SCHEMA INTEGRATION FOR OBJECT-RELATIONAL DATABASES WITH DATA VERIFICATION

Joseph Fong, Francis Pang, Anthony Fong[+] and Daniel Wong

Computer Science Dept., City University of Hong Kong, Tat Chee Avenue, Hong Kong,
[+]Electronic Engineering Dept., City University of Hong Kong, Tat Chee Avenue, Hong Kong,
Email: {csjfong, csfpang, eefong, csdaniel}@cityu.edu.hk

Abstract

Relational Database System (RDB) has been dominant in the industry for the last two decades. Object Oriented database application (OODB) is recognized as a post-relational technology that can improve productivity. Hence, most companies need to enhance their existing relational database systems to support new Object Oriented applications as and when needed. The trend of the current industrial is to implement an object-relational database system (ORDB) using a relational engine with OO features. This paper proposes a methodology to integrate existing ORDB based on user requirement. Data Exhaustive Search Algorithms (DESA) are used to examine data occurrence after data integration to provide confirmation of the consistence and correctness of the integrated schema. The result is a verified methodology for schema integration for Object Relational Database System.

Keywords: Schema Integration, Object Relational Database, Data Exhaustive Search Algorithms, and Verification

1. Introduction

This paper presents a schema integration methodology to integrate Object Relational Database Systems. The emergent of the Object-Oriented (OO) paradigm in current software industry prompts the application developers to produce new database applications using OO approach. In order to have coherence between new OO database applications and the existing database systems, leading database manufacturers gradually modifying their relational database system to support OO features. It results in so called Object Relational Database Management System (ORDBMS) in the current market. Most of these ORDBMS are power by a relational database engine with extensions to OO interface and features [26]. When designing database using these systems, user employ either relational view with some OO features, or use OO view under a relational core. The objective of this paper is to propose a practitioner approach to integrate this kind of ORDBMS. A simplified schema integration technique [11, 15] is applied to the source database schemas, either in relational or object-oriented structure, based on user requirement. For object schema, the semantic domains are specified by user assertions. [15] After data integration, the Data Exhaustive Search Algorithms are used to verify the semantic relationship between input schemas, the integrated global schema and the user requirement. The resultant is a verified schema of the Object Relational Database System.

2. Related work

[1] described a conceptual basis for schema integration in the sequence of pre-integration, comparison, confirmation and integration. [2] presented a methodology of reverse engineering relational schema to extended entity relationship model (EER). [24] and [18] discussed the problem in the context of distributed and heterogeneous databases. [20] showed how to merge relationship in schema integration [23] classified generic integration and translation tasks based on their operation goals. [19] verified the correctness of schema translation by use of information capacity. [22] performed a survey in distributed database systems. [5] provided a comprehensive survey of various issues in heterogeneous database integration, emphasized on schema mapping and view integration for a distributed multidatabase environment. [4] and [3] described rules of data integration in legacy systems and in federated database systems. [12, 10] developed a methodology for universal database integration for new application, and [17] verified the correctness of integrated schema by use of information capacity. [15] described conflict resolution and integration of Object Schemas using corresponding assertions among classes and attributes. The objective of this paper is to integrate Object-Relational database systems using an universal schema integration rules and capture semantic information of the result schema by an Object Relational metadata.

We apply bottom-up approach to integrate existing database schema into a global schema [11], which will be evolved into a modified global schema. We can integrate the existing database schemas in pairs by discovering their relationship as follows:

1. Compare the database schema,
2. Seek users' confirmation.
3. Conflicts resolutions
4. Merge Entities/Classes and Relationships/Associations
5. Verify the result schema by Data Exhaustive Search Algorithms (DESA) the data occurrences in the database

3. Schema Integration

Schema integration provides a global view of multiple schemas. Our approach uses a bottom up approach to integrate existing database into a global database by pairs. The main objective is to provide an integrated schema based on user requirement with no loss of information. The general algorithm is as follows:

```
Begin For each existing database do
  Begin If its conceptual schema does not exist
      then reconstruct its conceptual schema by
reverse engineering;
      For each pair of existing database schema
A and schema B do
        begin
          Resolve semantic conflicts between
schema A and schema B;
          /*step1*/
          Merge classes/entities and
relationship relations between schema A and B;
          /*step2*/
        end
  end
end
```

