J. Software Engineering & Applications, 2010, 3, 629-643
doi:10.4236/jsea.2010.37073 Published Online July 2010 (http://www.SciRP.org/journal/jsea)

Using an Ontology to Help Reason about the Information Content of Data

Shuang Zhu[1], Junkang Feng[2]

[1,2]Database Research Group, School of Computing, University of the West of Scotland, Paisley, UK; [2]Business College, Beijing Union University, Beijing, China.
Email: {shuang.zhu, junkang.Feng}@uws.ac.uk

Received March 25th, 2010; revised April 23rd, 2010; accepted April 25th, 2010.

ABSTRACT

We explore how an ontology may be used with a database to support reasoning about the "information content" of data whereby to reveal hidden information that would otherwise not derivable by using conventional database query languages. Our basic ideas rest with "ontology" and the notions of "information content". A public ontology, if available, would be the best choice for reliable domain knowledge. To enable an ontology to work with a database would involve, among others, certain mechanism thereby the two systems can form a coherent whole. This is achieved by means of the notion of "information content inclusion relation", IIR for short. We present what an IIR is, and how IIR can be identified from both an ontology and a database, and then reasoning about them.

***Keywords*:** *Ontology, Information Content of Data*

1. Introduction

Data mining techniques and tools are developed for finding otherwise hidden knowledge from data, and little seems to have been done on bringing "standard" domain knowledge into such a process, which we envisage would be helpful.

Ontologies as domain knowledge have been used in many fields. We want to explore how an ontology may help find hidden information from data. In this paper, the focus is on how to link an ontology with a relation database in order to reason about informational relationships between data constructs in the database and those between domain objects captured by an ontology. This may represent an innovative approach to knowledge discovery in a database.

Ontology [1] as a term used in computer science was started in the 1990's. Compared with the development of relational databases, it is a new scientific field. Ontology offers an opportunity to give an open and standardized description of database semantics with which we can substantially improve the quality and utilization of data. That is,

Ontology + Database = (Standards + Explicit Semantics) + Database,

which leads to improved data utilization and data quality [2].

Futhermore, *semantic web* [3] is a popular topic. Through semantic web we attempt to provide users with far better machine assistance than it is available now for their queries. Semantically annotated web pages with ontologies may assist reachers to achieve this purpose [4].

Through our work, we obtain an ontology from the DAML library, which represents some additional common knowledge, and link it with an existing database. In terms of linking an ontology and a database though, in the literature, we find a few different methods in using an ontology to assist a query process.

It appears that one way to achieve this is that an ontology is invoked at the very beginning of a query process [5] as shown in **Figure 1**. That is, it is through re-writing a query in order to get more information. A

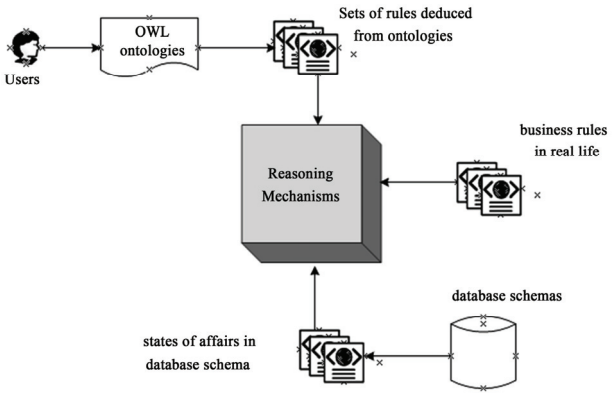


Figure 1. Invoking an ontology in the query processing

user query is translated into a set of queries with the help of the ontology, which better fits the structure of the data source. After query optimization strategies having been applied on them, the resultant transformed queries are equivalent to the submitted ones. Although seemingly a promising approach, it is not concerned explicitly with the information content of data, in which we are particularly interested and wish to explore and make use of.

Another approach that we have investigated is where an ontology is invoked in formulating a query process by Munir *et al*. [6]. In their approach, firstly, an ontology is generated based upon domain metadata including relationships between data in a relational database. Then such an ontology is enriched with domain knowledge. Secondly, ontology statements are translated into expressions in the OWL-DL language. Thirdly, the expressions are transformed into relational query statements. Finally, map the domain ontology to a relational database (as shown in **Figure 2**). Munir *et al.* [6] said little about the mapping between the created *ad hoc* ontology and the "standard" domain ontology if any, which we suspect is done intuitively. This is however one of the topics in which we are particularly interested.

We give an outline of our approach in Section 2. The key notion is informational relationship and its formalisation IIR [7]. We describe in details how IIR may be derived from a relational database and from an ontology in Section 3, which make use of inherent and *ad hoc* constraints between data constructs in a database and between concepts in an ontology. We present a full account on how our ideas are tested by using some implementation in Oracle in Section 4. Finally we give concluding remarks in Section 5.

2. Outline of our Approach

Our approach is to invoke an ontology when we work on a database. Namely, when a user submits a query, we do not change the query, but rather we involve the ontology in the reasoning process *per se* that is required for an swering the query (shown in **Figure 3**). Furthermore and most importantly for us, the reasoning is carried out on the basis of the notion of "information content" of data. This notion is the work of Xu, Feng and Crowe in 2008 [8], which extends substantially Dretske's [9] definition of "information content" of a signal. In this paper, they introduce another notion called IIR, as a formulation of the notion of "information content" of data.

Xu *et al*. [8] define IIR as follows: "Let X and Y be an event respectively, there exists an IIR, from X to Y, if every possible particular of Y is in the information content of at least one particular of X". Furthermore, they define that "Let X be a event, the *information content* of X, denoted I(X), is the set of events with each of which X has an information content inclusion relation". Moreover, they present a sound and complete set of inference rules (IIR rules) for reasoning about information content of data (states of affairs, or events in general). The six inference rules are cited below.

1) **Sum**

If $Y = X_1 \cup X_2 \cdots \cup X_n$ then $I(X_i) \ni Y$ for $i = 1, \ldots, n$

This rule says if it is the disjunction of a number of events, then an event X is in the information content of any of the latter. A trivial case is where X and Y above are not distinct.

2) **Product**

If $X = X_1 \cap X_2 \cdots \cap X_n, Y = X_i$ for $i = 1, \ldots, n$ then $I(X) \ni Y$

This rule says that if an event X is the conjunction of a number of events, then any of the latter is in the information content of the former. A trivial case is where X and Y above are not distinct.

