BIRZEIT UNIVERSITY

FACULTY OF HIGHER EDUCATION

MASTER IN COMPUTING

SEMINAR II

Mapping Relational Database into ORACLE RDF

Prepared By

YASER ABURRUB, 1095174

Supervised By

DR. MUSTAFA JARRAR

DR. WASEL GHANEM

BIRZEIT

JAN - 2011

TABLE OF CONTENTS

Acknowledgement

I have exerted a big effort in this seminar; it would not have been possible without help, so I would like to thank and appreciate everyone for their support.

I'm grateful to Dr. Mustafa Jarrar for his guidance and supervision, he also provided me with important information regarding the project, and the extra help in completing my seminar. I would like to express my appreciation to my parents, our seminar staff for their encouragement in completing the seminar. My thanks and appreciations also are given to my colleague in writing the seminar, and all people who have strongly helped me with their abilities.

Abstract

Nowadays, most companies try to offer online services to people that are available on the internet to be accessed easily, usually these services deal with relational databases and need extra information from other company's database to complete their functionality. This means that huge amount of data should be integrated together and stored with complex structures and relations; therefore, the idea of integrating data by transforming complex relational schemas into simple graph databases started to appear.

Graph database has a unified simple structure consisting of Subject, Predicate and Object, thus when different relational databases are transformed into graphs, and they have the same structure, and can be merged and integrated into one graph table. Data integration is also achieved when we apply different SQL statements on the generated graph database.

We are going to implement a tool that takes more than one database; each one has different structure, then converting them into RDF triples to dump them into ORACLE RDF table.

Problem Specification

we developed an application that integrates multiple relational databases by transforming them into RDF triples, then adding these triples into Oracle SPO table. The algorithm is developed by merging W3C R2RDF algorithm [1]. Another algorithm was used to extract features that are proposed in [2].

What is Data Integration

Database integration is the process when different applications store their data in a specific database called the integration database, which enables the data to be available across all of these applications. The data is being shared between these applications and can be accessed easily across them; the implementation into a new application becomes easier without the need to transfer it into a different database.

To achieve a successful integration, there should be a plan takes into account all client applications. It is not necessary whether the scheme is complicated or general because there is a specific group controls the database to connect between the different applications and the database group.

The basic reason for implementing database integration is that it allows data sharing through the organization without the need of using another integration services for each application. Doing a specific method for integrating different applications is a waste of resources, by implementing database integration, the organization can save its resources and in the same time information can be automatically integrated whenever data is needed to be pulled or accessed.

Another reason, database integration helps companies that are merged together since each one of them has its own database, by integration their data can be connected easily. Integrating before a merger is definitely ideal.

Companies nowadays are investing more and more in database integration technology, especially as the quantity and connectivity of data increases. People need to access more data and share it between departments, so companies have realized to have all their data to be integrated on a database.

Chapter 1:

Introduction:

The concept of storing data in structured formats started to come out early in 1946, it was called Network Data Model, in which data is presented as records and there are relations between them. This model was known by the representation of complex structures of data, and then it disappeared because it was built on the concept of pointers and relations which made it very slow [5]. After that, more than one model appeared started from Hierarchical Model up to Relational Model. In the relational model, data is presented as structured tables with primary keys, foreign keys link between them and other constraints to keep both entity and referential integrity [5]

RDF is the latest data model that has simple XML format, RDF is presented as a single database table with three columns called SPO, S stands for “subject” to represent the entity, P stands for “Predicate” to represent the relation and O stands for “Object” to represent the idea. RDF has a schema called RDFS which is the constraint that might apply on RDF data. Many algorithms were developed for mapping a relational database into RDF. [1] Put specifications for direct mapping between Relational Database and RDF. [2] Introduced a systematic way for mapping relational database into RDF by extracting database features such as Aggregation, PartOf and Inheritance relations.

Two main goals motivate us to search in this topic and write the report, when mapping different relational databases with different schemas into a unified structure, first, it can be used in Data Integration which is the process of standardize and unify different database schemas, or data heterogeneity into another unified data format while keeping the original data semantics and constraints. This mechanism gives us the ability to make queries on the integrated data, and retrieve significant results [3].

The second goal is; we can use inference rules, and other features supported by Oracle RDF when mapping different Relational Databases into Oracle RDF. For example, if we have the

following two facts in RDF, {Yaser lives in Ramallah}, and {Ramallah city is in Palestine} then we can conclude the fact {Yaser lives in Palestine}.

By mapping different relational databases into one single RDF table which has the same structure, this will be used to integrate multiple database into one database. Thus, the integration gives the power to make queries on data after merging from different schemas and to extract if some contradictions and replications found in the original data.

In this seminar, we will propose an algorithm to map multiple relational databases into Oracle Semantic RDF (OSR), in chapter 2 we are going to present a literature review covering Relational Database, Resource Description Framework, Oracle Semantic Technology, W3C Direct Mapping Algorithm and Relations Extraction. In chapter 3 we will present a methodology for our mapping algorithm, then, in chapter 4 and 5 we will discuss the project that maps the relational database to OSR, and a case study respectively. Finally, in last chapter we will present some conclusions, and some future work.

Chapter 2:

Literature Review

In order to fully understand the mapping process between relational database and Oracle RDF, the following topics will be covered in the literature review including Relational Database, Resource Description Framework (RDF), and Oracle RDF with inference and data retrievals, W3C Mapping, and Database features extraction.

2.1 Relational Database

Database is defined by [4] as the collection of structured data mainly shared or related to a specific field. If data is stored in table data structure linked by different relations, it is called Relational Database (RM).

Database Management System (DBMS) is software responsible for creating, managing and maintaining database. It provides indexing, views, data integrity, security, backup/recover and other useful functions.

2.1.1 Database Schema

Database Schema, and is also called Database Dictionary is a location where the description and other statistics about the database is stored. It is called 'database about database', it is used by DBMS in Query Optimizer and is also used for storing different database attributes including tables names, fields names, data types, primary keys, foreign keys and other attributes [5]. DBMS provides an Application Programming Interface (API) to query all the database schema data in our project.

2.1.2 Database Constraints

DBMS provides multiple constraints that might be applied on data to keep both data validity and integrity. The first constraint is the Super Key (SK) which is defined by [5] as "an attribute or set of attributes that uniquely identify a row in a relation". The second constraint is the Candidate Key (CK) which is defined by [5] as "the minimal super key". The third constraint is the Primary Key (PK) which is defined by [5] as "a candidate key which is selected to uniquely identify a

row in database table and is not allowed having NULL value". The fourth constraint is the Foreign Key (FK) which is defined by [5] as "an attribute or set of attributes to reference a table that matches a candidate key in referenced table in some relation".

