# Reverse Engineering Databases for Knowledge Discovery

**Stephen Mc Kearney**
Department of Computing
Bournemouth University
Talbot Campus, Fern Barrow
Poole, Dorset, UK
smckearn@bournemouth.ac.uk

**Huw Roberts**
Data Mining Group
pp13, MLB1
BT Laboratories
Ipswich, Suffolk, UK
huw.roberts@bt-sys.bt.co.uk

## Abstract

Many data mining tools cannot be used directly to analyze the complex sets of relations which are found in large database systems. In our experience, data miners rely on a well-defined data model, or the knowledge of a *data expert*, to isolate and extract candidate data sets prior to mining the data. For many databases, typically large legacy systems, a reliable data model is often unavailable and access to the data expert can be limited. In this paper we use *reverse engineering* techniques to infer a model of the database. Reverse engineering a database can be seen as knowledge discovery in its own right and the resulting data model may be made available to data mining tools as background knowledge. In addition, minable data sets can be produced from the inferred data model and analyzed using conventional data mining tools. Our approach reduces the data miner's reliance on a well-defined data model and the data expert.

## Introduction

Our experience of data mining large, complex databases has highlighted the importance of the role of the *data expert* in the data mining process (Roberts & Totton, 1996). A data miner often uses knowledge from both a domain expert, who knows about the world the data represents, and a data expert, who knows about how the data is structured and stored. By combining knowledge from both sources the data miner arrives at a *focus of interest* in the database from which minable data sets can be constructed. This occurs before conventional data mining tools are used. Without a data expert, large databases can be hard to understand since an accurate data model is often unavailable. In addition, getting access to an appropriate data expert may be difficult.

In this paper we describe a reverse engineering technique which produces data models of existing databases. This technique reduces the reliance of the data miner on the knowledge of the data expert by extracting knowledge directly from the database. In addition to constructing minable data sets, the method described can also extract useful *domain* knowledge. This can be seen as knowledge discovery in its own right or as a source of useful *background knowledge* usable by some data mining tools to guide their search.

While data mining has been used to support schema integration and reverse engineering in databases, there has been little or no investigation of using *reverse engineering to support data mining*. Specifically, we use reverse engineered data models to generate data sets which can be input to conventional data mining algorithms.

## Overview

Data mining a large relational database might start by generating a single large table containing all the data. Alternatively, a set of attributes may be selected based on the advice of a data expert or because they 'feel' like an appropriate set to be mined. The resulting table will contain a lot of attributes which may or may not contribute something to the data mining process. The missing component in this procedure is a method of systematically generating models of the data from which plausible relationships between entities in the database can be identified.

(McKearney, 1992) describes a method of building models of a database by analyzing relationships in the data. The resulting *abstract objects* describe relationships between object classes inferred from the data. Each abstract object depicts a relationship as one particular view of the data. A view describes the database in terms of individual data entities, for example, a relationship between employees and projects may be viewed as either an employee working on one or more projects or a project with one or more employees working on it. This work is similar to other reverse engineering efforts (for example, Springsteel & Kou, 1992) which mapped one data model schema, for example, the network model, to an equivalent (newer) schema, for example, the relational model (Boulanger & March, 1989).

Each abstract object can be converted to one or more database queries from which a data set may be produced and then analyzed.

## Method

The method has three stages: *dependency analysis, entity analysis,* and *object analysis.* Abstractions inferred during these stages can be mapped to a set of database queries which when executed generate minable data sets.

### Dependency Analysis

Attributes in a database relation can either describe concepts, *descriptive attributes,* or structure concepts, *structural attributes.* Structural attributes, which contain identifying data values, contain very little semantic knowledge about a concept and so a good knowledge discovery data set contains only descriptive attributes. The descriptive attribute *employee salary,* for example, contains more knowledge about an employee than the structural attribute *employee number.* Hence, it is important to remove *structural attributes* when preparing a data set. Classifying each attribute as descriptive or structural is possible when the dependencies between the attributes are known.