3) **Transitivity**

If $I(X) \ni Y, I(Y) \ni Z$ then $I(X) \ni Z$

This rule says that if the information content of an event X includes another event Y, and the information

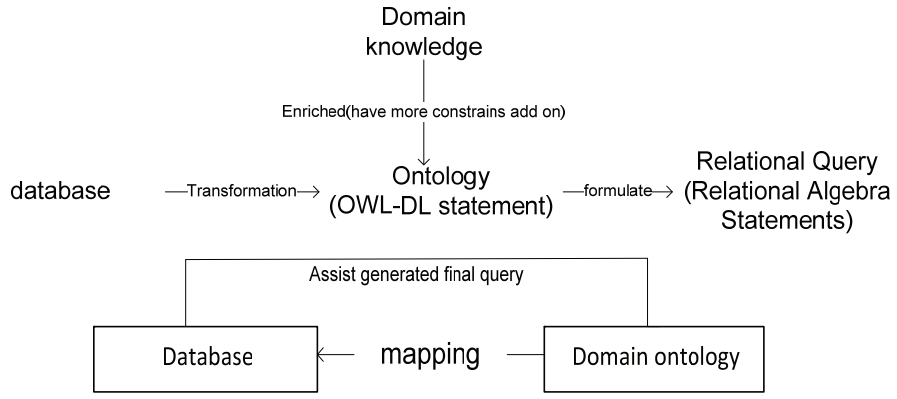


Figure 2. Ontology assisting the formulation of a query

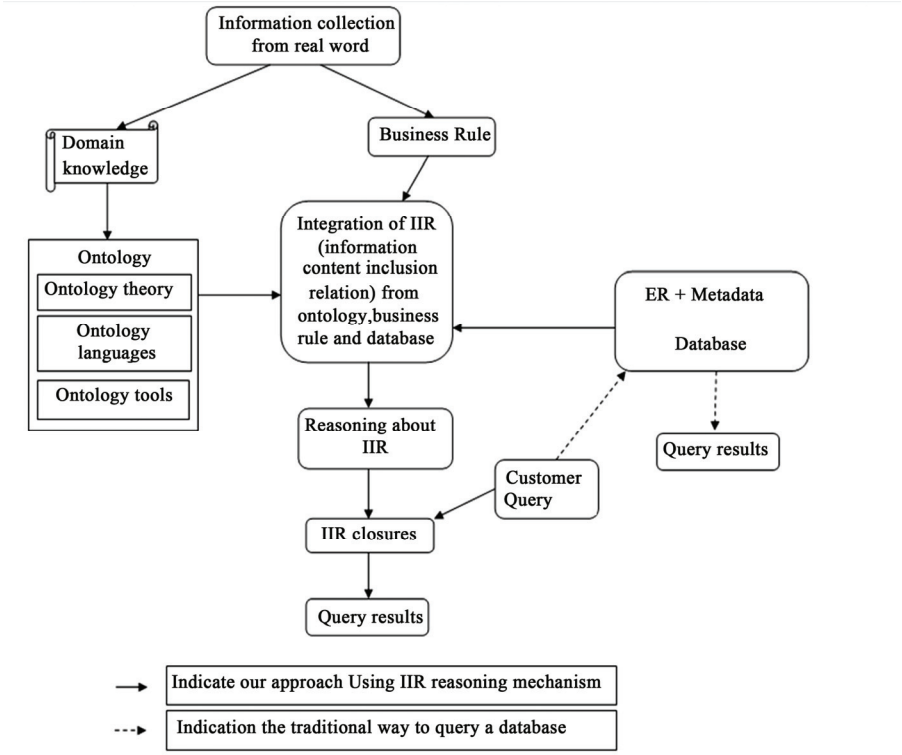


Figure 3. Ontology enhances reasoning about the information content in a database

content of *Y* includes yet another event *Z*, then the information content of *X* includes *Z*.

4) **Union**

If $I(X) \ni Y, I(X) \ni Z$, then $I(X) \ni Y \cup Z$

This rule says that if the information content of an event *X* includes another two events *Y* and *Z* respectively, then the information content of *X* includes event $Y \cap Z$ that is the product of *Y* and *Z*. And it is in this sense that *Y* and *Z* are in a "union".

5) **Augmentation**

If $W = W_1 \cap W_2 \cdots \cap W_n$, *Z* is the product of a subset of $\{W_1, W_2, \cdots, W_n\}$ then $I(W \cap X) \ni Z \cap Y$

This rule says that if $W_1 \cap W_2 \cdots \cap W_n$, event *Z* is the product of a subset of $\{W_1, W_2, \cdots, W_n\}$, and the information content of event *X* includes event *Y*, then the information content of the event $W \cap X$ formed by the product of *W* and *X* includes the event $Z \cap Y$ formed by the product of *Z* and *Y*.

6) **Decomposition**

If $I(X) \ni Y \cap Z$ then $I(X) \ni Y, I(X) \ni Z$

This rule says that if the information content of event *X* includes event $Y \cap Z$ that is the product of event *Y* and event *Z*, then *Y* and *Z*, as separate events, are in the information content of *X*, respectively.

In this paper, we exploit the ideas above. That is, in a way, we translate both the ontology and the database into IIR and then reason about them as a whole. Put another way, as what matters is information and IIR captures and formulates it, so we look at both an ontology and a database from the same perspective of IIR, and this enables the two different things to work together. The overview of our approach is illustrated in **Figure 3**.

On the very top of **Figure 3**, there is a block called "information collection from the real world". From this information, knowledge about a domain of interest including explicit business rules is arrived at. Domain knowledge is then formulated as an ontology by using software tools and languages.

Two different routes are there to deal with user queries. If it is in a conventional query language then a query is handled in a normal way. The dotted line indicates this route. If that does not work, we would invoke the other route, *i.e.*, to invoke ontology and reasoning about IIR. The second solution is the primary goal of this project, which is indicated by the solid line arrow in **Figure 3** of "Customer query"→"IIR closures"→"Query results".

The only difference between these two solutions lies in the middle part of the procedure, on which we concentrate. Within the "Integration of IIR" section, there are three different resources required to derive the "IIR", indicated by three arrows from "ontology", "business rule" and "database", which are the origins of initial IIR. Then there is a reasoning mechanism implemented in PL/SQL of Oracle. The result of the reasoning is *IIR*

closures. Given an event A, the *IIR closure* of A, denoted as A^+, is the set of all events that are in the information content of A, that is, if IIR(A, B), then $B \in A^+$. *IIR closures* are the basis of answering queries in our approach. Our work thus far shows that it is the *additional relationships between data constructs* especially "entities" that are revealed and made available through using an ontology that give us more and enlarged *IIR closures* than those that would otherwise be based on the database alone. This is how our approach makes a difference.

One of the main tasks is to derive IIR from the ontology, the database and business rule, and then integrate them as a whole. For instance, suppose that we have IIR(A, B) (meaning the information content of A includes B) and IIR(C, D) from a relational database, and IIR(E, F) and IIR(G, H) from an ontology. If we also know that A and E are equivalent, then with Transitivity, we get IIR(E, B) and IIR(A, F). Consequently A^+ and E^+ are enlarged.

We use Oracle [10] to implement this approach. An ontology in OWL [11] can be translated into relational tables [12]. Such tables do not hold data values however, if the ontology is an unpopulated one. In such a case, the involvement of an ontology results in additional objects and additional relationships between objects that are represented by data in the original relational database. This way, a query that does not have exact match with data in the database may be answered. An ontology may add an additional hierarchical structure to data in the database. Furthermore, as said earlier, we use ontologies in a special type of reasoning, *i.e.*, reasoning about the *information content* of data through a kind of special relationship between data items and between data items and real world objects, namely *informational relationships*, which is captured and formulated as IIR between *events* (in terms of probability theory). Thus, how to identify IIR from an ontology becomes a key factor in our approach.

