Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

CAPTURE DATABASE SEMANTICS BY RULE INDUCTION

Wesley W. Chu
Rei-Chi Lee
Kuorong Chiang

May 1990
CSD-900013

Capture Database Semantics by Rule Induction *

Wesley W. Chu, Rei-Chi Lee and Kuorong Chiang

Computer Science Department
University of California, Los Angeles
Los Angeles, California 90024

ABSTRACT

Database semantics can be classified into *database structure* and *database characteristics*. Database structure specifies the inter-relationships between database objects, while database characteristics defines the unique characteristics and properties of each object. To provide knowledge-based data processing, we need to gather and maintain knowledge about database characteristics. To capture the database characteristics, a Knowledge-based Entity Relationship (KER) Model that provides knowledge specification is proposed. A knowledge acquisition methodology is developed that uses machine learnine to induce the database characteristics knowledge from the database instances. Using a ship database as a test bed, a knowledge base has been generated by the proposed methodology. The induced knowledge has been used for semantic query optimization and data inference applications.

* This research is supported by DARPA Contract F29601-87-C-0072.

1. Introduction

The use of knowledge to support intelligent data processing has gained increasing attention in many database areas. For example, integrity constraints have been used as semantic knowledge to improve query processing [KING81, HAMM80]. Most of these works rely on human specified constraints and very few, if any, tools exist in gathering this knowledge.

In recent years, much effort has been devoted to the development of semantic data models [HAMM81, MCLE82, BORD84]. Most of these works emphasize structure modeling and allow for describing the database in a more natural way than traditional data models. These models are used mainly to capture the semantics of the database structure as described in the database schema. However, when a database is designed, most database designers use some *semantic rules* to distinguish among similar objects and specify the database schema according to these rules. In many cases, objects are often classified into different categories according to certain *characteristics* or *properties*. To distinguish them from the semantics of *database structure*, we shall refer to these semantics as *database characteristics* which are useful in knowledge-based data processing.

The type of database characteristics that are specified as integrity constraints [HAMM75] by human experts is not only time consuming to acquire but also is not too useful for knowledge-based data processing. To remedy this problem, we propose to use the database schema to guide the learning process and use the machine learning technique to induce database characteristics from the database. The database schema is specified in a knowledge-based Entity-Relationship (KER) Model which provides knowledge specification capability.

In this paper, we shall first discuss the problem of knowledge acquisition in databases. Next, we propose a model-based knowledge acquisition methodology that is based on the knowledge-based Entity-Relationship (KER) Model. We then present a rule induction algorithm and an example. Finally, we present the use of the induced rules for semantic query optimization, intensional query answering, and data inference applications.

2. Knowledge Acquisition in Databases

2.1 Knowledge Acquisition

The acquisition of knowledge is one of the most difficult problems in the development of a knowledge-based system. Currently, the acquisition of knowledge is still largely a manual process. The process usually involves a knowledge engineer who uses some expert system tools to transform the available knowledge into some internal form (knowledge representation) that is understandable by the expert system. It usually involves [MICH83]:

1. studying application literature to obtain fundamental background information,

2. interacting with the domain expert to obtain the expert level knowledge,

3. translating and encoding the expert knowledge for the system,

4. refining the knowledge base through testing and further interaction with the domain experts.

Such a manual process is very time-consuming. Further, even if the domain experts have the expertise, they may often not be able to describe their own expertise to others. As a result, useful knowledge may not be easy to collect. To remedy this problem, we propose to use the machine learning technique to construct the knowledge base. Rather than using knowledge engineers learning the application, or the domain experts learning the expert system tools and using their understanding of the application to construct the knowledge base, we propose to use machine learning technique to understand the database application and to create the knowledge base automatically.

2.2 Knowledge Acquisition by Rule Induction

Rule Induction [QUIN79, MICH83] is a technique of machine learning that has been used in AI research to induce rules from a set of training examples. For a given concept and a set of training examples representing the concept, find a description for the concept such that all

the positive examples satisfy the description and all the negative examples contradict the description. One approach is to examine the training examples simultaneously to determine which descriptors are most significant in identifying the concept from other related concepts. This approach recursively determines a set of descriptors that classify each example and selects the best descriptor from a set of examples based on a statistical estimation or a theoretical information content. The set of examples is then partitioned into subsets S_1, S_2, ..., S_n according to the values of the descriptor for each example. This technique is recursively applied to each S_i until each subset contains only positive examples so that the set of descriptors describes the example set. Although the automated approach speeds up the knowledge acquisition process, it has been used mainly in applications when the size of training examples is small. To apply this technique directly to a database would be too costly because the database usually consists of a very large volume of data. However, since a database schema is created by the designer based on the semantic characteristics of the application, and since semantic characteristics are the candidates for rule induction, we can use the database schema to guide the knowledge acquisition by machine learning and generate the rules automatically.