2.2 Resource Description Framework

It is a metadata data model designed as one of the W3C specifications. It is implemented as a general method for modeling and representing data in web resources using different syntax formats. RDF is based on making forms known as subject-predicate-object expressions, they are called triples. Subject refers to the resource; predicate refers to the aspects of the resource that defines the relationship between the subject and the object. RDF data is used in relational databases and native representations which are called Triplestores or Quadstores [9].

There were many motivations behind the development of RDF, among these, the applications that need open information models instead of constrained models such as annotation of web resources, allowing data to be processed outside the environment in which was created in order to work at internet scale, the internetworking among applications and the automated processing of web information by software agents. The RDF technology provides information with least constraints and in a flexible way so as to be used in isolated applications that has special formats, RDF facilitates those formats to be more direct and easily understood, giving greater value since it becomes more accessible to more applications across the entire internet[10].

RDF uses two common serialization formats; the first is the XML format which is also called RDF referring to W3C specifications that defined RDF. The second one is the N3 (Notation 3) as non-XML serialization of RDF models, this method was found to be easily written by hand, and to follow. The difference between the two types is that N3 serialization makes the underlying triples encoded in documents easier to be recognized compared to XML serialization. Resource identification: Subject in RDF terminology is either a URI or a blank node. The two are known as resources. Blank nodes are called anonymous resources, which are not identified directly from the RDF statement. The URI represents a relationship whereas the predicate is a URI which identifies a resource. RDF is not limited to the description of internet based resources. Any URI that refers to resource should not be distinguishable [9], for example; if we have a URL begins

with "http", and denotes a subject in an RDF statement does not have to represent a resource accessible by HTTP.

Semantics of resource identifiers must be agreed on between producers and consumers of RDF statements, the agreements are controlled by some vocabularies in common use such as Dublin Core Metadata. Publishing RDF based-Ontology help establishing the intended meanings of the resource identifiers that are used to express data in RDF.

If we have the following CD-list relation R={id, title, artist, country, company, price, year}:

Id	Title	Artist	Country	Company	Price	Year
1	Inside My Heart	John Isaq	Germany	ASB dem	20	1988
2	Age of Empire	Criss Jerry	Italy	Columbia	15	1994

Then we can represent using RDF as follows

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.cdlist.com/cd#">

<rdf:Description
rdf:about="http://www.cdlist.com/cd/Inside My Heart">
  <cd:id>1</cd:id>
  <cd:artist>John Isaq</cd:artist>
  <cd:country>Germany</cd:country>
  <cd:company>ASB dem</cd:company>
  <cd:price>20</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>

<rdf:Description
rdf:about="http://www.cdlist.com/cd/Age of Empire">
  <cd:id>2</cd:id>
  <cd:artist>Criss Jerry</cd:artist>
  <cd:country>Italy</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>15</cd:price>
  <cd:year>1994</cd:year>
</rdf:Description>

</rdf:RDF>
```

The first line in RDF file is an XML declaration, then RDF root element which is rdf:RDF, after that we define the used namespaces, in our example we have rdf and cd namespaces. After that for each entity we should declare an XML node specifying its IRI i.e. “Inside My Heart” and “Age Of Empire”. Then we list all of its attributes inside it by building IRI for each attribute.

2.3 Oracle Semantic Web Technology

Oracle has released a package installed over Oracle 11g played as a database for semantic data; it is called Oracle Semantic Web Technology (OSWT). It is an external package installed over Oracle 11g; it supports different ways to store and retrieve semantic data.

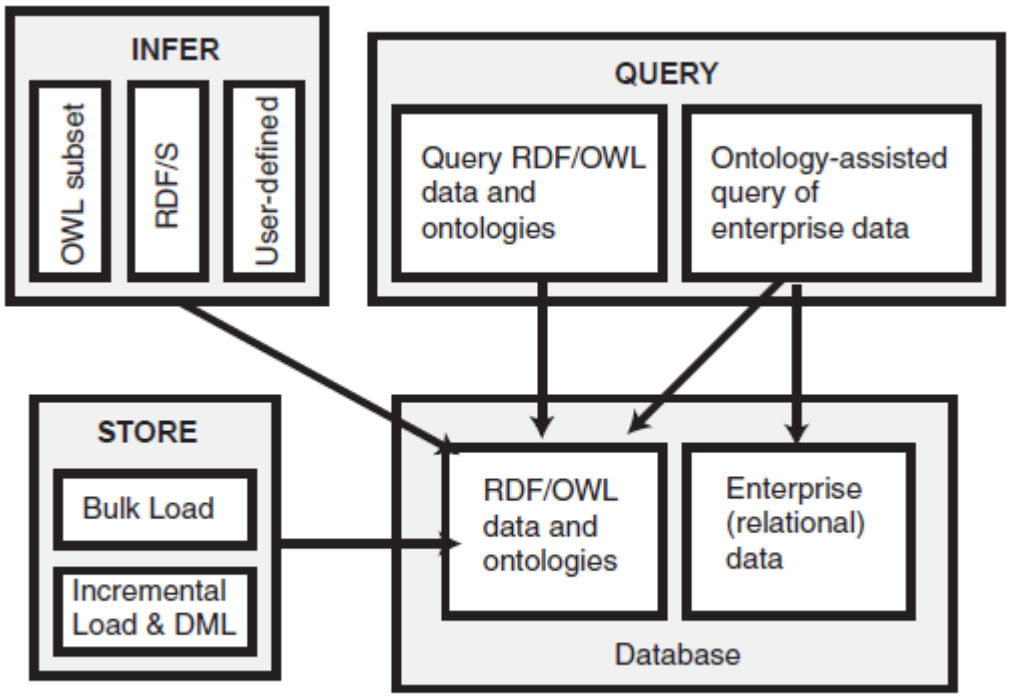

Figure.01 : Oracle Semantic Capabilities

Figure 1-1 [7] depicts the capabilities of Oracle product, the database block contains both RDF/OWL data stored in triples form and ontology, OSWT provides Bulk Load and Incremental Load methods to input data. The INFER section provides some functions and methods to predict or conclude some rules from the original facts stored in database. For example, if we have the two rules R1= {Mohammad-LiveIn-Ramallah} and R2= {Ramallah-City In-Palestine} then we can conclude R3= {Mohammad-Live In-Palestine}.We can get advantage from this process when we migrate data from different systems. Query block is used for querying data by SEM_MATICH function and inferring methods using Rule-Bases.

Data stored in OWTS in triple form is called SDO_RDF_TRIPLE_S, it is an object that contains three fields or values, Subject and Object represent entities, Predicate is presented as a directed arrow that goes from Subject to Object. To be able to store these triples in database, first we have to create a semantic data network, then; we create a semantic data model inside, after that triple Objects are stored within these models by passing network, model names as parameter to SQL statement [7]. Entities inside triples should have an IRI form (Internationalized Resource Identifier), it is a generalization of the Uniform Resource Identifier (URI). The IRI contains characters from the Universal Character Set in order to include Chinese, Japanese and other complex characters [6]. There is a methodology we are going to talk about in mapping algorithm to generate an IRI for each database value.

When data are inserted into RDF table via either bulk load or incrementally, we can use SPARQL SQL statements to query the required data, the following table shows a SPARQL statement that retrieve all (m,y) values, such that m is an rdf:type for y.