3. Deriving IIR

An IIR is a relationship between two *states of affairs* (*i.e.*, *events*) such that one's existence results in the certainty that the other exists, and without the former, the latter is not certain. Following Dretske 81 [9], we say that the latter is in the "information content" of the former.

It would appear that to express IIR(X, Y) must be based on and revolved around two elements. One is two individual values (two individual parts or two sets of groups) captured as X and Y, and the other is relationships between X and Y.

We use part of a "university" database and part of ontology "Academic" to present how IIR can be derived from a database and an ontology. Then the IIR are reasoned about by applying aforementioned Inference Rules. The reasoning is implemented by a program.

3.1 Deriving IIR from an Ontology

According to characteristics of ontologies, these are two different sources that may help the derivation of IIR. One is concerned with relationships between "Classes" in an ontology. The other is "ObjectProperty".

3.1.1 IIR Derived from Classes

Generally, there are two different types of relationships between classes from which IIR exist. One is "subClassOf", and the other "equivalentClass". The syntax for these two in an OWL ontology is as follows:

- A relationship between "Class" and "subClassOf",

```
<owl:Class rdf:ID="Lecturer">
   <rdfs:subClassOf rdf:resource="# Faculty">
</owl:Class>
```

- A relationship between "Class" and "equivalentClass",

```
<owl:Class rdf:ID="Teachers">
<owl:equivalentClass rdf:resource="#Faculty"/>
</owl:Class>
```

The IIR could be derived from these two relationships thusly:

- IIR(Class, subClassOf),
- IIR(Class, equivalentClass) and IIR (equivalentClass, Class).

As shown above, we have a relationship "Lecturer is a subclass of Faculty, and Teachers is an equivalent class to Faculty". Hence we have IIR(Lecturer, Faculty), IIR(Teacher, Faculty), IIR(Faculty, Teacher), and IIR(Lecturer, Teacher).

3.1.2 IIR Derived from ObjectProperty

There are four different types of ObjectProperty relationships, which capture relationships between classes in an OWL ontology. These are: "ObjectProperty", "subPropertyOf", "equivalentProperty" and "inverseOf".

As aforementioned, to create IIR needs two classes (X and Y) from the ontology. As ObjectProperty represents a relation for connecting two classes of "domain" and "range" in an OWL ontology, an ObjectProperty already contains a set of classes, which can be expressed as "ObjectProperty=('domain', 'range')". Accordingly, we obtain IIR(domain, range). That is, the IIR that can be derived from these four types of ObjectProperty is all of the form:

- IIR(domain, range)

Note that IIR must be of a many-to-one relationship (including one-to-one). How to handle many-to-many is to be addressed shortly.

The relevant syntax of OWL is as follows:

- A relationship between "ObjectProperty",

```
<owl:ObjectProperty rdf:ID="research_by ">
<rdfs:domain rdf:resource="Professors"/>
<rdfs:range rdf:resource="Projects"/>
</owl:ObjectProperty>
```


Then we have "IIR(domain, rang)", for example, IIR(Professors, Projects).

- A relationship between "ObjectProperty" and "subPropertyOf",

```
<owl:ObjectProperty rdf:ID="research_in">
<rdfs:domain rdf:resource="Postgraduates"/>
<rdfs:range rdf:resource="Projects"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="study_in">
<rdfs:subPropertyOf rdf:resource="research_in"/>
<rdfs:domain rdf:resource="Postgraduates"/>
<rdfs:range rdf:resource="Projects"/>
</owl:ObjectProperty>
```

Then we have "IIR(domain, range)", for example, IIR(Postgraduates, Projects).

- A relationship between "ObjectProperty" and "equivalentProperty",

```
<owl:ObjectProperty rdf:ID="attend_course">
<rdfs:domain rdf:resource="Student"/>
<rdfs:range rdf:resource="Course"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="join_course ">
        <owl:equivalentProperty
rdf:resource="attend_course"/>
<rdfs:domain rdf:resource="Students"/>
<rdfs:range rdf:resource="Course"/>
</owl:ObjectProperty>
```

Then we have "IIR(domain, range)"', for example, IIR(Students, Course).

- A relationship between "ObjectProperty" and "inverseOf",

```
<owl:ObjectProperty rdf:ID="teache_of">
<rdfs:domain rdf:resource="Faculty"/>
<rdfs:range rdf:resource="Course"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="instruct_by">
<owl:inverseOf rdf:resource="teaches_of"/>
<rdfs:domain rdf:resource="Course"/>
<rdfs:range rdf:resource="Faculty"/>
</owl:ObjectProperty>
```

Then we have "IIR(domain, rang)", for example, IIR(Course, Faculty).

Moreover, there are relationships between classes and "Not Null" FunctionalProperty, the syntax of which is as follows:

- A relationship between "Class" and "FunctionalProperty",

```
<owl:Class rdf:ID="Course">
   <owl:DatatypeProperty rdf:ID="courseNo">
   <rdfs:type
rdf:resource="&owl;FunctionalProperty"/>
<!-- NOT NULL -->
<rdfs:domain rdf:resource="Course"/>
        <rdfs:range rdf:resource="&xsd;short"/>
   </owl:DatatypeProperty>
```

Then we have "IIR(Class, DatatypeProperty)", for example, IIR(Course, courseNo).

Furthermore, to handle a many-to-many relationship, we transform it into two many-to-one relationships. Consider firstly this scenario: "one course is taken by more than one students and one student takes more than one course". This is a many-to-many relationship. We decompose such a relationship into two many-to-one by creating a new class and then they are treated in the same way as the second method for the ObjectProperty transformation. We create an intermediate table and use the ObjectProperty name as the new class name (as well as the table name which will be the transformation of this class). In details, the relationship "StudentTakeCourse" between the class "student" and the class "course" is many-to-many. We create a new class CourseLearning, which contains two ObjectProperty relationships as shown below:

Class (Student)

DatatypeProperty (studentNo domain (Student) range (xsd: short) Functional)

DatatypeProperty (studentName domain (Student) range (xsd: string))

DatatypeProperty (major domain (Student) range (xsd: string))

DatatypeProperty (enrollmentDate domain (Student) range (xsd: date))

Class (Course)

DatatypeProperty (courseNo domain Course) range (xsd: short) Functional)

DatatypeProperty (courseName domain (Course) range (xsd: string))

DatatypeProperty (creditHour domain (Course) range (xsd: integer))

Class (CourseLearning)

ObjectProperty (takenBy domain (CourseLearning) range (Student))

ObjectProperty (inv - takenBy domain (Student) range (CourseLearning) inverseOf (takenBy))

ObjectProperty (takesCourse domain (CourseLearning) range (Course))

ObjectProperty (inv - takesCourse domain (Course) range (CourseLearning) inverseOf (takesCourse))

Accordingly the IIR obtained in this process are IIR(CourseLearning, Student) and IIR(CourseLearning, Course) (**Figure 4)**.