3. Model-based Knowledge Acquisition Methodology

3.1 The Knowledge-based Entity-Relationship (KER) Model

To enhance the modeling of such capabilities as type hierarchy and knowledge specification, we introduce a Knowledge-based E-R (KER) model, an extension of the Entity-Relationship Model [Chen76]. KER provides the following three generic constructs of data modeling:

1. **has/with** (*aggregation*) which links an object with another object and specifies a certain property of the object (e.g., a CLASS has an instructor);

2. **isa/with** or **contains/with** (*generalization/specialization*) which links an object type with another object type and specifies an object as a subtype of another object (e.g., PROFESSOR is-a subtype of PERSON or PERSON contains PROFES-

SOR, STUDENT, and STAFF);

3. **has-instance** (*classification*) which links a type to an object that is an instance of that type (e.g., "John Smith" is an instance of PROFESSOR).

Note that in addition to the semantic constructs provided by most semantic data models, KER also provides knowledge specification which is represented by the *with-constraint* information. Such knowledge specification associated with each database definition is useful for knowledge-based data processing.

In KER, an entity is a distinctly identified object, for example, a specific person, a department, or a course. An entity set is a collection of entities. Each of these entities is distinguished by a unique identifier. The set of unique identifiers is called the primary key of the entity set. A relationship specifies the connections between different entities. Conceptually, both entity type and relationship type can be considered as an object type and can be modeled using the **has/with** construct. For example, Figure 1 shows an object type SUBMARINE represented in KER.

object type SUBMARINE

has key:	ShipId	**domain:**	**char[10]**
has:	ShipName	**domain:**	**char[20]**
has:	ShipType	**domain:**	**char[4]**
has:	ShipClass	**domain:**	**char[4]**
has:	Displacement	**domain:**	**integer**

with Displacement **in** [2000..30000]

Figure 1. The KER representation of an object type SUBMARINE.

The object type can also be represented mathematically as:

$$\{ [a_1, a_2, ..., a_n] \mid a_1 \in D_1, a_2 \in D_2, a_n \in D_n \text{ with } \Psi \}$$

where each tuple $[a_1, a_2, ..., a_n]$ is an instance of such a type. Note that each a_i defines an attribute of the object type, and D_i specifies its attribute domain while Ψ states constraints on the allowable values the tuple can have. An attribute domain can also be an entity type. The system

provides a set of basic domains such as **integer**, **real**, **string**, and **date**. A more complex domain can be constructed from these basic domains. For example, we can define a domain AGE on the basic domain INTEGER with the range [0..200]. A BNF description of the KER model is given in the Appendix A.

A type hierarchy uses specialization/generalization constructs (**isa** or **contains** relationships) to define the subtype and supertype relationships. For example, SSBN (Ballistic Nuclear Missile Submarine) is a subtype of SUBMARINE, and CLASS-0101 is a subtype of SSBN, and therefore, a type hierarchy consisting of SUBMARINE, SSBN, and CLASS-0101 is formed (see Figure 2).

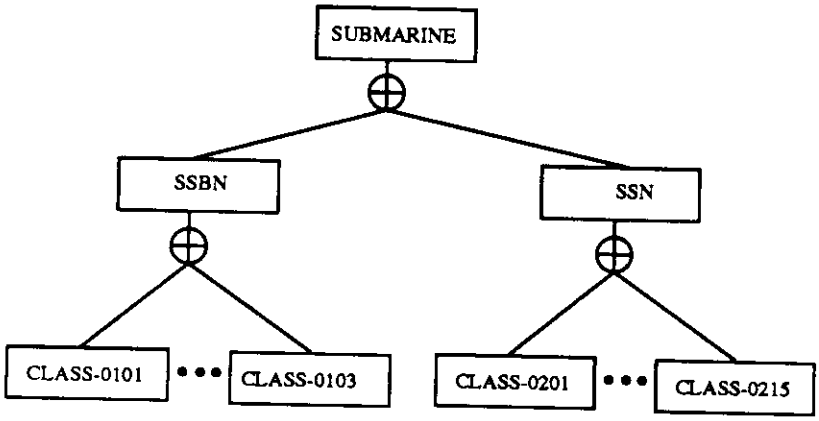

Figure 2. A Type Hierarchy SUBMARINE

A subtype inherits all the properties of its supertypes, unless some of the properties have been redefined in the subtype. For example, type SUBMARINE has attributes ShipID ,and ShipName, and type SSBN has attribute TypeID and TypeName; subtype CLASS-0101 will automatically inherit properties ShipID and ShipName from supertype SUBMARINE, and inherit properties TypeID and TypeName from another supertype SSBN.