(McKearney, 1992) discusses a method of measuring the degree to which one attribute determines another, call *informativeness.* Informativeness is a measure of the dependency between attributes and may be used to infer unique or near-unique attributes in the database. Highly informative attributes may be used to infer entities and simple relationships in a database. The informativeness of attribute set $Y$ over attribute set $\Omega$ is defined as:

$$I(\Omega|Y)=\frac{\sum_{X_i\in\Omega} w(X_i)H(X_i|Y)}{\sum_{X_i\in\Omega} w(X_i)H(X_i)}$$

where $H(X_j)$ is the entropy of attribute set $X_j$ and $H(X_j|Y)$ is the conditional entropy of attributes set $X_j$ given attribute set $Y_j$. $w(X_j)$ is a domain dependent weighting for the attribute set $X_j$ which is normally $1/n$, where $n$ is the number of attributes in $\Omega$.

### Entity Analysis

The second stage in producing a model of the data is to group the database attributes into entities. Relationships between entities are represented by foreign keys between relations. We generalize the concept of a foreign key to that of a *reference* between relations. A reference from relation $r_1$ to relation $r_2$ is made by storing a highly informative attribute about $r_2$ in $r_1$. For example, a relation *car* references a relation *person* by storing *person-no* in each tuple of *car,* thus representing the fact that each person owns a car. Entities are inferred by analyzing these references.

Depending on the types of reference in which a relation participates, it can be classified as either an entity or a relationship (Batini et al, 1992). We use database design rules to classify each relation as either (i) an *entity,* when it is referenced via a highly informative attribute set, (ii) a *weak entity,* when it is referenced via part of a highly informative attribute set, or (iii) a *relationship,* when two or more relations reference it and the combination of the references are a highly informative attribute set. The present method assumes that references can be identified from the database schema.

### Object Analysis

The references identified during the entity analysis are simple relationships between individual relations. More complex relationships, called *abstract objects,* can be inferred by using database design rules to analyse each reference. An abstract object is a description of the part, or *role,* played in a relationship by the entities in the database. Each abstract object is a high level description of a database region from which a minable data set can be produced.

For example, in an employee database, an abstract object *employee-and-dependants,* which describes the relationship between employees and their dependants, might consist of the classes *employee, employee with dependants, employee without dependants, a set of dependants* and a *dependant.* In this example the concept being represented is that of an *employee* who may or may not have *dependants.* During the object analysis each entity is treated as a class in the object model.

A single reference can consist of one or more classes, each playing one or more roles. There are three pairs of roles (Smith & Smith, 1977; Brodie, 1981):

- **Aggregate/Component** A class which relates two or more classes because of the relationship between them is an aggregate, for example, an aggregate class *car* relates the component classes *door, wheel,* and *engine.*
- **Superclass/Subclass** A class which describes the common properties of one or more classes is a superclass, for example, *vehicle* is a superclass of subclasses *car, bus* and *train.*
- **Set/Member** A class which groups a set of classes together is a set, for example, *car wheels* is a set of member class *wheel.*

Currently, object analysis uses database design rules derived from common methods of mapping an entity-relationship model to a set of relations (Elmasri & Navathe, 1989). For example, subclasses in the relational model are often represented as relations with common primary key attributes. Each rule consists of evidence describing an observation about the content of the database and a conclusion which can be inferred given the evidence. The following rule classifies two entities as subclasses of an

*inferred* superclass ($\underline{e_j.a}$ is the set of values in attribute $a$ of $e_j$):

**Evidence** A reference exists between two informative attributes $e_1.a$ and $e_2.a$, and $\underline{e_1.a} \neq \underline{e_2.a}$.

**Conclusion** The entities $e_1$ and $e_2$ both play the role of disjoint subclasses of an inferred superclass $e_3$ which contains the common attributes of $e_1$ and $e_2$.

**Example** If data about skilled and unskilled workers is held in two relations, *skilled-worker* and *unskilled-worker,* and both relations are identified by the attribute *enumber* then a new class *employee* can be inferred whose instances are both a unskilled-worker and a skilled-worker.