The paragraphs that follow illustrate details at the "instances" (data values) level of the above example.

The original class CourseLearning is divided into two parts takenBy and takesCourse (as ObjectProperty), either of which only shows a one-way relationship. These combined however form the relationship between students and courses. **Table 1** shows some instances.

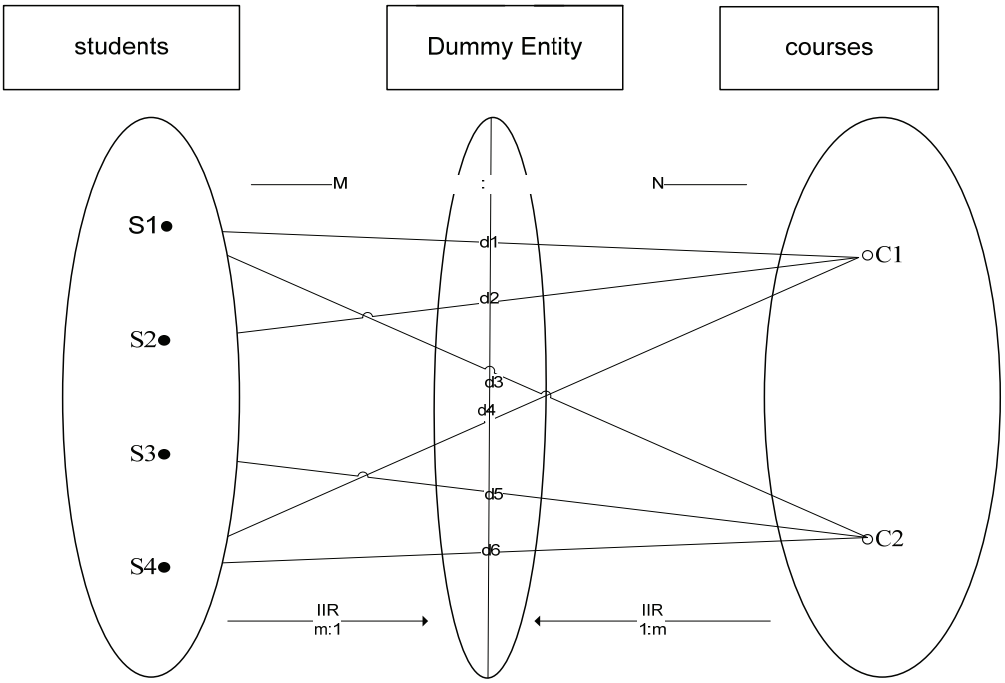


Figure 4. The overview for IIR relationship of CourseLearning

Table 1. Table transformation from CourseLearning

CourseLearning	Student	Course
d1	S1	C1
d2	S2	C1
d3	S1	C2
d4	S4	C1
d5	S3	C2
d6	S4	C2

3.2 Deriving IIR from a Database

In a database, initial IIR (*i.e.*, IIR that is not implied by others) come from three sources: relationships between tables, relationships between attributes and relationships between individual data values. Two different ways can be used to derive such IIR.

- A relationship between a "subclass" and a "super class"

Figure 5 shows part of a "university" database schema. An IIR exists between two tables if one is a super class of the other, for example, IIR(postgraduate, student).

Note that, IIR is a relationship between *event*s as we said earlier. For tables, we define events as follows: if we randomly chose a tuple from a database, that the tuple happens to be in a particular table is an event. Thus the above IIR(postgraduate, student) means that the existence of a tuple in table Postgraduate makes certain that a tuple that corresponds to the former exists in table Student.

- A many-to-one relationship between two tables

Similar to deriving IIR from an "ObjectProperty" with an ontology, we obtain IIR(table 1, table 2) if they have a many-to-one relationship for example, IIR(undergraduate, course).

- Constraints of the relational data model and business rules on data

A third source of IIR is *constraints* of a relational database and business rules on data, for example, IIR(table 1, PK) and 'IIR(PK, attribute1). For **Figure 5**, we have IIR(courses, courseID) and IIR(courseID, courseName). The former means that the existence of a tuple of table Courses makes certain that a corresponding course ID (a value) exists. The latter means that the existence of a course ID makes certain that a corresponding course name exists.

Another type of IIR is IIR(FK1, PK1), for example:

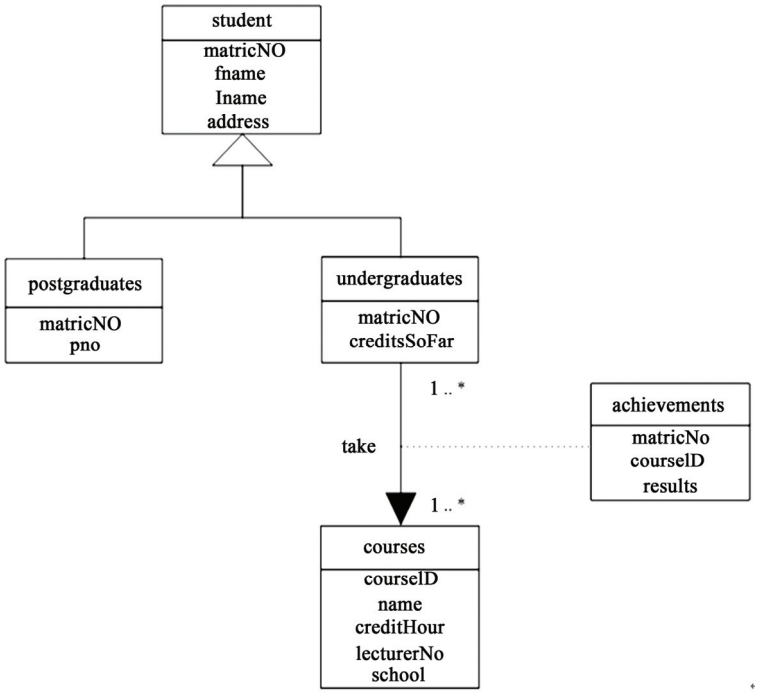


Figure 5. 6 partial EER diagram of "University" relational database

IIR(courseIDoftableStudents, courseIDoftableCourses), which means that the existence of a course ID in table Students makes certain that a corresponding course ID exists in table Courses.

A database is normally populated with data values. Inaddition to the above IIR on the "table" level and "attribute" level, there could be IIR identifiable at the "data value" level.

The information that each individual "data value" holds in a relational database comes from the semantics of the attributes to which the data value belongs. This is due to the capacity of a concept's "giving meaning to its instances" [9]. An attribute may be seen as representing a concept. For example, "student name" is seen as a concept. Relationships between entities in a database can be seen as "complex concepts" [9] and therefore also give meaning to data values that are instances of the relationships. That is, data in the relational database already hold relationships upon which there are constraints imposed.