A subtype can also be derived from another type by providing a *derivation specification*. For example, one can define a subtype SSBN (all the ships with ship type SSBN) of type SUBMARINE by specifying:

SSBN **isa** SUBMARINE **with** ShipType = "SSBN"

The with-clause defines the derivation specification of the subtype SSBN. It can also be considered as associating a constraint with this subtype.

The type hierarchy is represented in KER as:

E_1 **isa** E with Ψ_1

E_2 **isa** E with Ψ_2

...

E_n **isa** E with Ψ_n

or alternatively, it can also be represented as:

E **contains** $E_1, E_2, ..., E_n$ **with** Ψ.

This definition states that the instances of E can be divided into n disjoint subsets $E_1, E_2, ..., E_n$, with the constraint Ψ. Each E_i is a subtype of E.

To provide a graphical representation of the inter-relationships among the entity types/subtypes, relationship types, and derivation specification, we can extend the ER diagram by adding the type hierarchy with constraint representation as shown in Figure 3. A representation of a ship database schema by the KER Diagram is shown in Figure 4.

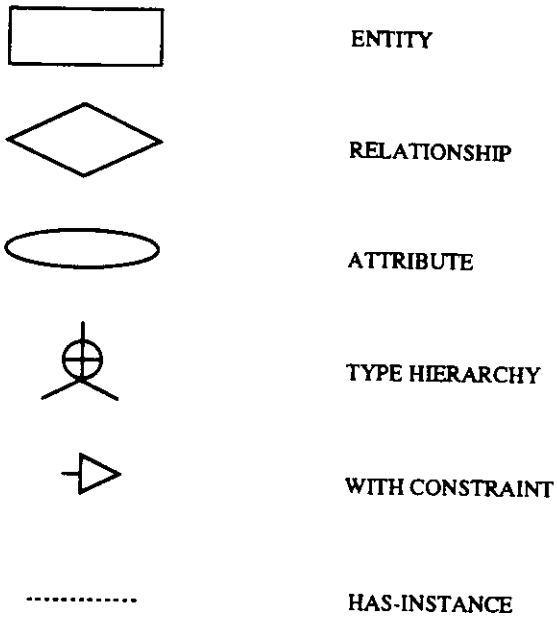

Figure 3. Components of the KER Diagram

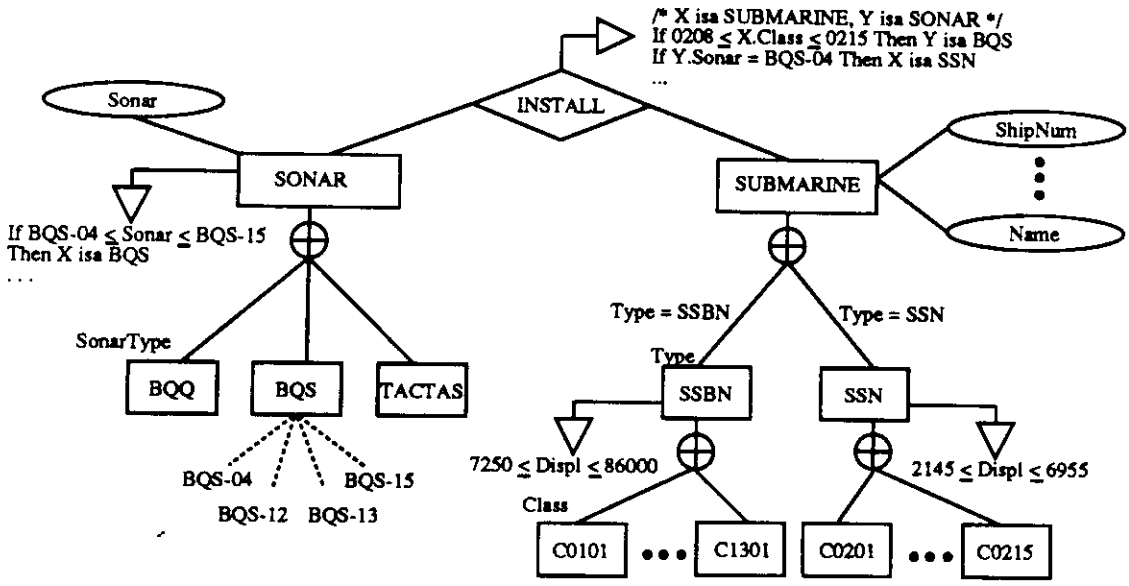

Figure 4. Representing the Ship Database Schema in KER Diagram

3.2 Classification of Semantic Knowledge

Semantic knowledge of a database can be divided into two categories: *enterprise knowledge* and *database knowledge*. Enterprise knowledge is the semantics of the database application. For example, the integrity constraint is the enterprise knowledge. Database knowledge is an instance of the enterprise knowledge which describes the current database contents. For example, enterprise knowledge specifies that the displacement of a ship must be greater than 2,000 tons, while database knowledge specifies that the displacements of the ships are between 2,360 tons to 81,600 tons. Thus, database knowledge is more effective for query optimization.