The input schemas must analysis in pairs and resolve semantic conflicts in different areas. Conflicts are resolved using well-define semantic rules [11] with user supervisions. Classes/Entities are merged by union or abstractions like subtype, generalization, aggregation and others. To demonstrate this step, EER and OMT diagrams are used to represent the conceptual schema of relational and object oriented respectively. [7-9, 21]

We apply a method which is similar to the assertion-based approach. We first define corresponding assertions for DBA to specify the semantic correspondences among component object-oriented schemas as follows:

1) Class-equivalent – Two classes are class-equivalent if they are semantically equivalent. Their semantic domains are the same. For example class Student in schema A and class Student in schema B are class-equivalent.
2) Class-correspondent – Two classes are class-correspondent if they are semantically related but not equivalent. Three kinds of classes correspondences are identified: class-containment, class-overlap and class-disjointness. For example, class Woman and class Person are class-containment. Class Student and class Teacher are class-overlap. Class Male and Class Female are class-disjointness.
3) Attribute-equivalent – Two attributes are attribute-equivalent if they are semantically equivalent. For example, Student.name and S-name are attribute equivalent.
4) Attribute-set-equivalent – Two attribute sets are attribute-set-equivalent if they are semantically equivalent. For example, attribute set (City, Street, No) and attribute Address are attribute-set-equivalent.
5) Attribute-set-class-equivalent – An attribute set and a class are attribute-set-class-equivalent if the attribute set and the class are semantically equivalent. For example, attribute set (blood-type) and Class Blood are attribute-set-class-equivalent.
6) Attribute-class-set-equivalent – An attribute and a set of classes are attribute-class-set-equivalent if the attribute is semantically equivalent to a division characteristic of a class. For example, attribute Person.sex and a set of classes (Male, Female) are attribute-class-set-equivalent because the division characteristic of class Person for the subclasses is sex.

The details of each of the above steps are demonstrated as follows. (Refer to [11] for detail demonstration in relational schema and [15] for Object schema)

Step 1. Identify and resolve the semantics integrity conflicts among input schemas.

Input: Schema A and B with entities/classes and attributes in conflicts to each other on semantics
Output: Integrated Schema Y after data transformation

In dealing with definition related conflict like inconsistency in keys or synonyms/homonyms in names, user supervision is essential. For

instance, two entities may have some candidate keys overlapping with each other but using different key as the primary key. User has to clarify in this kind of situation.

On the other hand, for conflicts arising from structural differences. The goal here is to capture as much information from the input schemas as possible. The most conservative approach is to capture the superset from the schemas. For example, in dealing with cardinality, the cardinality of the same relationship relation in schema A is 1:1 while the other one in Schema B is 1:n. Since a 1:n relationship is the superset of a 1:1 relationship, the 1:n cardinality is used for the integrated relation. Another example is the participation constraint. If the same relationship relation in different schemas have different level of participation constraints, partial participation always override total participation in the integrated schema. It is because total participation is a subset of partial participation.

When dealing with data type and subtype conflict, association/relationship relation is used for resolution. To illustrate this, assume we have an attribute *Department* of the entity *School* in one schema and an entity *Department* in another schema. To resolve the data type conflict, a 1:n relationship is formed in the integrated schema to link up these two entities.

Step 2. Merge classes/entities and relationship relations
Input: existing schema A and B
Output: merged (integrated) schema X

Classes/Entities are merged using the union operator if their domain is the same. Otherwise, abstractions are used under careful user supervision. By examining the same keys with the same entity name in different database schemas, we can merge the entities by union. The integrated entity takes all the attributes from both entities. Abstractions like generalization; aggregation are used in merging entities/classes in different input schemas when they fulfill the semantic condition. The details are as follows and summarized in Table 1.