In the relational model aggregates are represented by a foreign key in one relation referencing the primary key of another relation. Using this knowledge the following rule classifies an entity as an aggregate:

**Evidence** A reference exists between an uninformative attribute $e_1.a$ and an informative attribute $e_2.a$.

**Conclusion** The entity $e_1$ is an aggregate with component $e_2$.

**Example** If an entity *employee* contains an attribute *car-reg* which is also an identifier for an entity *car* then an aggregate *employee*, with component *car*, can be inferred.

Analyzing all the references in a database will produce many different abstract objects. Therefore, each abstraction must be reviewed by the data miner and domain expert and a suitable set of abstractions selected for data mining. Our approach hides the complexities of data structuring by describing each relationship using higher level constructs.

## Query Generation

Each role defined in an abstract object is mapped to one or more database queries. For example, the abstract object *employee,* a superclass consisting of two subclasses, *unskilled-worker* and *skilled-worker*, maps to a data set composed of those attributes which are common to both *unskilled-worker* and *skilled-worker* and which, therefore, are properties of the *employee* concept. Discoveries generated from the *employee* data set are true for both *unskilled-workers* and *skilled-workers.*

Inferred classes, which consist of structural attributes and no descriptive attributes, are used to label instances of the classes from which they were inferred. For example, assume that two subclasses of *car, employee-car* and *company-car,* are inferred from a reference between *employee* and *car.* The instances of *employee-car* and *company-car* are contained in *car.* Therefore, a new attribute, *type*, can be added to *car* which classifies each instance as either 'employee owned' or 'company owned'. This additional information can be used during the data mining search.

Data sets can be generated for each pair of roles:

- **Aggregate/Component** An aggregate data set is generated by joining the relations making up its components.
- **Superclass/Subclass** A superclass data set is generated from the union and projection of the shared attributes of its subclass relations.
- **Set/Member** A set is more complex as it describes the properties of a group of instances. Two methods can be used to analyse such data. First, summary data may be calculated for each instance of the set. Second, each set instance may be generated as a distinct data set.

## Example

Consider the following relation schemas:

**employee** (fname, minit, lname, ssn, address, sex, salary, superssn, dno)
**dependant** (ssn, dependant-name, sex, relationship)
**works-on** (ssn, pno, hours)
**project** (pname, pno, plocation, dno)
**department** (dname, dno, mgrssn).

A dependency analysis of the *employee* relation in produces three potential keys *fname, lname* and *ssn.* Of these keys only one takes part in a reference to other relations in the database, *ssn.* It is assumed to be the primary key. A dependency analysis of the remaining relations identifies the attributes *(ssn, pno)* and *pno* as keys in the relations *works_on* and *project,* respectively. Three entities, *employee, works_on* and *project,* are inferred from these findings. An abstract object consisting of an aggregate class *works_on* and components *employee* and *project* is inferred using the rule:

**Evidence** A reference exists between an informative attribute $e_1.a$ and attribute $e_2.a_1$, and a reference exists between an informative attribute $e_3.a$ and attribute $e_2.a_2$. The attribute set $(e_2.a_1, e_2.a_2)$ is informative.

**Conclusion** Entity $e_2$ is an aggregate class with component classes $e_1$ and $e_3$.

This object produces the relational schema:

**emp-proj** (fname, minit, lname, address, sex, salary, superssn, dno, pname, pno, plocation, dno)

The corresponding SQL query is:

```
SELECT fname, minit, lname, ssn,
address, sex, salary, superssn, dno,
hours, pname, pno, plocation, dno
FROM employee, works_on, project
```

```
WHERE employee.ssn = works_on.ssn AND
works_on.pno = project.pno
```

The result of running this query is a data set consisting of examples of the relationship between employees and projects. The data set can be analyzed using traditional data mining tools.