The semantic knowledge associated with each database are *domain knowledge*, *intra-object knowledge*, and *inter-object knowledge* as follows:

Domain Knowledge:

Domain knowledge defines specific properties of each entity set such as the attribute domains, value ranges, etc., and restricts the allowable instances of an entity set. For example, the displacement of a ship is in the range of 2,000 tons - 100,000 tons, and the displacement of an Attack Aircraft Carrier is in the range of 75,700 tons - 81,600 tons.

Intra-Object Knowledge:

Intra-Object knowledge specifies the relationships between attributes within an object type. For example, the object type submarine has the intra-object knowledge that if the displacement is less than 7,000 then it is a nuclear submarine (SSBN).

Inter-Object Knowledge:

The inter-object knowledge specifies the constraints that the instances of a relationship set must satisfy. For example, the relationship VISIT involves entities of

SHIP and PORT and satisfies the constraint that the draft of the ship must be less than the depth of the port. The inter-object knowledge can be induced from the interrelationship between SHIP and PORT linked by the VISIT relationship.

3.3 The Knowledge Acquisition Methodology

After classifying different types of knowledge and defining the target attributes for knowledge acquisition, let us now describe our Knowledge Acquisition Methodology (KAM), which consists of three stages: *schema generation, automated knowledge acquisition,* and *knowledge base refinement* as follows:

1. *Schema Generation*:

   The DBA uses the KER model to define a database schema. This step includes:

   a. Identifying entities and associated attributes.

   b. Identifying generalization hierarchies. If the database already exists, use the clustering indexes to define subtype entities. The indexes are the target attributes.

   c. Identifying aggregation hierarchy. Designate each of the referential keys as the target attributes. A referential key is the attribute of a relationship which is a key to some other entity.

2. *Automated Knowledge Acquisition*:

   a. Determine the *domain constraint* for each attribute of each entity type.

   b. Use the *rule induction algorithm* (described in the next section) to induce *inter-structure* and *intra-structure* knowledge related to the target attributes from the database.

3. *Knowledge Base Refinement*:

Based on the expert's own knowledge, domain experts can refine the rules in the knowledge base to improve system performance. Unlike the manual approach to knowledge acquisition, our methodology uses the database schema to guide the learning process to induce knowledge from the database contents. Such an automated process reduces the time for knowledge acquisition.

3.4 The Rule Induction Algorithm

We have implemented a rule induction algorithm in EQUEL (Embedded QUEL) and C on top of the INGRES system. Rule induction is performed using the relational operations to generate semantic rules for pairwise attributes. We shall present the algorithm that induces correlated relationships of the rule scheme X --> Y for attribute pair (X, Y).

Rule Induction Algorithm.

1. *Retrieving (X, Y) value pairs*

   Retrieve into S the instances of the (X, Y) pair from the database. The corresponding QUEL statement is:

```
range of r is relation
retrieve into S unique (r.Y, r.X)
sort by r.Y
```

2. *Removing inconsistent (X, Y) value pairs*

   Retrieve all the (X, Y) pairs that have multiple Y values for the same value of X. Let T be the result of this relation.

```
range of r is relation
range of s is S
retrieve into T unique (s.Y, s.X)
where (r.X = s.X and r.Y != s.Y)
```

   Remove all the (X, Y) pairs that have different Y values for the same X value from S.

range of s is S
range of t is T
delete s
where (s.X = t.X and s.Y = t.Y)

3. *Constructing Rules*

For each distinct value of Y in S, say y, determine the *value range* x of X and create a rule in the form of

if $x_1 \leq X \leq x_2$ **then** $Y = y$.

A value range is defined as a consecutive sequence of X values that occur in the database. The rules generated for the same attribute pair (X, Y) consist of the *rule set* designated by the rule scheme X --> Y. Note that when $x_1 = x_2$, then the rule reduces to

if $X = x$ **then** $Y = y$.

4. *Pruning the Rule Set*

Although storing more rules in the knowledge base provides more opportunities for inference, it also increases the overhead for storing and searching these rules. Therefore, when the number of rules generated becomes too large, the system must reduce the size of the rule set. In general, there are two criteria for rule pruning:

1. *Coverage Measure.*

   Each rule is satisfied by a certain number of database instances. Coverage specifies the number of database instances where the rule is satisfied.

   For applications such as semantic query processing, the higher the coverage measure of a rule, the higher the probability that this rule will be used. Thus, we keep these rules that have a coverage measure higher than a prespecified number N_c, which is a cut-off point for pruning the rules.