Sub-step 2.1 Merge relationship associations by capturing cardinality
IF (class(A_1) = class (B_1)) $\wedge$ (class(A_2) = class (B_2)) $\wedge$ (cardinality(A_1, A_2) = 1:1) $\wedge$
(cardinality(B_1, B_2) = 1:n)
THEN cardinality(A_1, A_2) $\leftarrow$ 1:n;
ELSE IF (class(A_1) = class(B_1)) $\wedge$ (class(A_2) = class(B_2)) $\wedge$ (cardinality(A_1, A_2) = 1:1
or 1:n) $\wedge$ (cardinality(B_1, B_2) = m:n)
THEN cardinality(A_1, A_2) $\leftarrow$ m:n;

Sub-step 2.2 Merge classes/entities by Subtype Relationship
IF domain(A) $\subset$ domain(B)
THEN begin Class(X_1) $\leftarrow$ Class(A)
Class(X_2) $\leftarrow$ Class(B)
Class(X_1) isa Class(X_2)
End;

Sub-step 2.3 Merge entities/classes by Generalization
IF ((domain(A) $\cap$ domain(B)) $\neq$ 0) $\wedge$ ((I(A) $\cap$ I(B))=0)
THEN begin Class(X_1) $\leftarrow$ Class(A)
Class(X_2) $\leftarrow$ Class(B)
Domain(X) $\leftarrow$ domain(A) $\cap$ domain(B)
(I (X_1) $\cap$ I(X_2))=0
end
ELSE IF ((domain(A) $\cap$ domain(B)) $\neq$ 0) $\wedge$ ((I(A) $\cap$ I(B)) $\neq$ 0)
THEN begin Class(X_1) $\leftarrow$ Class(A)
Class(X_2) $\leftarrow$ Class(B)
domain(X) $\leftarrow$ domain(A) $\cap$ domain(B)
(I (X_1) $\cap$ I(X_2)) $\neq$ 0
end;

Sub-step 2.4 Merge classes/entities by Aggregation
Aggregation [25] is an abstraction in which a relationship among objects is represented by a higher level, aggregate object. In relational terminology, aggregation consist of an aggregate entity which is an association of a relationship set with corresponding entities into a single entity set. This aggregate entity is treated as a single unit without concern for the details of its internal structure. [6, 16] In the object-oriented view, aggregation provides a convenient mechanism for modeling the relationship *IS_PART_OF* between objects. [14] By extending the semantics of slot values, an attribute stores either the reference of another object or a copy of that object to make it a composite value. An object becomes dependent upon another if the dependent object is referred by an attribute in the 'parent' object. When an object is deleted, all dependent objects it related to are also deleted. [13] Since the

implementations of this abstraction are different in relational and OO model, the merging procedure are different as well.

Relational View of Aggregation

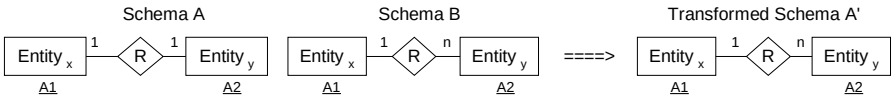
IF relationship(B) $\rightarrow\rightarrow$ class(A) /* MVD $\rightarrow$ means multi-value dependency */
THEN begin aggregation(X_1) $\leftarrow$ (class B_1, relationship B, class B_2)
class(X_2) $\leftarrow$ class(A)
cardinality (X_1, X_2) $\leftarrow$ 1:n
end;

Object Oriented View of Aggregation

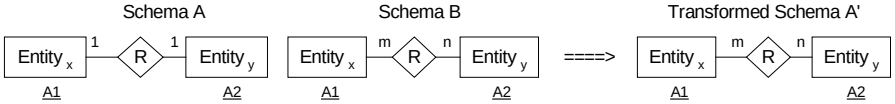
If Domain (Key(B1)) $\subset$ Domain (Attr(A)) AND Domain (Key(B2)) $\subset$ Domain (Attr(A))
THEN begin aggregation(X_1) $\leftarrow$ Class(A)
Class(X_2) $\leftarrow$ Class(B1, association, B2)
End;

