td>family_car (id, weight, color, size, gas_mileage, no_doors, no_seats)</td>
</tr>
<tr>
<td>sports_car (id, weight, color, size, gas_mileage, no_doors, speed)</td>
</tr>
<tr>
<td>high_speed_car (id, weight, color, size, gas_mileage, no_doors, speed)</td>
</tr>
<tr>
<td>normal_speed_car (id, weight, color, size, gas_mileage, no_doors, speed)</td>
</tr>
<tr>
<td>water_vehicle (id, weight, color, level, speed)</td>
</tr>
<tr>
<td>ship (id, weight, color, level, speed, size)</td>
</tr>
<tr>
<td>high_speed_ship (id, weight, color, level, speed, size)</td>
</tr>
<tr>
<td>normal_speed_ship (id, weight, color, level, speed, size)</td>
</tr>
<tr>
<td>• IDB</td>
</tr>
<tr>
<td>&quot;IS_A&quot;-rules</td>
</tr>
<tr>
<td>subclassing rules</td>
</tr>
</tbody>
</table>

5.3 Example

To show the application of the algorithm introduced in the above section, we will use our example database given in (Figure 1).

<table>
<thead>
<tr>
<th>(Algorithm 4)</th>
</tr>
</thead>
</table>
watervehicle(X) ← ship(X)
sportscar(X) ← high_speed_car(X)
sportscar(X) ← normal_speed_car(X)
ship(X) ← high_speed_ship(X)
ship(X) ← normal_speed_ship(X)

• subclassing rules
  high_speed_car(X) ← speed(X, Y) ∧ greater(Y, 200)
  normal_speed_car(X) ← speed(X, Y) ∧ greater(Y, 100)
  ∧ less(Y, 200)
sportscar(X) ← no_doors(X, Y) ∧ equal(Y, 2)
  family_car(X) ← no_doors(X, Y) ∧ equal(Y, 4)
high_speed_ship(X) ← speed(X, Y) ∧ greater(Y, 50)
  normal_speed_ship(X) ← speed(X, Y) ∧ lesseq(Y, 50)

• new rules
  vehicle(X) ← vehicle'(X)
  automobile(X) ← automobile'(X)
  family_car(X) ← family_car'(X)
sportscar(X) ← sportscar'(X)
  high_speed_car(X) ← high_speed_car'(X)
  normal_speed_car(X) ← normal_speed_car'(X)

Now let us consider the query "Find a set of vehicle ids where the speed of the vehicle is greater than 50 miles per hour". The query can be written us

\[ Q(X) = \text{vehicle}(X) \land \text{speed}(X, Y) \land \text{greater}(Y, 50). \]

Thus the goal clause is

\[ \text{vehicle}(X), \text{speed}(X, Y), \text{greater}(Y, 50). \]
Since there are three branches that have intensional answers, we have following intensional answers:

- \( \text{ANS}_1^I(X) = \text{high-speed car}(X) \)
- \( \text{ANS}_2^I(X) = \text{normal-speed car}(X) \)
- \( \text{ANS}_3^I(X) = \text{high-speed ship}(X) \)

6. Conclusions and Remarks

In this paper, we developed a formalism to obtain intensional answers for a class-hierarchy model. By introducing rules into the OODB systems and applying the IQP techniques to the OODB systems, we are able to use the advantages of the semantics of OODB schema.

By using rules derived from the schema information, we are able not only to answer incomplete queries without knowing the exact structure of the database but also to express the answer-set abstractly as the names of classes. It provides us better understanding of the answer.

However, the method in this paper also has a disadvantage. For complete queries, the algorithm in this paper is less efficient than current query languages since our method answers a query only after it generates the complete resolution tree by using structural rules, subclassing rules, and new rules.

In this paper, we only obtain intensional answers for a class hierarchy model but not for a class-composition hierarchy. Our method does not seem to be powerful enough to represent a complex object hierarchy. It is probable that we need more powerful logic for reasoning intensional answers on complex objects.

References