Based on this criteria, we remove rules from the knowledge base when the coverage measure is less than N_c. Selecting the N_c depends on the tradeoff between the applicability of the rules and the overhead for storing and searching these rules.

2. *Completeness Measure.*

Each rule scheme (X --> Y) contains a set of rules specifying the relationship between the attributes X and Y. The completeness measure is the sum of the coverage of the rules of the same scheme divided by the total number of (X, Y) value pairs in the database. If the completeness measure is equal to 100%, that means we can always find a rule (and Y's value) for each X's value.

For data inference applications, higher completeness measure enable us to infer more accurate answers. Therefore, for such applications, we want to keep the completeness measure of rule schema higher than a prespecified number C_c, which is a cut-off point for pruning the rules. Thus, we remove all the rule schema where completeness measure are less than C_c.

4. A Rule Induction Example

To experiment the proposed knowledge acquisition methodology, we shall use a ship database which was created by the System Development Corporation (now UNISYS) to provide a generic naval database based on [JANE81]. The database is currently running on INGRES on a Sun 3/60 machine. We shall use the nuclear submarine portion of the database which consists the following relations (a sample database instance is given in the Appendix C):

SUBMARINE = (*Id, Name, Class*)

CLASS = (*Class, ClassName, Type, Displacement*)

TYPE = (*Type, TypeName*)

SONAR = (*Sonar*, *SonarType*)

INSTALL = (*Ship*, *Sonar*)

The database consists of five entity types: SUBMARINE, CLASS, TYPE, SONAR, SONAR_TYPE and one relationship type: INSTALL. The three entity types SUBMARINE, TYPE, and CLASS form a ship hierarchy and the entities SONAR and SONAR TYPE form another hierarchy as shown in Figure 4. Each submarine type contains a set of submarine classes and each submarine class contains a set of submarine instances. For example, submarines are divided into two types: SSBN (Ballistic Nuclear Missile Submarine) and SSN (Nuclear Submarine). The SSBN ships contain three classes of ships: 0101 (Ohio), 0102 (Benjamin Franklin), and 0103 (Lafayette), and there are three ships that belong to the ship class 0103 (Lafayette).

Each ship class has its specific characteristics such as displacement, length, beam, etc.. For tactical or strategic reasons, different sonars are installed on different ships. The relationship INSTALL indicates the sonars installed on the different ships. A textual representation of the database schema is given in Appendix B.

Applying our knowledge acquisition technique to the ship database generates 17 rules as shown below (rules are grouped by object types):

(1) SUBMARINE

R_1: **if** SSN623 ≤ *Id* ≤ SSN635 **then** x **isa** C0103
R_2: **if** SSN648 ≤ *Id* ≤ SSN666 **then** x **isa** C0204
R_3: **if** SSN673 ≤ *Id* ≤ SSN686 **then** x **isa** C0204
R_4: **if** SSN692 ≤ *Id* ≤ SSN704 **then** x **isa** C0201

(2) CLASS

R_5: **if** 0101 ≤ *Class* ≤ 0103 **then** x **isa** SSBN
R_6: **if** 0201 ≤ *Class* ≤ 0215 **then** x **isa** SSN
R_7: **if** Skate ≤ *ClassName* ≤ Thresher **then** x **isa** SSN
R_8: **if** 2145 ≤ *Displacement* ≤ 6955 **then** x **isa** SSN
R_9: **if** 7250 ≤ *Displacement* ≤ 30000 **then** x **isa** SSBN

(3) SONAR

R_{10}: **if** BQQ-2 ≤ *Sonar* ≤ BQQ-8 **then** x **isa** BQQ

R_{11}: **if** BQS-04 ≤ *Sonar* ≤ BQS-15 **then** x **isa** BQS

(4) INSTALL (x isa SUBMARINE and y isa SONAR)

R_{12}: **if** SSN582 ≤ x.*Id* = SSN601 **then** y **isa** BQS
R_{13}: **if** SSN604 ≤ x.*Id* = SSN671 **then** y **isa** BQQ
R_{14}: **if** x.*Class* = 0203 **then** y **isa** BQQ
R_{15}: **if** 0205 ≤ x.*Class* ≤ 0207 **then** y **isa** BQQ
R_{16}: **if** 0208 ≤ x.*Class* ≤ 0215 **then** y **isa** BQS
R_{17}: **if** y.*Sonar* = BQS-04 **then** x **isa** SSN

For the intra-object relationships, we have found rules about the relationships between *Ship Id* and *Ship Class*; *Ship Class* and *Ship Type*; *Class Name* and *Ship Type*, and *Displacement and Ship Type*. The classification of the submarines into different classes and types is fairly stable, thus, these rules are stable as well. For the inter-object relationships, the following typical rules have been found: the submarines of the classes from 0205 to 0207 carry only the "BQQ" type of sonars (R_{15}); and the sonar "BQS-4" is only carried by the SSN (Nuclear Submarine) type of submarines (R_{17}); etc.