Merge relationship associations by capturing cardinality

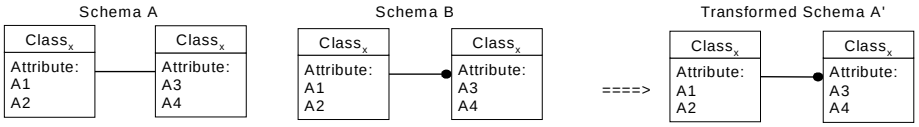
RDB
Case1



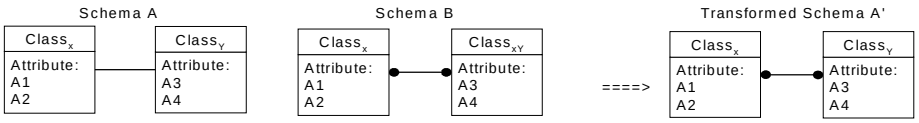
Case2



OODB
Case1

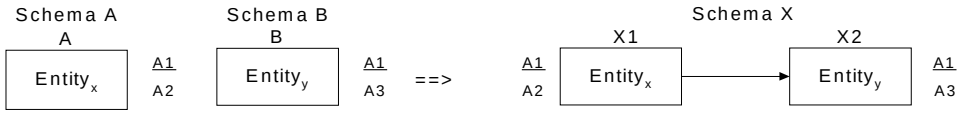


Case2

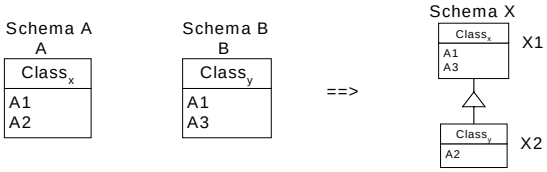


Merge class/entities by subtype relationships

RDB



OODB



Merge classes/entities by generalization

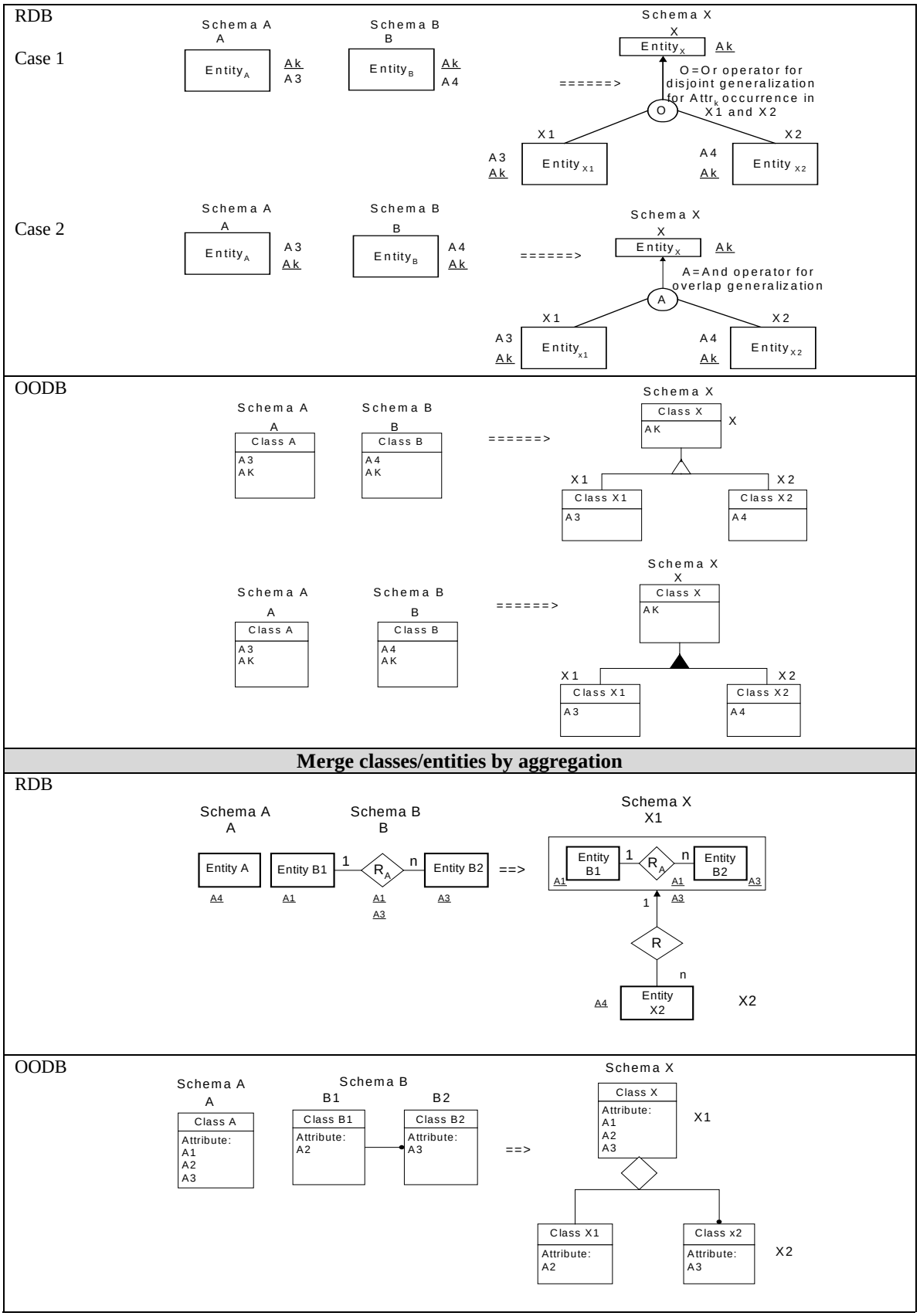


Table 1 Merge classes/entities and relationship relations

4. Verification of the global schema

To discover and verify the abstractions and data semantics in the input schemas and integrated schema. After data integration, data mining algorithms are employed to "mine" these abstractions from the physical data to verify the correctness of the integrate schema based on the user requirement. These algorithms are used to provide verification and confirmation of schema integration by making intelligent guesses from the physical data in finding the abstractions. The intention for using these algorithms assists the user in finding the true data semantics from the integrated schema. The result from the algorithms may not agree with the user-specification in some cases. For example, in extracting the types of generalization (overlap or disjoint) from the data, the physical data may appear as disjoint at some particular database states while the relationship is defined as overlap by the user. In such case, if the user-specification and the result from the algorithm are not contradicted to each other, let user clarify them. However, if the user-specification violate the semantics in the physical data. It will be overruled by the result from the algorithm. Since the input schemas may be in relational view or object oriented view, we provide the algorithms for both views.

1) Verifying algorithm for cardinality

Relational View