We now use a simple example to summarise how IIR may be derived on the three levels. Suppose three tables shown in **Figure 6** in the "University" database.

Table level

According to **Figure 6**, table administration staff is a subclass of table staff, which gives the following IIR between these two tables:

IIR(administration_staff, staff)

As previously mentioned, the meaning of IIR indicates that first arguments existence results in the certainty that the other exists, and without the former, the latter is not certain. Therefore, the meaning of the two part relationship in this particular IIR, may be explained as: "if there is a member of administration staff then a corresponding member of staff must exist, otherwise the latter is not certain". In this particular case, the IIR is true because any member of administration staff is a member of staff.

Attribute level

If an attribute "A" is in a table which includes attributes "A", "B" and "C". Then, any combination of "A", "B" and "C" that includes "A" would have "A" in its information content. For example, IIR(A∩B, A), which means that if an instance, say (*a*, *b*), of "A∩B" exists, then there must be an corresponding instance of "A" existing - in this case, it is *a*. In general, this type of IIR is IIR("a set of attributes", "a subset of the attributes").

Using the values in **Figure 6**, an example is shown below. The attributes in table administration staff include sno, position, and deptNo. So the IIR are:

IIR(sno∩position∩deptNo, sno)
IIR(sno∩position∩deptNo, sno∩position)
IIR(sno∩position∩deptNo, sno∩ deptNo)
IIR(sno∩position∩deptNo, position)
IIR(sno∩position∩deptNo, position∩deptNo)
IIR(sno∩position∩deptNo, deptNo)

These IIR are derived on the attributes level, which may be seen as based on the aforementioned "Product" Rule, *i.e.*, if an event *X* is the conjunction of a number of events, then any of the latter is in the information content of the former.

Data value level

DB-staff

sno	fname	lname	sex	Address	tel	office
s02923	John	Key	M	6 Lawrence St, Glasgow	2384	E110
s02933	Julie	Lee	F	8 George St, Glasgow	2234	G203
s04885	Ann	White	F	18 Taylor St, Glasgow	5112	G133
s04995	Susan	Brand	F	28 High St, Paisley	3001	G229
s06465	Mary	Tregear	F	7 George St, Paisley	7754	F232
s06883	David	Ford	M	64 Well St, Paisley	8772	F231

DB-administration staff

sno	position	deptNo
s04885	secretary	d01
s04995	accountant	d03

DB-departments

deptID	departmentName
d01	administration
d03	finance

Figure 6. Three tables within the "University" relational database

Unlike the unpopulated ontology in use for this project, data values are a very important part in relational databases and it also the largest constituent of a relational database. Before explaining how to derive IIR on the data value level, let us re-cap the meaning of the terms we have been using, *i.e*., "random variable" "event" and "particular of an event".

A *random variable* is an entity used mainly to describe chance and probability in a mathematical way. An *event* is a set of outcomes (a subset of the sample space) to which a probability is assigned. Typically, when the sample space is finite, any subset of the sample space is an event (*i.e.,* all elements of the power set of the sample space are defined as events) (A WorldViewer.com, 2009). Moreover, a specific event at a particular time and in a particular space is called a *particular* of an event. For example, consider the following situation. For an electric circuit, two random variables can be identified: one is "the condition of the lamp", and the other "the condition of the switch". There are two states about the lamp: "lit" and "unlit", and two for the switch: "closed" and "open". There are 2^2 events for either. Moreover, "unlit" at 10:30 am and "lit" at 10:30 pm, are two particulars.

Table 2. IIR between data values

Random variables	IIR
'sno', 'position'	iir('s04885', 'secretary')
	iir('s04995', 'accountant')
'sno', 'deptNo'	iir('s04885', 'd01')
	iir('s04995', 'd03')

Table 2 shows some random variables and associated for the university database given in **Figure 6**. In a rela-tional database, an attribute, e.g., sno, can be seen as a random variable, and then each possible data value in this column is an event. That is, randomly picking up a tuple in this column, then its value could be any one of all those that are allowed. An attribute is therefore a variable. The variable holding a particular value is an event.

3.3 Deriving IIR from Business Rules

Business rules are domain dependent, established by an individual organisation and they are *ad hoc* logical limitations on data. Business rules may be embedded in an ontology and also could be in a database. In order to derive IIR from these business rules, we treat an Object-Property in an OWL ontology as if it were a constraint in a database. Both could be represented as additional relationships. For example, in a university, there might be a rule: "Any newly recruited lecturer must hold a PhD". Then we have an IIR(newly recruited lecturer, PhD), which means that if someone is a newly recruited lecturer, then she/he must hold a PhD corresponding to him/her.

4. Testing our Idea

In this section, we show a case study that verifies our idea. We created an ontology entitled "Academic" and a relational database "University". The program runs in Oracle using PL/SQL. This case study elucidates the different results when reasoning is based on the database in question only, and on both the database and the ontology integrated through IIR.

There are 11 tables in the "University" database shown in **Figure 7**.

And as we mentioned in previous sections, in the OWL ontology, a "Class" is transformed into a "table". "subClassOf" is treated as a "Class". A "DatatypeProperty" is changed to an "attribute". An "ObjectProperty" is a relationship between classes, which are transformed into constraints upon these tables. So, we arrive at 10 tables shown in **Figure 8** from the "Academic" ontology.

Thus the schema of the "University" is substantially extended as shown in **Figure 9**, from which more IIR are derived.

Figure 10 shows the 10 tables in SQL Plus of Oracle.

4.1 Original IIR and Derived IIR

A few business rules are defined for this case study. They specify correspondences between the "Academic" OWL ontology and the "University" relational database. For example, there is a table in the "University" database called "staff". There is a class in the "Academic" OWL ontology named "Person", and "staff" is a subclass of "Person". Other 18 business rules are concerned with equivalent classes between ontology and the database at class level. There is one on the ObjectProperty.

The original IIR derived from the ontology and the 'database and business rules are shown in **Table 3**.

Applying the IIR inference rules listed earlier to the IIR identified, more IIR are derived.

4.2 Implementation and Results

As we previously mentioned, firstly, we created tables for the relational database (named 0db.sql), the ontology (named 0onto.sql) and IIR (named 0IIR.sql), used for storing both original IIR from the ontology and the database and 0IIR_DB.sql is used for storing the original IIR from the relational database alone.

Secondly data values are inserted into the database tables and all the original IIR are entered into the IIR tables. Then all single attributes and class names and attribute names from the ontology and the database are obtained and inserted into the attributes table with duplicate components removed.

Thirdly, three intermediate tables are created. The tables named fo1 and fo2 store the former and the latter part of original IIR respectively, and the table t1 stores intermediate results.

Fourthly, a procedure is invoked. For instance, when a user asks a question, relevant IIR closures will be looked at. They embody relevant information for the query. There is a string match function in this procedure.

In order to find out the difference that the ontology makes, we compare the two results. One was obtained by using both the "Academic" OWL ontology and the "University" database, and the other obtained using the database alone. They are shown in **Table 4**.