5. Applications

In this section, we shall illustrate the use of induced rules in such areas as semantic query optimization, intensional query processing, and data inference applications.

5.1 Semantic Query Optimization

Semantic query optimization uses the semantics to transform a given query to a new query. The transformed new query produces the same results as the original query but is more efficient to process [KING81, HAMM80]. Integrity constraints are commonly used as semantic knowledge for query transformation. However, due to the generality of the integrity constraints that describe all the valid states of the database instances, they are not effective for semantic query optimization.

Using the proposed rule induction, the relationships between the attributes as well as the inter-relationships between relations can be captured as semantic rules. These semantic rules provide more efficient semantic query transformation than the integrity constraints.

It is well-known that using a clustering index provides faster access to relations than that of a segment scan. Therefore, a clustering index is often used to answer the queries. Due to similar data characteristics, objects are often clustered together. If we select the clustering index as the *target attribute* during rule induction, useful classification rules representing the characteristics of each cluster can be induced. For example, using ShipType as a clustering index, we obtained the induced rules in Section 4. Based on R_9, the following query

Q: *"List the names of the submarines with displacement greater than 8,000."*

can be transformed to

Q': *"List the names of the SSBN submarines with displacement greater than 8,000."*

Processing **Q'** is much faster than **Q**, since processing **Q** requires a scan of the entire relation, while **Q'** can use ShipType as a clustering index.

5.2 Intensional Query Processing

Conventional query processing provides answers in the form of an enumeration of database instances retrieved from the database. Intensional query processing provides answers that characterize the instances satisfying the query rather than listing all the instances [SHUM88, MOTR89]. Traditionally, the knowledge about the database structure such as type hierarchy is used to derive intensional answers.

However, due to the limited semantics in database structure, using the type hierarchy alone can only generate very limited forms of intensional answers. The induced rules can be used to derive much more specific intensional answers. Based on the database schema, intensional answers can be derived from the induced rules by traversing down the type hierarchies of the object types as specified in the query. Such a technique is called *type inference* [CHU 90a]. For example, considering the query

"List the submarines with displacement greater than 8,000."

Since the condition "*Displacement* > *8000*" is subsumed by "*Displacement* ≥ 7250", based on the induced rule R_9, we can traverse down from the submarine hierarchy to derive an intensional answer for the query which is "SSBN".

5.3 Data Inference Applications

In a distributed database system, databases are often partitioned into fragments and replicated and stored at several sites. However, during network partition, the data fragments may be inaccessible which reduces data availability. Since database attributes are often related to each other, the values of certain attributes often can be *inferred* from other attributes. To improve data availability, we can use *data inference* to infer *inaccessible* data from *accessible data* [CHU 90b].

Using the proposed knowledge acquisition approach, correlated knowledge between attributes can be extracted from the database contents. Since these rules represent summarized information, the storage size of these rules is much less than that of the replicated copies of the data. Such induced rules can then be replicated at each site or certain critical sites to improve data availability during network partitions. For example, given the query,

"*Which submarines carry the BQS type of sonar?*"

To process the above query, we need the INSTALL relation. However, due to network partitioning, the INSTALL relation (See Section 4) is no longer available. As a result, we are unable to answer the query by conventional query processing. However, using the induced rule R_{12}, we can derive that the Id of the ships are in the range from "SSN582" to "SSN601". Then, from the SUBMARINE relation we can derive the following answer for the query:

{SSN582, SSN584, SSN592, SSN601}.

For more discussion on this, the readers should refer to [CHU 90b].

6. Conclusions

Database semnantics can be classified into database structure and database characteristics. Semantic data models emphasize the modeling structural aspects of the database, while the database characteristics define the unique characteristics and properties of objects which are important to knowledge-based data processing. A Knowledge-based Entity-Relationship (KER) Model is proposed to provide knowledge specification capability and to maintain the database semantic knowledge. A knowledge acquisition methodology is developed that is based upon the KER Model and machine learning techniques to acquire the database characteristics from the database. These database characteristics are useful for semantic query optimization and data inference applications.

REFERENCES

[BROD84] Brodie, M., Mylopoulos, J., and Schmidt, J. W., (eds.) *On Conceptual Modelling. Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer, New York, 1984.

[CHEN76] Chen, P.P.S., "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Transaction on Database Systems*, Vo. 1, No. 1, March 1976.