```
SELECT m, y
FROM TABLE(SEM_MATCH(
'(?m <http://www.w3.org/1999/02/22-rdf-syntax-ns#rdf:type> ?y)',
SEM_Models('rdfdatamodel'),
null,
SEM_ALIASES(SEM_ALIAS('','http://mop-gov.ps/gpcdb/')),
null));
```

2.4 W3C Direct Mapping Algorithm

Strong features of relational databases including efficiency and precise definitions allow tools like SQL to handle and examine contents in an efficient way. The RDF is a data format implemented as a general method for modeling and representing data in web resources, in this section we will present the mapping from relational representation to the RDF representation. The direct mapping method will be illustrated here as a simple transformation to build the basis for the complex transformations. This method enables RDF graphs to be materialized and to define virtual graphs.

Direct Mapping Description (Informative) is the RDF graph representation of the data in the relational database is shown by a set of common data types. The input is the relational database both data and schema, to generate the direct graph. The reference from any row in a table is only one row in a different table is achieved by using foreign keys. Everything in the relational database is moved to the direct graph including references and data. Now we will present an example of how we directly mapping relational database into RDF based on W3C R2RDF algorithm, we will start by figure 2 which is an SQL DDL sample code to create two tables (Addresses, People), each of one column primary key, related to each other by a foreign key addr(ID).

```
CREATE TABLE Addresses (
     ID INT, PRIMARY KEY(ID),
     city CHAR(10),
     state CHAR(2)
)
CREATE TABLE People (
     ID INT, PRIMARY KEY(ID),
     fname CHAR(10),
     lname CHAR(10),
     addr INT,
     FOREIGN KEY(addr) REFERENCES Addresses(ID)
)
INSERT INTO Addresses (ID, city, state) VALUES (18, 'Cambridge',
'MA')
INSERT INTO People (ID, fname, addr) VALUES (7, 'Bob', 18)
INSERT INTO People (ID, fname, addr) VALUES (8, 'Sue', NULL)
```

Table I: Sql statement for creation of Addresses and people Data [1]

The primary key is marked by PK. Foreign key is shown using the notation "? Address(ID)" to denote the SQL foreign key defined above.

	People		
PK			→ Address(ID)
ID	**Fname**	**Lname**	**Addr**
7	Bob	John	18
8	Yaser	Aburrub	20

Addresses		
PK		
ID	**City**	**State**
18	Cambridge	MA
20	Ramallah	Palestine

Table II: Sample data for Addresses and people Data tables [1]

For the example shown above and using the following IRI http://foo.example/DB/, we get the following direct graph triples.

- (People?id=1, rdf:type, People)
- (People?id=1, Fname, Bob)
- (People?id=1, Lname, John)
- (People?id=1, addr, 18)
- (:addr, rdf:domain, People)
- (:addr, rdf:range, Address)
- (People?id=2, rdf:type, People)
- (People?id=2, Fname, Yaser)
- (People?id=2, Lname, Aburrub)
- (People?id=2, addr, 20)
- (Address?id=18, rdf:type, Address)
- (Address?id=18, city, Cambridge)
- (Address?id=18, state, MA)
- (Address?id=20, rdf:type, Address)
- (Address?id=20, city, Ramallah)
- (Address?id=20, state, Palestine)

Table III: RDF entries for Addresses and people Data tables [1]

The row like (7, "Bob", 18), gives a group of triples with a common subject. The subject here is an IRI which is composed by concatenating: base IRI, table name, primary key column name, primary key value. The predicate for each column us an also another IRI formed by concatenating: base IRI, table name, column name. Triples are formed by foreign keys; those triples are with predicates formed by foreign key column names, the referenced table, and the referenced column names. The row identifiers like ((<Addresses/ID-18>) are the objects of theses triples. The direct method of mapping does not generate triples for NULL values [1].

2.4.1 Relations Extraction

RDF is a simple form for expressing facts by two entities and a relation among them, we can view it as a relational database table of three columns Subject, Object and Predicate. Thus, all mapping algorithms focused on this point, taking into consideration that these values must be unique across the whole RDF table. To achieve this property we identify IRIs for entities by taking the domain, schema and table [2].

Example: if we have 'GPCDB' database located at domain 'HTTP://SOMEDOMAIN.COM', and we want to express the fact ('STUDENT' : ('ID', 'FIRST_NAME') : ('123456', 'YASER

ABURRUB')), the first part of the IRI will be generated by concatenating the domain name with the database name 'HTTP://SOMEDOMAIN.COM/GPCDB'.

The Primary Key (PK) identifies uniquely each row in the database table, we depend on the primary key in forming distinct triple form, back to our example the PK is '123456' and the fact value is 'YASER ABURRUB', so the second part will be:
'STUDENT?id=123456&FIRST_NAME=YASER ABURRUB'.
The value for the IRI is:
'HTTP://SOMEDOMAIN.COM/GPCDB/STUDENT?id=123456&FIRST_NAME=YASER ABURRUB'
We certainly know that concatenating the two parts together; will produce an absolute IRI over the World Wide Web if we assume no identical domains.

Multiple research papers focused on extracting additional features other than the direct mapping algorithms, [3] presents an algorithm to extract associations, inheritance, aggregation and cardinality relations, the process starts by identifying the entities. Two conditions must be met on a table to consider it as an entity table. First, each table with single attribute primary key is an entity table, i.e. assuming that table Account={accountID, CustomerId, balance}, in RDF we have an 'Account' entity. Second, the table is an entity if it has a primary key that has more than one field, and at least one of them is not a foreign key, i.e. if we have table 'Payment'={loanId is a foreign key (FK), paymentId, paymentDate}, then Account is an entity.

Here we describe how to extract association relation from relational database, "Association between tables is based on two cases. First, each table whose primary key has more than one attribute and is entirely composed of foreign keys represents an association between all the entities corresponding to the foreign keys. The association is actually many-many association. The corresponding table is called an association-table. If the association-table has an own attribute, it is referred to as an entity-table.
We establish an association between an entity and the corresponding entity for each foreign key, unless it also happens to be a part of the primary key" [3].

TABLE I. EXAMPLE OF RELATION SCHEMA

Table list	Attribute list	Foreign key and referenced relation
Account	accountID,customerID, isbalance	customerID(customer)
Customer	customerID, name, city	no
Subbank	bankID, bankname	no
Loan	loanID, customerID,loanAmount	customerID(customer)
SavingAccount	accountID, interrate	accountID(account)
CheckingAccount	accountID, overdraftNumber	accountID(account)
Establish	bankID, customerID, edate	bankID (subbank),customerID(customer)
Payment	loanID, paymentID, paymentDate, paymentAmount	LoanID(Loan)

Table IV : Table for Business Database Schema [2]

Looking at Table IV shown above, we conclude that Establish satisfies condition 1, it is an association table because it has a primary key of two attributes, and each of them is a foreign key. Also we have an association between (Account, Customer) and (Account, Loan). Applying condition 2 to identify association, generates relations between (Account, SavingAccount) and (Account, CheckingAccount).

After extracting the entities that have inheritance relationship, then we move to get all entities that have inheritance relationship. The process starts by "Every pair of entity-tables (ET1, ET2) that have the same primary key may be involved in an inheritance relationship. The inheritance relationship ET1 "is-a" ET2 exists if the primary key of ET1 is also a foreign key and that refers to the entity-table ET2" [2]. By applying this condition to the relation schema in figure 1, there is an inheritance relation between (SavingAccount, Account) and (CheckingAccount, Account), since both combinations of tables have the same primary key (accountID). The next step is to identify aggregation tables, if we have an entity table 'A' with primary key consisting of more than one field, and at least one of them is a foreign key to table B, then A is part-of B, back to figure 1, we simply conclude that Payment is part-of Loan [3].

Chapter 3:

Methodology

In this chapter we introduce in details our mapping algorithm, which is built on merging W3C algorithm with the feature that we extracted in section 2.4.1.

3.1 Dynamic Reading of Database Schema

The only parameter that our application used is the database schema name, from this value we established a connection between Microsoft Visual Studio and Postgresql database through the 'Devart PostgreSQLDirect .NET' connector by specifying the database username, password and database server. Then, we started reading information by using 'GetSchema' API function, for each table we built an object of type 'dbtable' by reading table name, its primary keys and foreign keys including the FK's name, column and the referenced table as shown in figure.03.