As **Table 4** shows, the first column "Attributes" indicate all attributes that are extracted from "Academic" OWL ontology and the "University" database. The column "Closures from both ONTO and DB" shows the IIR closures that we derive by running our prototype when the "Academic" OWL ontology is involved, The column "Closures from DB only" consists of the IIR closures that we derive by running our prototype when only the "University" database is involved.

We use the same five questions in the testing. As **Table 4** shows, the attributes that are included in the results are ticked. When the database alone is used, a query for "sno" gives 12 results and a query for "matricNo" gives 7 results. When both the ontology and the database are used, the same query for "sno" gets 14 results and the same query for "matricNo' gives 9 results. That is, two more attributes are found to be included in the respective IIR closures when the ontology is involved, which means that more information is made available due to the ontology.

5. Conclusions

We have described how an ontology may be linked with database in order to derive hidden information. A prototype in Oracle was developed to verify our ideas. We use the notion of IIR (Information content Inclusion relation) and inference rules for IIR.

We have found that if we do not invoke a relevant ontology, a query may be unanswerable. After invoking an ontology, more relationships between objects become available, and therefore more elements can connect to one another, and as a result, a query may become answerable, and as a result, more information can be derived from data in a database. To achieve this, a key is to be able to identify IIR from both a database and an ontology. We have presented a way of doing so.

More work need to be done in the future, for instance, to display correspondences between a query and the answers in a more accurate and specific way, *i.e.*, not just listing the answers. One issue that is not aesthetic is how to achieve semantic alignment between an ontology and a database, on which we are currently working.

sno	fname	lname	sex	Address	tel
s02923	John	Key	M	6 Lawrence St, Glasgow	2384
s02933	Julie	Lee	F	8 George St, Glasgow	2234
s04885	Ann	White	F	18 Taylor St, Glasgow	5112
s04995	Susan	Brand	F	28 High St, Paisley	3001
s06465	Mary	Tregear	F	7 George St, Paisley	7754
s06883	David	Ford	M	64 Well St, Paisley	8772

1. DB-staff

sno	title	school
s02923	lecturer	computing
s02933	professor	business

2. DB-faculty

sno	school	pno
s06465	business	p00203
s06883	engi-neering	p00334

3. DB-researcher

sno	position	deptNo
s04885	secretary	d01
s04995	accountant	d03

4. DB-administration staff

matricNo	pno
ts030283	p00334
tm051083	p00203

6. DB-postgraduates

matricNo	fname	lname	sex	Address
ts030283	Tony	Shaw	M	20 George St, Paisley
tm051083	Tina	Murphy	F	16 George St, Paisley
rn050385	Robert	Nielson	M	11 George St. Paisley
hf151186	Henry	Ford	M	7 Well St. Paisley
jw010483	John	White	M	5 Novar Dr, Glasgow
sb210682	Susan	Brand	F	2 Manor Rd, Glasgow
cp020381	Chris	Paul	M	6 Lawrence St, Glasgow

5. DB-student

matricNo	creditsSoFar	matricNo
rn050385	155	M050385

8. DB-projects

matricNo	creditsSoFar
rn050385	155
hf151186	65

7. DB-undergraduates

courseID	courseName	creditHour	lecturerNo	school
c0054	Oracle Development	24	s02923	computing
c0021	International Finance Planning	24	s06465	business
c0154	Advanced Oracle Development	24	s02923	computing
c0155	Networking Principles	16	s06883	computing
c0220	Software Development	24	s06883	computing

9. DB-courses

matricNo	courseID	results
rn050385	c0054	A
hf151186	c0021	C1
cp020381	c0154	B1
cp020381	c0155	C2
sb210682	c0220	B2

10. DB-achievements

deptID	depart-mentName
d01	administration
d03	finance

11. DB-departments

Figure 7. Tables in the "University" database

name	age	sex

1. onto-Person

specialty	educationDegree	back-ground

2. onto-Worker

school	title	background

3. onto-Faculty

department	Position	background

4. onto-Administration staff

school	background

5. onto-Assistants

supervisor

7. onto-Postgraduates

credits

8. onto-Undergraduates

studentNo (PK)	studentName	major	address	E-mail	sex

6. onto-Student

projectNo (PK)	projectName

9. onto-Projects

courseNo (PK)	courseName	creditHour

10. onto-Course

Figure 8. Table transformations from the "Academic" ontology

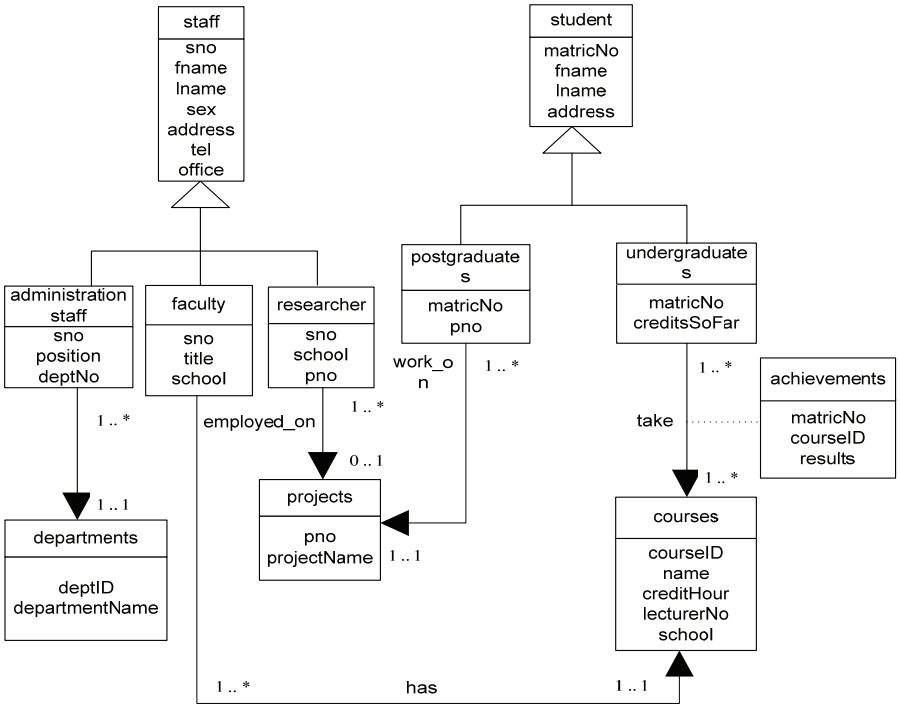
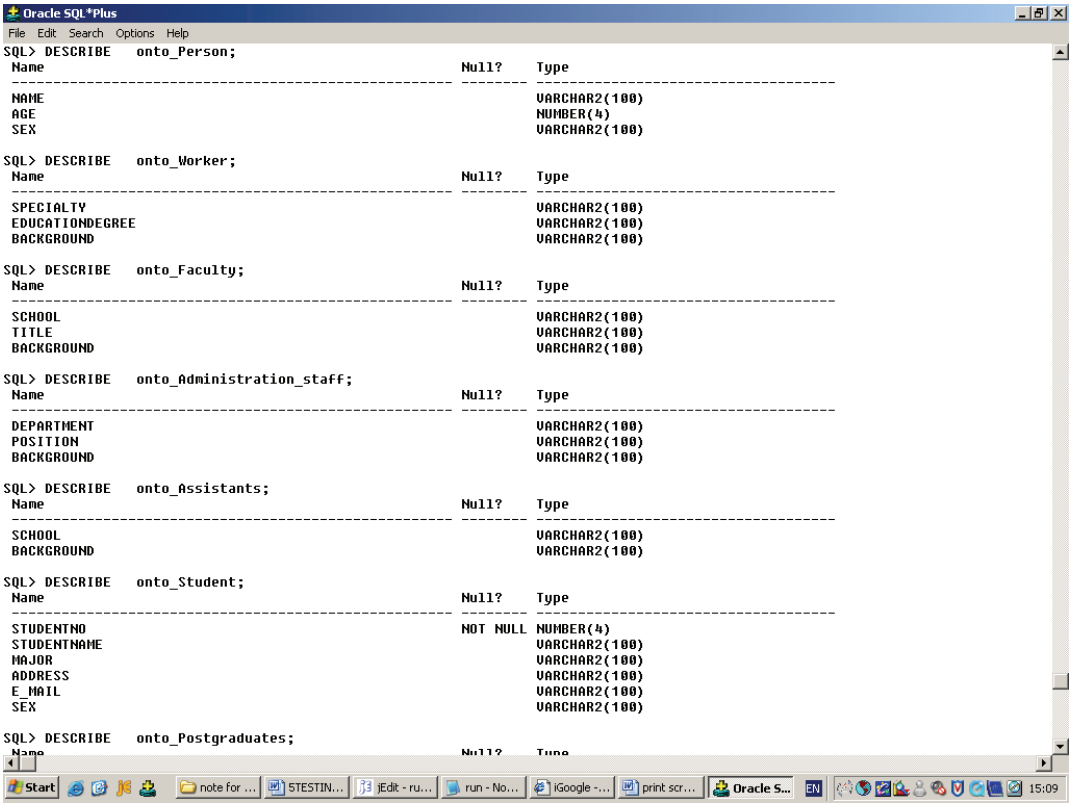