[CHU 90a] Chu, W. W, and Lee, R., "Using Type Inference and Induced Rules to Provide Intensional Answers," Technical Report CSD-9000006, Computer Science Department, UCLA, Los Angeles, 1990.

[CHU 90b] Chu, W. W, and et al, "An Inference Technique for Distributed Query Processing in a Partitioned Network," Technical Report CSD-9000005, Computer Science Department, UCLA, Los Angeles, 1990.

[HAMM75] Hammer, M., and McLeod, D., "Semantic integrity in a relational data base system," In *Proceedings of the First International Conference on Very Large Data Bases*, IEEE, New York, pp. 25-47, 1975.

[HAMM80] Hammer, M. and Zdonik, S. B., Jr., "Knowledge-based query processing," In *Proceedings of the 6th International Conference on Very Large Data Bases* (Montreal, Oct. 1-3). IEEE, New York, pp. 137-147, 1980.

[HAMM81] Hammer, M., and McLeod, D., "Database Description with SDM: A Semantic Database Model," *ACM Transactions on Database Systems*, Vol. 6, No. 3, September 1981.

[JANE81] *Jane's Fighting Ships*, Jane's Publishing Co., 1981.

[KING81] King, J. J., "QUIST: A system for semantic query optimization in relational databases," In *proceedings of the 7th International Conference on Very Large Data Bases* (Cannes, Sept. 9-11). IEEE, New York, pp. 510-517.

[MCLE82] McLeod, D., and Smith, J. M., "Abstraction in Database," *Proc. Workshop on Data Abstraction, Databases, and Conceptual Modelling, SIGMOD Record*, Vol. 11, No. 2, February 1981.

[MICH83] Michalski, R. S., et al, (eds.) *Machine Learning: An Artificial Intelligence Approach*, Tioga Press, Palo Alto, 1983.

[QUIN79] Quinlan, J. R., "Induction Over Large Data Bases", STAN-CS-79-739, Stanford University, 1979.

Appendix A. A BNF definition of the KER Model

We will use the following BNF conventions:

<...> non-terminal symbol

{ x } x appears 0 or more times

[x] x appears 0 or 1 time

x1 | x2 | ... | xn x1 or x2 or ... or xn

'1' literal symbol

A.1 Data Definition Statements

<KER definition> ::=

<domain definitions> |

<object type definitions> |

<type hierarchy definitions>

A.2 Domain Definition Statements

<domain definitions> ::=

<domain definition> {, <domain definition> }

<domain definition> ::=

domain <domain name> **is** <domain description>

[<domain sepcification>]

<domain name> ::= **identifier**

<domain description> ::= <standard domain> | <object domain>

<standard domain> ::= **string** | **integer** | **real** | **date**

<domain specification> ::=

<range specification> |

<set specification>

<range specification> ::=

range <lower boundary> <value> ',' <value> <upper boundary>

<lower boundary> ::= '[' | '('

<upper boundary> ::= ']' | ')'

<set specification> ::=

set of '{' <value> {, <value> } '}'

<value> ::= **identifier** | <integer> | <real>

<object domain> ::= <object type name>

<object type name> ::= **identifier**

A.3 Object Type Definition Statements

<object type definition> ::=

object type <object type name>

<attribute list>

<with constraints>

<attribute list> ::=

<attribute> {, <attribute> }

<attribute> ::=

has [**key**] ':' <attribute name> **domain** <domain name>

<with constraints> ::=

with <constraints>

A.4 Type Hierarchy Definition Statements

<type hierarchy definition> ::=

<object type name> **contains** <sub-type list>

[<attribute list>]

[<with constraints>]

<sub-type list> ::=

<object type name> {, <object type name> }

A.5 Constraint Definition Statements

<constraints> ::=

<constraint> {, <constraint> }

<constraint> ::=

<domain range constraint> |

<semantic rule>

<domain range constraint> ::=

<attribute name> **in** <domain sepcification>

<semantic rule> ::=

<constraint rule> |

<structure rule>

<constraint rule> ::=

if <premise> **then** <consequence>

<premise> ::=

<conjuntives>

<conjunctives> ::=

<clause> { **and** <clause> }

<clause> ::=

<attribute> <operator> **constant**

<consequence> ::=

<attribute> '=' **constant**

<structure rule> ::=

if <role definitions>

and <conjunctives>

then <variable> **isa** <object type name>

<role definitions> :: =

<role> { **and** <role> }

<role> ::= <variable> **isa** <object type name>

<variable> ::= **identifier**

Appendix B. A KER Representation of a Naval Ship Database Schema.