## Related Work

There has been little published work on reverse engineering relational databases as part of the knowledge discovery process. (Goldberg & Senator, 1995) introduce two new database operations, consolidation and link formation, which re-structure the database for knowledge discovery. Several data mining tools have been developed to analyse complex data structures. In the KATE system, (Manago & Kodratoff, 1991) apply an ID3-style algorithm to analyse frame-based data structures. This algorithm assumes the existence of a frame-based representation of the data. The abstract objects inferred in our approach may be used as the input to the KATE system.

(Ribeiro et al, 1995) propose a method of analyzing individual relations and combining the results using knowledge of the primary and foreign keys. Other algorithms make use of concept hierarchies which impose structure on otherwise simple data sets, for example: attribute oriented induction (Cai et al, 1991). (Ketterlin et al, 1995) have extended the COBWEB algorithm to discover useful clusters in structured databases. Their algorithm also analyses entity instances, but does not induce structure; instead it relies on a pre-defined entity-relationship model of the data. Ketterlin argues that databases are designed using entity-relationship models and that the analysis of these systems should be performed at this level of abstraction.

## Conclusion

There is a need to apply data mining to real databases which are characterized by complex data structures. We have presented a method of generating minable data sets from real databases. Our approach differs from other approaches by using database reverse engineering techniques to infer models of the data contained in the database. Database design heuristics have been used to analyse relationships in the data. Abstract descriptions of the data, called *abstract objects*, are inferred from these relationships and these are used to produce data sets which are minable by conventional data mining tools. We are currently studying the application of this technique in an operational environment.

Data mining real databases remains a complex problem and our approach raises some issues which we hope to investigate in the future. These include selecting the most suitable abstract objects using criteria specified by the data miner and using the abstract object descriptions as input to more complex data mining algorithms.

## Acknowledgment

This work was funded by BT Laboratories.

## References

Batini, C., Ceri, S., Navathe, S. B., 1992. *Conceptual Database Design.* Benjamin/Cummings.

Boulanger, D., March, S. T., 1989. An Approach to Analyzing the Information Content of Existing Databases. *Database* (Summer issue), 1-8.

Brodie, M. L., 1981. Association: A Database Abstraction for Semantic Modeling. In *Entity-Relationship Approach to Information Modeling and Analysis*, Chen, P. P. (ed.), North-Holland, 577-602.

Cai, Y., Cercone, N., Han, J., 1991. Attribute-Oriented Induction In Relational Databases. In *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley (eds.), AAAI Press, 213-228.

Elmasri, R., Navathe, S. B., 1989. *Fundamentals of Database Systems*, Benjamin/Cummings.

Goldberg, H. G., Senator, T. E., 1995. Restructuring Databases for Knowledge Discovery by Consolidation and Link Formation. In *KDD-95: Proc. of the 1st Int'l Conf. on Knowledge Discovery and Data Mining*, 136-141. AAAI Press.

Manago, M, Kodratoff, Y., 1991. Induction of Decision Trees From Complex Structured Data. In *Knowledge Discovery in Databases*, G. Piatetsky-Shapiro and W. J. Frawley (eds.), AAAI Press, 289-308.

McKearney, S., 1992. Inferring Object Schemas from a Database Instance. DPhil Thesis, University of Ulster.

McKearney, S., Bell, D., Hickey, R. J., 1991. Inferring Abstract Objects in a Database. In *CIKM-92: Proc. of the 1st Int'l Conf. on Information and Knowledge Management.*

Ribeiro, J. S., Kaufmann, K. A., Kerschberg, L., 1995. Knowledge Discovery from Multiple Databases. In *KDD-95: Proc. of the 1st Int'l Conf. on Knowledge Discovery and Data Mining*, 240-245. AAAI Press.

Roberts, H., Totton, K., 1996. Data Mining in BT. In *Proc. of Data Mining 96*, Unicom Seminar, London.

Smith, J. M., Smith, D. C. P., 'Database Abstractions: Aggregation and Generalization', in *Readings in AI*, Mylopoulos, J., Brodie, M. L. (eds.), Morgan-Kaufmann, 1989, pp138-145.

Springsteel, F., Kou, C.,1992. Reverse Data Engineering Technology for Visual Database Design. *Information and Technology*, 34 (2), 97-105.