Figure 9. "University" database extended due to an ontology



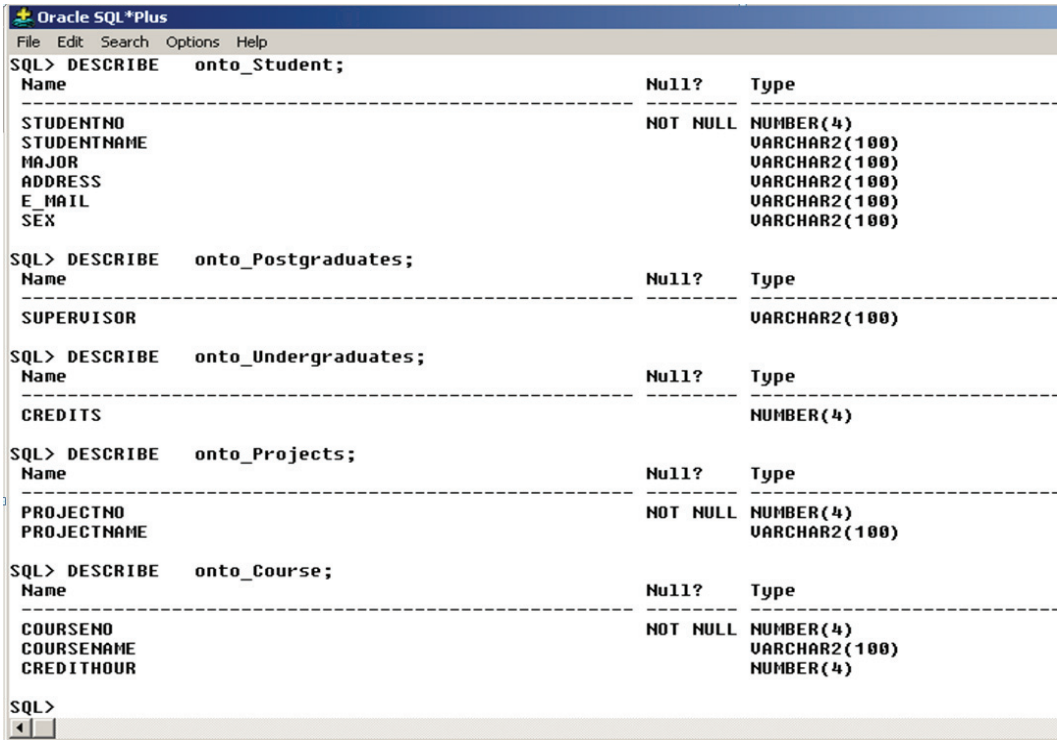


Figure 10. The "Academic" ontology represented in SQL Plus

Table 3. IIR derived from the "Academic" OWL ontology and the "University" relational database