```
tablesFromSchema = Me._connection.GetSchema(RestrictionName.Tables.ToString,
tableRestrictions)
For Each tableRow As DataRow In tablesFromSchema.Rows

tableName = tableRow.ItemArray(TableFieldFromSchemaIndex).ToString
Dim tableObj As New dbtable(tableName.ToLower)

primaryKeyRestriction(1) = tableName ' for each table we should get primary keys
primaryKeysFromSchema = Me._connection.GetSchema(RestrictionName.PrimaryKeys.ToString,
primaryKeyRestriction)
' save primarykey columns
For Each primaryKeyRow As DataRow In primaryKeysFromSchema.Rows
  For Each pk As String In getPrimaryKeyColumnFromPrimaryKeyName(primaryKeyRow)
    tableObj.appendKey(TableFieldType.PRIMARY_KEY, pk)
  Next
Next

foreignKeyRestriction(1) = tableName ' for each table we should get primary keys
foreignKeysFromSchema = _connection.GetSchema(RestrictionName.ForeignKeys.ToString,
foreignKeyRestriction)

For Each foreignRow As DataRow In foreignKeysFromSchema.Rows
tableObj.appendFK(TableFieldType.FOREIGN_KEY,getForeinKeyColumnFromPrimaryKeyName(foreign
Row))
Next
```

Figure0.3 Reading Primary and Foreign Keys snapshot

3.2 Features Extraction

Now, all the required data is saved in its own classes; the next step is to extract all required features from the database.

3.2.1 Entity Tables

The first extracted feature from the database was to decide whether a table is an entity or not. We marked a table as an entity if its primary key has either a single attribute or at least one its fields is not a foreign key. Figure 5 below shows 'SetEntityTable' function which is used to determine if a table is an entity table or not.

```
Public Sub SetEntityTable()
    Dim result As Boolean = False
    result = If(COUNT(TableFieldType.PRIMARY_KEY) = 1, True, False)
    For Each fkname As String In Me._foreignKey.Keys
        If Me._foreignKey(fkname) Is Nothing Then
            result = True
            Exit For
        End If
    Next
    Me._entityTable = result
End Sub
```

Figure0.5 Check Whether table is entity or not

Back to Table IV, all tables are entity tables except Establish because its primary key consist of two field pk1 and pk2 and both of them are foreign keys to tables subbank and customer respectively.

3.2.2 Association Relationship

Association MxN between the two is defined in two cases. First, if we have an entity table ET1 whose primary key is composite form more than one field i.e. pks_s1, pks_s2, all of them are foreign keys referencing tables S1 and S2 respectively. Then we have the following association relation R1={ET1, S1} and R2={ET1, S2}. Second, we establish an association relation between entity table ET1 with the entity table ET2 corresponding to each foreign key unless the foreign key is not a part of the primary key. Figure 0.6 shows part of the code that is responsible for extracting association relation. For each table stored in our schema applying either the first or the second condition, we add an association relation into 'association' data structure.

```
For Each tbl As dbtable In Me.tables.Values
If tbl.IS_ENTITY_TABLE And Not tbl.FKs Is Nothing Then
For Each fk As dbfk In tbl.FKs
e2e = True
For Each pk As String In tbl.PKs
If fk.FOREIGNKEY = pk Then
e2e = False
Exit For
End If
Next
If e2e Then
```

```
sharedModule.insertWithoutReplication(tbl.NAME, Me.associationTables)
sharedModule.insertWithoutReplication(fk.FK_TABLE, Me.associationTables)
addNewAssociation(New association(tbl.NAME, fk.FK_TABLE))
End If
Next
End If
```

Figure0.6 Extraction of MxN Association Relation

Back to Table IV, we have association relation between (Account and Customer), and (Customer and Loan).

3.2.3 Inheritance Relationship

Inheritance relationship is found between two entity tables ET1 and ET2, if both tables share the same primary key, and the primary key for ET1 is a foreign key refers to ET2. Figure 0.7 shows snippet of code that defines inheritance.