Given relations and their primary keys R_1, $PK(R_1)$, ...R_S, $PK(R_S)$ in a relational schema S, we can locate its cardinality as:

```
Select PK(R) from R;
Let i = 1;
While not at end of instance(Pki(R)) do
Begin Select Count(FK(Rj)) = Ci from Rj where
FK(Rj)= Instance(Pki(R));
        Let i = i + 1;
End;
Let minimum(Rj) = minimum(C1,…Cn);
Let maximum(Rj) = maximum(C1,..Cn);
If Minimum(Rj) = 0
Then cardinality (R, Rj) = 1: (0, n)
Else If maximum (Rj) = 1
     Then cardinality (R, Rj) = 1: 1
     Else cardinaliy (R, Rj) = 1:n;
If cardinality (R, Rj) = n:1 and  cardinality (R,
Rh) = n: 1
Then cardinaltiy (Rj, Rh) = m:n
```

Object Oriented View

Given two classes and their reference attributes C_1, $Ref(C_1)$, C_2, $Ref(C_2)$ in an OO schema S, we can locate its cardinality as:

```
For i = 1 to 2 do
begin
Select Ref(Ci), Ci from S;
max(i) = 1
While not end of instance(Ref(Ci)) do
  Begin If instance(Ref(Ci)) = NULL
          then Minimun = True;
  Else If Ref(Ci) is a set reference
            max(i) = n;
  End;
End ;
If Minimum then
     Card(i) = (0, max(i));
Else
     Card(i) = max(i);
End;
End /* For loop */
Let Cardinality (C1, C2) = card(1) : card (2)
```

2) Verifying algorithm for disjoint generalization

Relational View

Given a superclass relation and its primary key: R, PK(R), referring to its subclass relations and their primary key: R_{j1}, $PK(R_{j1})$, …R_{jn}, $PK(R_{jn})$, their generalization can be located as:

```
If ISA-relationship (Rj1, R) = True and … and
ISA-relationship (Rjn, R) = True
Then  Generalization (R, Rj1, …Rjn) := Disjoint;
For h := 1 to n do Select PK(Rjh) from Rjh;
For k := 1 to n do
 begin for m := 1 to n do
    begin if k < m
            then begin
       Select Count(*)=Allcount from PK(Rm)
where PK(Rm) is in PK(Rk);
       If Allcount > 0 then
       Begin
        Generalization (R, Rj1, …, Rjn) :=
Overlap;
         Exit;
       end;
    end;
  end;
end
```