B.1 Domain Definitions

domain: NAME **isa** CHAR[20]

domain: CLASS_NAME **isa** NAME
domain: SHIP_NAME **isa** NAME

domain: TYPE_NAME **isa** CHAR[30]

domain: SONAR_NAME **isa** CHAR[8]

B.2 Object Type Definitions

object type CLASS

has key: Class **domain**: CHAR[4]
has: Type **domain**: type
has: ClassName **domain**: CLASS_NAME
has: Displacement **domain**: INTEGER

with /* constraint rules */

if "0101" ≤ Class ≤ "0103" **then** Type = "SSBN"
if "0201" ≤ Class ≤ "0216" **then** Type = "SSN"

CLASS **contains** SSBN, SSNs

Bwith /* x **isa** CLASS */

if 2145 ≤ x.Displacement ≤ 6955 **then** x **isa** SSN
if 7250 ≤ x.Displacement ≤ 30000 **then** x **isa** SSBN

object type SUBMARINE

has key: Id **domain**: CHAR[7]
has: Name **domain**: SHIP_NAME
has: Class **domain**: class

SUBMARINE **contains** C0101, ..., C1301

object type TYPE

has key: Type **domain**: CHAR[4]
has: TypeName **domain**: TYPE_NAME

object type SONAR

has key: Sonar **domain**: CHAR[8]
has: SonarType **domain**: SONAR-NAME

SONAR **contains** BQQ, BQS, TACTAS

with /* x **isa** SONAR */

if BQQ-2 ≤ x.Sonar ≤ BQQ-8 **then** x **isa** BQQ
if BQS-04 ≤ x.Sonar ≤ BQS-15 **then** x **isa** BQS
if x.Sonar = "TACTAS" **then** x **isa** TACTAS

object type INSTALL

has key:	Ship	**domain:** SUBMARINE
has:	Sonar	**domain:** SONAR

with /* x **isa** SUBMARINE and y **isa** SONAR */

if x.Class = 0203 **then** y **isa** BQQ
if 0205 ≤ x.Class ≤ 0207 **then** y **isa** BQQ
if 0208 ≤ x.Class ≤ 0215 **then** y **isa** BQS
if y.Sonar = BQS-04 **then** x **isa** SSN

Appendix C. A Ship Database and Its Induced Rules

A Ship Database:

Relation SUBMARINE

Id	Name	Class
SSBN130	Typhoon	1301
SSBN623	Nathaniel Hale	0103
SSBN629	Daniel Boone	0103
SSBN635	Sam Rayburn	0103
SSBN644	Lewis and Clark	0102
SSBN658	Mariano G. Vallejo	0102
SSBN730	Rhode Island	0101
SSN582	Bonefish	0215
SSN584	Seadragon	0212
SSN592	Snook	0209
SSN601	Robert E. Lee	0208
SSN604	Haddo	0205
SSN610	Thomas A. Edison	0207
SSN614	Greenling	0205
SSN648	Aspro	0204
SSN660	Sand Lance	0204
SSN666	Hawkbill	0204
SSN671	Narwhal	0203
SSN673	Flying Fish	0204
SSN679	Silversides	0204
SSN686	L. Mendel Rivers	0204
SSN692	Omaha	0201
SSN698	Bremerton	0201
SSN704	Baltimore	0201

Relation INSTALL

Ship	Sonar
SSBN130	BQQ-2
SSBN623	BQQ-5
SSBN629	BQQ-5
SSBN635	BQS-12
SSBN644	BQQ-5
SSBN658	BQS-12
SSBN730	BQQ-5
SSN582	BQS-04
SSN584	BQS-04
SSN592	BQS-04
SSN601	BQS-04
SSN604	BQQ-2
SSN610	BQQ-5
SSN614	BQQ-2
SSN648	BQQ-2
SSN660	BQQ-5
SSN666	BQQ-8
SSN671	BQQ-2
SSN673	BQS-12
SSN679	BQS-13
SSN686	BQQ-2
SSN692	BQS-15
SSN698	TACTAS
SSN704	BQQ-5

Relation TYPE

Type	TypeName
SSBN	ballistic nuclear missile sub
SSN	nuclear submarine

Relation SONAR

Sonar	SonarType
BQQ-2	BQQ
BQQ-5	BQQ
BQQ-8	BQQ
BQS-04	BQS
BQS-12	BQS
BQS-13	BQS
BQS-15	BQS
TACTAS	TACTAS

Relation CLASS

Class	ClassName	Type	Displacement
0101	Ohio	SSBN	16600
0102	Benjamin Franklin	SSBN	7250
0103	Lafayette	SSBN	7250
0201	LosAngeles	SSN	6000
0203	Narwhal	SSN	4450
0204	Sturgeon	SSN	3640
0205	Thresher	SSN	3750
0207	Ethan Allen	SSN	6955
0208	George Washington	SSN	6019
0209	Skipjack	SSN	3075
0212	Skate	SSN	2360
0215	Barbel	SSN	2145
1301	Typhoon	SSBN	30000