```
For Each t1 As dbtable In Me.tables.Values
    fc = If(t1.FKs Is Nothing, 0, t1.FKs.Count)
    If Not (t1.NAME = tbl.NAME) And fc = 1 Then
        If t1.hasExactField(TableFieldType.PRIMARY_KEY, pk) And t1.hasExactField(TableFieldType.FOREIGN
            Me.AbstractionsRelations.Add(New relation(tbl.NAME, t1.NAME, RelationType.IS_A_RELATION))
            Continue For
        End If
    End If
Next
```

Figure0.7 Code responsible for extracting inheritance relation

Back to Table IV, Account and CheckingAccount have the same primary key, and primary key for checkingAccount is a foreign key to Account table, so checkingAccount has inheritance relation with Account.

3.2.4 Aggregation Relationship

Aggregation relation is also called partOf relation; it is identified between two entity tables ET1 and ET2, if ET1 has a primary key from more than one attribute, and one of them is a foreign key refers to ET2. Figure 0.7 depicts the code that extracts tables that have Aggregation

relationship.

```
If tbl.et Then
    Dim fks_in_pk As String() = tbl.getAllForeignKeysInPrimaryKey2
    Dim fk_in_pk_flag As Boolean = If(fks_in_pk Is Nothing, False, True)
    If fk_in_pk_flag And tbl.IS_ENTITY_TABLE And tbl.PKs.Count >= 2 Then
        For Each t1 As dbtable In Me.tables.Values
            If Not (t1.NAME = tbl.NAME) And t1.IS_ENTITY_TABLE Then
                For Each fkValue As String In fks_in_pk
                    fkAsPkFlag = False
                    For Each pkValue As String In t1.PKs
                        If fkValue = pkValue Then
                            fkAsPkFlag = True
                            Exit For
                        End If
                    Next
                    If fkAsPkFlag Then
                        Me.AbstractionsRelations.Add(New relation(tbl.NAME, t1.NAME, RelationType.PART_OF_RELATION))
                    End If
                Next