IIR derived from the 'Academic' OWL ontology	IIR derived from the 'University' Relational Database	IIR derived from Business Rules (corresponding relations)
***class, subclass* (17) *and equivalent class* (1)**	***class, subclass* (5) *and equivalent class* (0)**	***class, subclass* (1) *and equivalent class, attributes* (20)**
1.IIR(Worker, Person)	1.IIR(faculty, staff)	1.IIR(staff, Person)
2.IIR(Faculty, Worker)	2.IIR(administration_staff, staff)	2.IIR(Worker, staff)
3.IIR(Professors, Faculty)	3.IIR(researcher, staff)	3.IIR(Faculty, faculty)
4.IIR(Lecturer, Faculty)	4.IIR(postgraduates, student)	4.IIR(Administration_staff, administration_staff)
5.IIR(Postdoc, Faculty)	5.IIR(undergraduates, student)	5.IIR(Projects, projects)
6.IIR(Administration_staff, Worker)	According to the 'University' relational database EER	6.IIR(Student, students)
7.IIR(Dean, Administration_staff)	diagram (**Figure 8**), these 5 IIR could be derived from it.	7.IIR(Postgraduates, postgraduates)
8.IIR(Chair, Administration_staff)		8.IIR(Undergraduates, undergraduates)
9.IIR(Clerical_staff, Administration_staff)		9.IIR(Course, courses)
10.IIR(System_staff, Administration_staff)		10.IIR(courseNo, courseID)
11.IIR(Director, Administration_staff)		11.IIR(projectNo, pno)
12.IIR(Assistants, Worker)		12.IIR(staff, Worker)
13.IIR(Reacher_assistants, Assistants)		13.IIR(faculty, Faculty)
14.IIR(Teaching_assistants, Assistants)		14.IIR(administration_staff, Administration_staff)
15.IIR(Student, Person)		15.IR(projects, Projects)
16.IIR(Postgraduates, Student)		16.IIR(students, Student)
17.IIR(Undergraduates, Student)		17.IIR(postgraduates, Postgraduates)
18.IIR(Teachers, Faculty)		18.IIR(undergraduates, Undergraduates)
		19.IIR(courses, Course)
		20.IIR(courseID, courseNo)
		21.IIR(pno, projectNo)
***ObjectProperty* (7) *and equivalent ObjectProperty* (2)**	***ObjectProperty* (5) *and equivalent ObjectProperty* (0)**	***ObjectProperty* (0) *and equivalent ObjectProperty* (3)**
1.---teache_of IIR(Faculty, Course)	1.---work_in IIR(administration_staff, departments	1.IIR(has, teache_of)
2.---attend_course IIR(Student, Course)	2.---has IIR(faculty, courses)	2.IIR(research_in, work_on)
3.---research_by IIR(Professors, Projects)	3.---employed_on IIR(researcher, projects)	3 IIR(study_in, work_on)
4.---instruct_by IIR(Course, Faculty)	4.---work_on IIR(postgraduates, projects)	
5.---research_in IIR(Postgraduates, Projects)	5.---take IIR(undergraduates, courses)	
6.---study_in IIR(Postgraduates, Projects)		
7.---join_course IIR(Student, Course)		
8. IIR(study_in, research_in)		
9.IIR(join_course, attend_course)		
***Constraints----NOT NULL* (6)**	***constraints----PK* (22)**	***constraints* (0)**
1.IIR(studentNo, 'studentNo, studentName,major,address,E-mail,sex')	1.IIR(sno, 'sno,fname,lname,sex,address,tel,office')	
2.IIR(courseNo, 'courseNo, courseName, creditHour')	2.IIR(sno, 'sno,title,school')	
3.IIR(projectNo, 'projectNo, projectName')	3.IIR(sno, 'sno,school,pno')	
4.IIR(Student, studentNo)	4.IIR(sno, 'sno,position,deptNo')	
5.IIR(Projects, projectNo)	5.IIR(matricNo, 'matricNo,fname,lname,sex,address')	
6.IIR(Course, courseNo)	6.IIR(matricNo, 'matricNo,pno')	
	7.IIR(matricNo, 'matricNo,creditsSoFar')	
	8.IIR(pno, 'pno,projectName')	
	9.IIR(courseID, 'courseID, courseName, creditHour, lecturerNo, school')	
	10.IIR('matricNo,courseID', 'matricNo,courseID,results')	
	11.IIR(deptID, 'deptID,departmentName')	
	12.IIR(staff, sno)	
	13.IIR(faculty, sno)	
	14.IIR(researcher, sno)	
	15.IIR(administration_staff, sno)	
	16.IIR(student, matricNo)	
	17.IIR(postgraduates, matricNo)	
	18.IIR(undergraduates, matricNo)	
	19.IIR(projects, pno)	
	20.IIR(courses, courseID)	
	21.IIR(achievements, 'matricNo,courseID')	
	22.IIR(departments, deptID)	

Table 4. IIR closures compared

Attributes	Closures from both ONTO and DB (the number of results)					Closures from DB only (the number of results)				
	Worker(3)	faculty(25)	student(22)	sno(14)	matricNo(9)	Worker(1)	faculty(16)	students(1)	sno(12)	matricNo(7)
Person	√	√	√							
Worker	√	√	√			√				
Student			√							
Faculty		√	√							
Course		√	√							
deptNo		√		√			√		√	
sex		√	√	√	√		√		√	√
school		√	√	√			√		√	
title		√		√			√		√	
position		√		√			√		√	
studentNo			√							
studentName			√							
major			√							
address		√	√	√	√		√		√	√
E-mail			√							
courseNo		√	√							
courseName		√	√							
creditHour		√	√							
projectNo				√	√					
projectName		√		√	√		√			
sno		√	√	√			√		√	
fname		√		√	√		√		√	√
lname		√		√	√		√		√	√
tel		√		√			√		√	
office		√		√			√		√	
pno		√		√	√		√		√	√
matricNo					√					√
creditSoFar					√					√
courseID		√	√							
lecturerNo		√	√							
staff	√	√	√				√			
faculty		√	√				√			
students			√					√		
courses		√	√				√			

6. Acknowledgements

This work is partly sponsored by the a grant for Distributed Information Systems Research from the Carnegie Trust for Universities of Scotland, 2007, a grant for research on Semantic Interoperability between Distributed Digital Museums from the Carnegie Trust for Universities of Scotland, 2009, and a PhD studentship of the University of the West of Scotland, UK.

REFERENCES

[1] T. R. Gruber, "A Translation Approach to Portable Ontologies," *Knowledge Acquisition*, Vol. 5, No. 2, 1993, pp. 199-220.

[2] M. West, "Database and Ontology [Online]," 2008. Wiki HomePage. http://ontolog.cim3.net/cgi-bin/wiki.pl?DatabaseAndOntology

[3] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web," *Scientific American*, Vol. 284, No. 5, 2001, pp. 34-43.

[4] Z. M. Xu, S. C. Zhang and Y. S. Dong, "Mapping between Relational Database Schema and OWL Ontology for Deep Annotation," *Proceedings of the* 2006 *IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society, 2006, pp. 548-552. http://portal.acm.org/citation.cfm?id=1248823.1249215&coll=AC

M&dl=ACM&CFID=16616566&CFTOKEN=44022427

[5] C. B. Necib and J. C. Freytag, "Query Processing Using Ontologies," *Proceedings of* 17*th International Conference on Advanced Information Systems Engineering,* Springer, Porto, Portugal, 13-17 June 2005.

[6] K. Munir, M. Odeh and R. McClatchey, "Ontology Assisted Query Reformulation Using the Semantic and Assertion Capabilities of OWL-DL Ontologies," *Proceedings of the* 2008 *International Symposium on Database Engineering & Applications*, ACM, Coimbra, Portugal, 2008, pp. 81-90.

[7] J. Feng, "The 'Information Content' Problem of a Conceptual Data Schema," *Systemist*, Vol. 20, No. 4, 1998, pp. 221-233.

[8] K. Xu, J. Feng and M. Crowe, "Defining the Notion of 'Information Content' and Reasoning about it in a Database," *Knowledge and Information Systems*, Vol. 18, No. 1, 1 January 2009, pp. 29-59

[9] F. I. Dretske, "Knowledge and the Flow of Information," MIT Press, Cambridge, 1981.

[10] K. Loney, "Oracle Database 10g: The Complete Reference," McGraw-Hill Companies, Inc., NY, 2004.

[11] B. C. Grau and B. Motik, "OWL 1.1 Web Ontology Language: Model-Theoretic Semantics. W3C Working Draft [Online]," 8 January 2008. W3C. http://www.w3.org/TR/owl11-semantics/

[12] Z. M. Xu and Y. J. Huang, "Conversion from OWL Ontology to Relational Database Schema," College of Computer and Information Engineering, Hohai University, Nanjing, 2006.