            End If
        Next
    End If
```

Figure0.7 Extraction of Part-Of Relation

Back to Table IV, Payment PK consists of two attributes, paymentID is not FK, also loanID is a foreign key reference to the table Loan, so we have an aggregation relation between Payment and Loan.

3.3 Transformation Process

The next step after features extraction is to map the database with extracted feature into Oracle RDF. RDF is a structured format of data that is formed by Subject, Predicate and Object, we need an algorithmic procedure that transforms different database schemas into unified RDF structure, the transformation should take into account how the inheritance, and aggregation relations can be embedded within raw data.

3.3.1 Mapping of raw data:

Raw data is the actual data stored in the database; it is the largest amount of data that will be mapped into RDF, most of our time was spent doing this step. A Primary Key (PK) is a unique value for each row in database table; we can use it to build Subject IRI which becomes unique across RDF table. Predicate IRI is created from column name except the primary key. Object is the actual table field value. A second point in RDF table; we should present to which table a foreign key refers, this will be done by rdfs:domain, and rdfs:range, i.e. assume we have a table R={**A:1**,B:2,C:3}, and C is a foreign key that refers to table R2={**id:3,** value:20}, then RDF triples reflects this table are:

- {A, B, 2}

- {A, C, 3}
- {C, rdf:domain, R}
- {C, rdf:range, R2}

3.3.2 Mapping of relations:

The only two relations that are converted into RDF triples are inheritance and aggregation relations. Entity table and association relation are used to extract inheritance and aggregation, so both were discarded and were not stored as RDF triples. Inheritance relation is presented in triple form by using RDFS keywords subClassOf and partOf respectively, i.e. if we have an inheritance relation R between two entity tables A and B, then this would be expressed in RDF as the following.

- {A, rdfs:subClassOf, B}

Chapter 4:

Application

Figure 1.03 shows our application which is called ‘Schema2Rdf’ that takes multiple relational database schemas and maps them into Oracle RDF entries. In this chapter we present general requirements for running the application, discussion of the application structure, introduction of the applied methodology, and finally we present an example for mapping sample database schema into RDF.

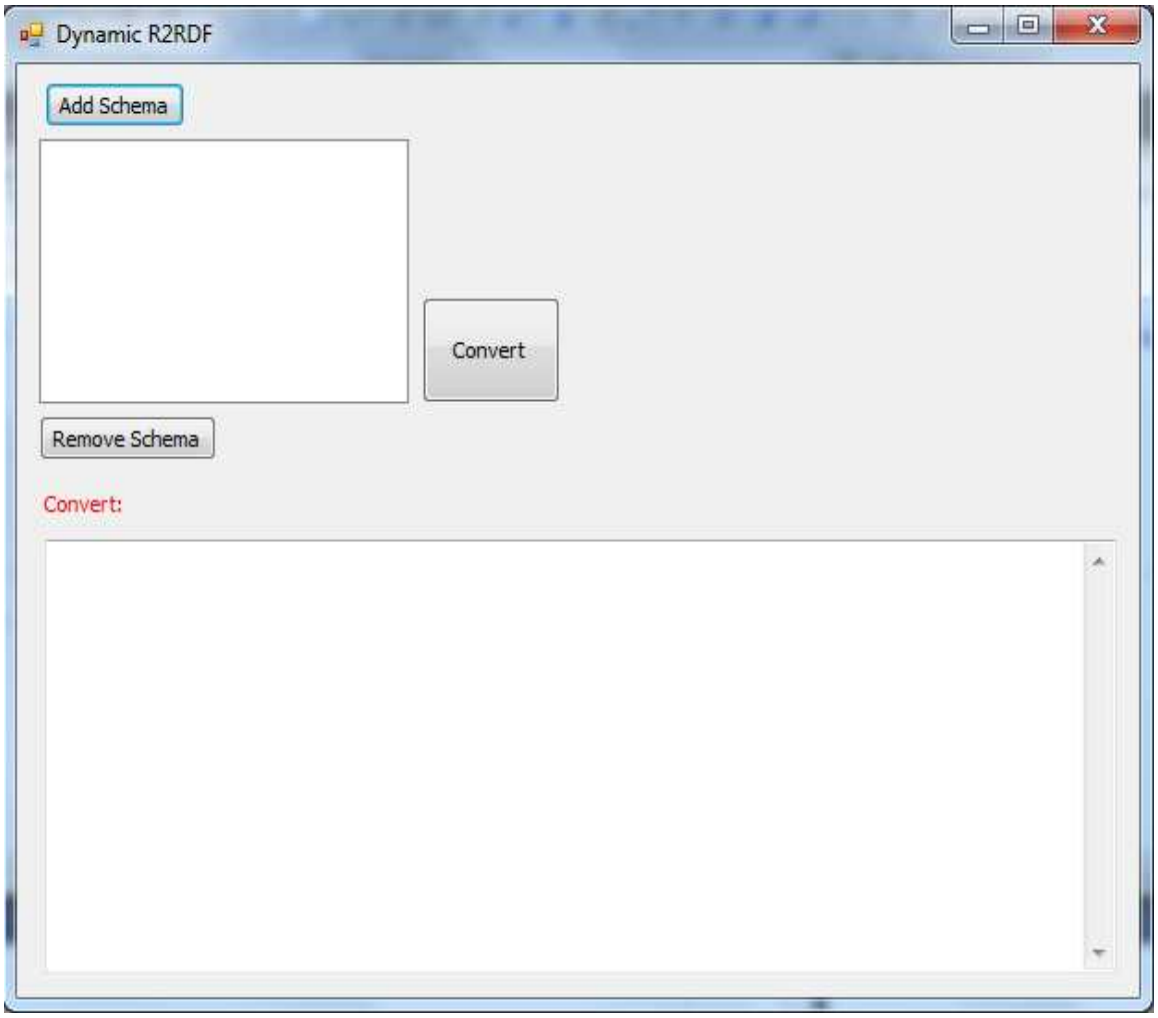

Figure 1.03 R2RDF Interface

4.1 Application Requirements

‘Schema2Rdf’ were developed under Visual Studio Framework (VSF), some applications and drivers should be installed to complete running the application. Any application programmed

under VSF need to be run under Microsoft Windows Operating System, and needs the Microsoft Dot Net Framework driver which “provides a comprehensive and consistent programming model for building applications that have visually stunning user experiences and seamless and secure communication [8]”.

Input data for our application is Postgresql database, so we installed Postgresql database on our machine. VSF and Postgresql are completely different applications; we needed a middleware driver that helped us to read Postgresql data within VSF. ‘Devart PostgreSQLDirect .NET’ should also be installed. The output data for mapping procedure was Oracle RDF triples, so we installed Oracle Semantic Technology package over Oracle 11g.

4.2 Application Structure

Schema2Rdf is fully object oriented application, it has three main classes, each one of them has specific task. The important one is called ‘dbSchema’ which identifies the connection of the schema, read all database tables with its primary and foreign keys, and extract associations, aggregation and inheritance relations. The second class is ‘RdfMapping’ which is responsible for opening ngconnection into OWTS and applying the mapping algorithm. The last class is ‘dbIntegration’ which is used for unifying similar tables which has different names in each schema.

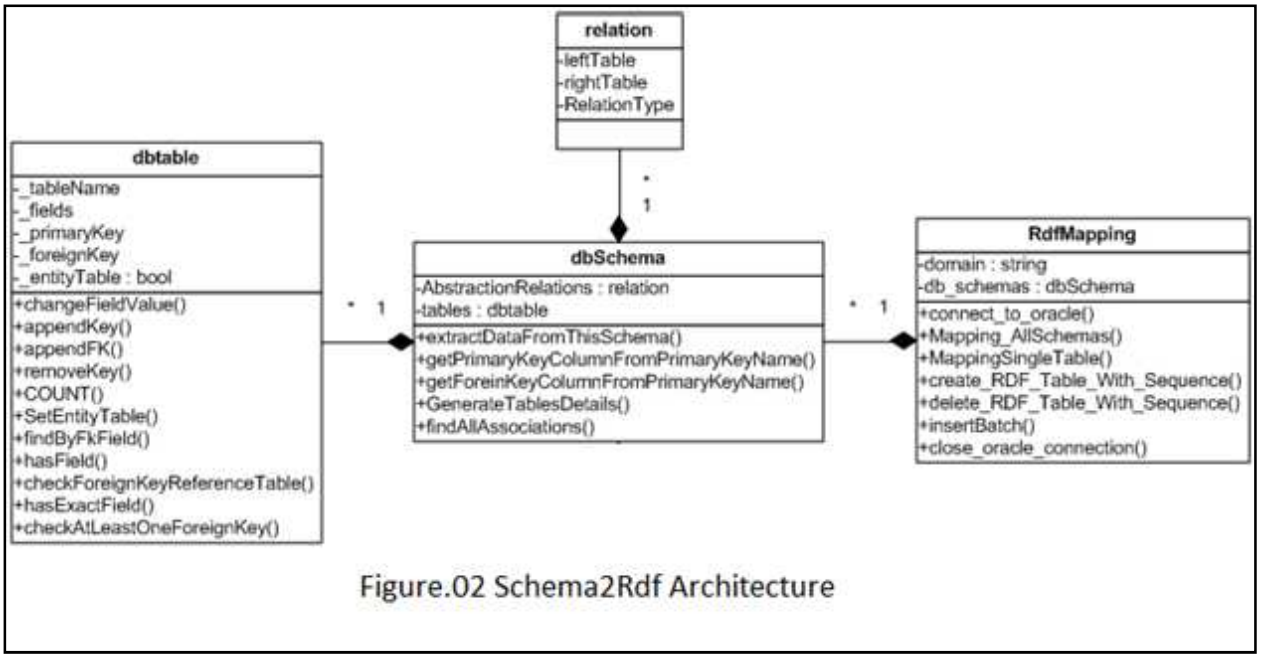

Figure.02 Schema2Rdf Architecture

Figure 02 shows software architecture for Schema2Rdf application. We stored all schema information in dbSchema, we showed the aggregation relation by `relation` class, it stores tables that have aggregation, inheritance or association relations. Also we have an aggregation relation with dbtable which stores all possible information about single table. RdfMapping is responsible for mapping process; we notice the references for all database schemas through aggregation relation.

Chapter 5:

Case Study

In this chapter, we present a real case about transferring more than one database schema into RDF table.

The first schema is called ‘business’ database, figure 1.4 shows the UML diagram for it, we use as a test database we have added some features like aggregation and inheritance, it is include the following tables { checkingAccount, customer, establish, loan, payment, savingaccount and subbank}, it consist about 20 tuples of data.

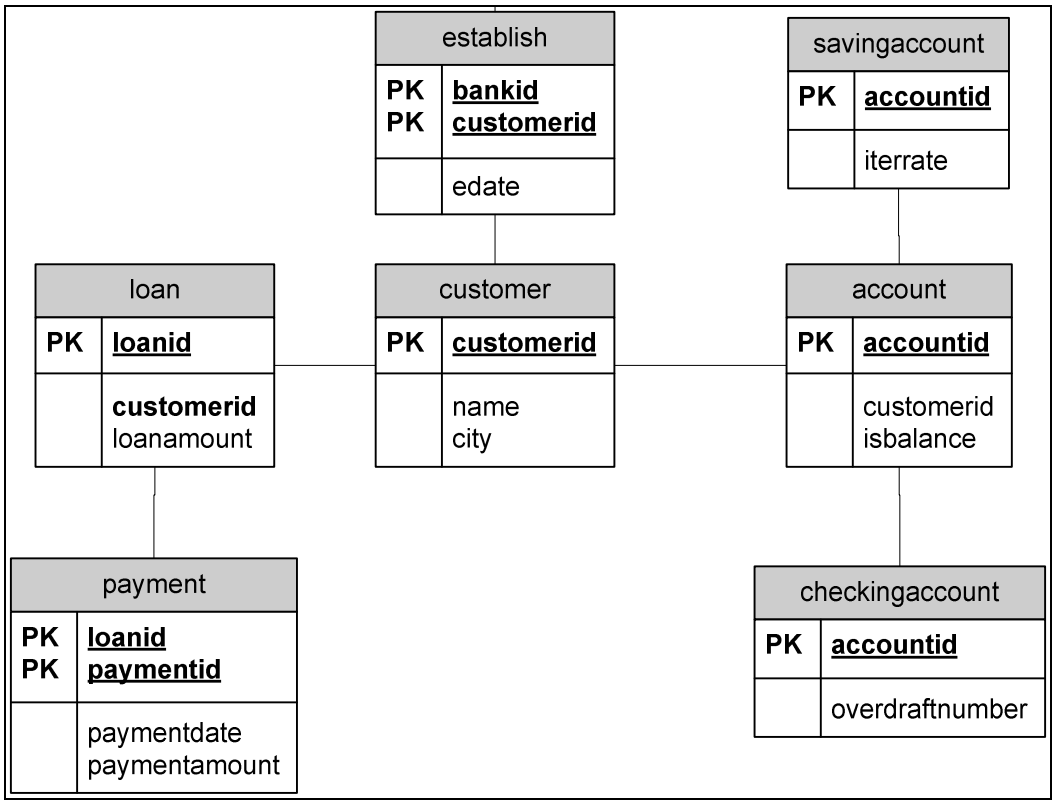

Figure 1.4 UML diagram for business database

Figure 1.5 shows some SPO entries from our application output when we run it over business database.

- {(http://mop-gov.ps/schema_1/subbank?bankid=4), (http://www.w3.org/1999/02/22-rdf-syntax-ns#rdf:type), (subbank)}
- {(http://mop-gov.ps/schema_1/subbank?bankid=4), (http://mop-gov.ps/schema_1/subbank?bankname), (http://mop-gov.ps/schema_1/subbank?obj=Palestine)}
- {(http://mop-gov.ps/schema_1/loan?loanid=1), (http://www.w3.org/1999/02/22-rdf-syntax-ns#rdf:type), (loan)}
- {(http://mop-gov.ps/schema_1/loan?loanid=1), (http://mop-

gov.ps/schema_1/loan?loanamount), (http://mop-gov.ps/schema_1/loan?obj=50)}

- {(http://mop-gov.ps/schema_1/loan?loanid=1), (http://mop-gov.ps/schema_1/loan?customerid), (http://mop-gov.ps/schema_1/loan?obj=1)}
- {(http://mop-gov.ps/schema_1/loan?loanid=2), (http://www.w3.org/1999/02/22-rdf-syntax-ns#rdf:type), (loan)}
- {(http://mop-gov.ps/schema_1/loan?loanid=2), (http://mop-gov.ps/schema_1/loan?loanamount), (http://mop-gov.ps/schema_1/loan?obj=100)}
- {(http://mop-gov.ps/schema_1/loan?loanid=2), (http://mop-gov.ps/schema_1/loan?customerid), (http://mop-gov.ps/schema_1/loan?obj=2)}
- {(http://mop-gov.ps/schema_1/loan?loanid=3), (http://www.w3.org/1999/02/22-rdf-syntax-ns#rdf:type), (loan)}
- {(http://mop-gov.ps/schema_1/loan?loanid=3), (http://mop-gov.ps/schema_1/loan?loanamount), (http://mop-gov.ps/schema_1/loan?obj=500)}
- {(http://mop-gov.ps/schema_1/loan?loanid=3), (http://mop-gov.ps/schema_1/loan?customerid), (http://mop-gov.ps/schema_1/loan?obj=1)}
- {(http://mop-gov.ps//schema_1/payment), (http://www.w3.org/2000/01/rdf-schema#partOf), (http://mop-gov.ps//schema_1/loan)}
- {(http://mop-gov.ps//schema_1/account), (http://www.w3.org/2000/01/rdf-schema#subClassOf), (http://mop-gov.ps//schema_1/savingaccount)}
- {(http://mop-gov.ps//schema_1/account), (http://www.w3.org/2000/01/rdf-schema#subClassOf), (http://mop-gov.ps//schema_1/checkingaccount)}

Figure 1.5 Some Of SPO triples resulted from business database

The second schema is real database schema for General Personnel Council GPC as shown in figure 0.9.

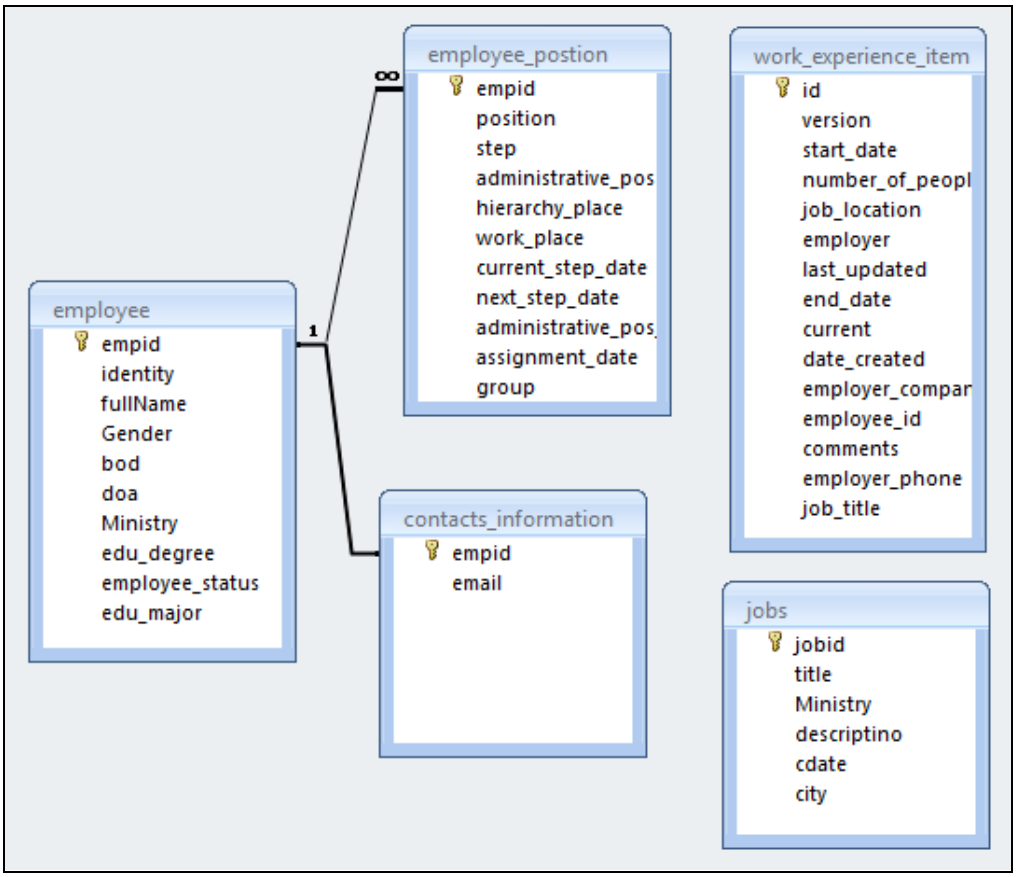

Figure 0.9 Schema for GPC database

This database is used to store information related to employees in General Personnel Council, it includes five tables {employee, contacts_information, employee_position, work_experience and jobs}, it contains about 500 records of data, figure 1.6 displays some of SPO records after transform this schema into RDF triples.

- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://www.w3.org/1999/02/22-rdf-syntax-ns#rdf:type), (employee)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?edu_degree), (http://mop-gov.ps/schema_5/employee?obj=بكالوريوس)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?employee_status), (http://mop-gov.ps/schema_5/employee?obj=على رأس عمله)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?fullName), (http://mop-gov.ps/schema_5/employee?obj=شهناز صالح احمد ابوعزه)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?identity), (http://mop-gov.ps/schema_5/employee?obj=992666552)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?Ministry), (http://mop-gov.ps/schema_5/employee?obj=وزارة التخطيط)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-

gov.ps/schema_5/employee?Gender), (http://mop-gov.ps/schema_5/employee?obj=انثى)}

- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?doa), (http://mop-gov.ps/schema_5/employee?obj=03/05/1998)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?bod), (http://mop-gov.ps/schema_5/employee?obj=10/22/1964 12:00:00 AM)}
- {(http://mop-gov.ps/schema_5/employee?empid=64706), (http://mop-gov.ps/schema_5/employee?edu_major), (http://mop-gov.ps/schema_5/employee?obj=لغة انجليزية)}
- {(http://mop-gov.ps/schema_5/employee?empid=93128), (http://www.w3.org/1999/02/22-rdf-syntax-ns#rdf:type), (employee)}

Figure 1.6 Some Of SPO triples resulted from GPCDB database

The goal from transform and integrate these two databases to get advantages that we talked about in data integration section, in addition to this give the user the ability of create and execute SQL queries over the generated database, and get the advantages and benefits that is provided by Oracle Semantic Technology such as inference rules capabilities.

When data for two ministries as example take Ministry of Planning and General Personnel Council are merged and integrated together, this will give us some hidden information, we can know if there are some replications in data specially when we are talking about personal data, most of ministries stored the personal information about their employees, so we could found that same employee will have two entries regards “firstname”, “lastname” or “age”. This replication will be slightly appeared when data are merged together. Another issue, RDF is a graph database so when we have one hundred employee who have gender of “Male”, we only need to create one node of value “Male”, and all male employees entities just has a line to this node, this will help us in minimize of stored data.

Many problems we were face it while implementing R2RDF software, the main problem is how dynamically reading data from database features; this is done by using some functions provided by DLL linker to the database management. Other problem we were face is to how to install Oracle Semantic Technology Package; this package needs to follow long steps of installing and activating complex batches and installers on Oracle 11g.

Chapter 6

Conclusion and Future Work

In this seminar we proposed a dynamic algorithm that maps multiple database schemas into a unified RDF model. It extracts additional semantics that are not found in the relational model such as inheritance and aggregation without losing any of the original relational model structures and constraints.

Future work will be focused on developing new algorithms to extract features; enriching the generated RDF. Another issue depends on data to generate part of relations in addition to the dependence on primary and foreign keys. To develop “a feasible method is to compare the results returned from relational database with SQL and RDF model with SPARQL which is a powerful new query language for searching RDF data” [2].

Chapter 7

References

- [1] M.Arenas, A.Bertails, E.Prud'hommeaux and J.Sequeda - (2011, September 20), A Direct Mapping of Relational Data to RDF. Available: http://www.w3.org/TR/rdb-direct-mapping/.
- [2] Y. Lv and Z. M. Ma, : "Transformation of Relational Model to RDF Model," In IEEE International Conference on Systems, Man and Cybernetics (SMC 2008).College of Information Science and Engineering, Northeastern University, 2008.
- [3] "A Federated Architecture for Information Management," D. Heimbigner and D. Mcleod. ACM Transaction on Office Information Systems 3(3):253-278 (July 1985).
- [4] E. F. CODD, : "A Relational Model of Data for Large Shared Data Banks," Communication of ACM.College of Information Science and Engineering, IBM Rsearch Laboratory, San Jose, California, 1970.
- [5] Elmasri R. and Navathe S.,"Fundamental of Database Systems", 5th edition, 2007.
- [6] Internationalized Resource Identifiers, http://www.w3.org/International/O-URL-and-ident.html.
- [7] C.Murray, : "Semantic Technologies Developer's Guide,". 2005.
- [8] Dot Net Framework. http://www.microsoft.com/net
- [9] D.Beckett - (2010, February 10), RDF/XML Syntax Specification. Available: http://www.w3.org/TR/REC-rdf-syntax/.
- [10] G.Klyne and J. Carroll - (2004, February 10), Resource Description Framework (RDF): Concepts and Abstract Syntax. Available: http://www.w3.org/TR/rdf-concepts/.