Object Oriented View

Given a superclass and its OID: C, OID(C), referring to its subclass and their OID: C_{j1}, $OID(C_{j1})$, …C_{jn}, $OID(C_{jn})$, their generalization can be located as:

If ISA-relationship (C_{j1}, C) = True and … and ISA-relationship (C_{jn}, C) = True
Then Generalization (C, C_{j1}, …C_{jn}) := Disjoint;
For h := 1 to n do Select OID(C_{jh}) from C_{jh};
For k := 1 to n do
 begin for m := 1 to n do
  begin if k < m
   then begin
  Select Count(*)=Allcount from OID(C_m) where OID(C_m) is in OID(C_k);
  If Allcount > 0
  then begin
   Generalization (C, C_{j1}, …, C_{jn}) := Overlap;
   Exit;
  end;
 end;
 end;
end;

The above algorithm takes *O*(mn) times for two subclasses with n-tuple and m-tuple respectively. For vast amount of data, a more efficient algorithm in identifying disjoint generalization is desirable. One can sort the keys (for relational view) or OID (for OO view) of the two sub-relations/sub-classes using efficient sorting algorithms like radix sort or merge sort. (Which takes *O*(n) and *O*(n log n) respectively) Then merge the two lists together and halt on duplication of data. The halt in the merge procedure indicate overlap generalization. In worst case analysis, (i.e. overlap generalization) the merge only takes *O*(max(m,n)) and the whole process takes maximum *O*(max(m,n)) to complete. (With Radix sort)

3) Verification algorithm for aggregation

Relational View

Given an aggregate relation with its primary keys, AR, PK(AR) referring to its component relations with its foreign key, CR_1,…,CR_n, FK(CR_1),…,FK(CR_n) from schema S, the aggregation can be located as:
Let i = 1;
IF PK(AR) = FK(CR_i)
 then Select FK(CR_i) from S;
 While not at the end of instance(FK(CR_i)) do
  Begin Select Count(FK(CRi)) = C_i from CR_i where Instance(FK(CR_i)) = NULL;
  End;
 Let i = i + 1;
 End;
For i = 1 to n
 IF Ci > 0
 THEN Aggregation (AR, CR_i) = False;
 ELSE Aggregation (AR, CR_i) = True;
 End;
Next;

Object Oriented View

Given an aggregate class with its reference attribute pointers, AC, Ref_1(AC),…,Ref_n(AC) referring to its component classes with its OID, CC_1,…,CC_n, OID(CC_1),…, OID(CC_n) from schema S, the aggregation can be located as:
For i = 1 to n
 For j = 1 to n
  IF Ref_i(AC) = OID(CC_j) THEN
  /* Reference attribute refer to the component class */
   Select Ref_i(AC) from AC;
   While not at the end of instance(Ref_i(AC)) do
    Begin Select Count(Ref_i(AC)) = C_j from AC where Instance(Ref_i(AC)) = NULL;
   End;
   Break;
  End;
 Next;
Next;
For j = 1 to n
 IF C_j > 0
  THEN Aggregation (AR, CC_j) = False;
  ELSE Aggregation (AR, CC_j) = True;
 End;
Next;

5. CONCLUSION

This paper proposes a methodology to integrate existing object-relational database schemas in both relational and object oriented view to facilities different application requirements. The main objective of this methodology is to integrate existing source schemas to fulfill user requirement with no loss of information. A bottom-up schema integration technique is used to integrate existing object-relational schemas. Data Exhaustive Search Algorithms are used to verify the semantic correctness with regards to the user requirement after data integration. The next logical enhancement of this research is to apply the algorithm to different OO modeling standard like the ODMG and UML.

6. REFERENCE

[1] Batini, C., Lenzerini, M. and Navathe, S. (1986) A Comparative Analysis of Methodologies for Database System Integration, ACM Computing Survey, Vol 18, No 4.

[2] Batini, C., Ceri, S. and Navathe, S.(1992) Conceptual Database Design: An Entity-Relationship Approach, The Benjamin/Cummings Publishing Company, Inc, p 227-243.

[3] Boudjlida, N. and Perrin, O. (1995) A Formal Framework and a Procedural Approach for Data Integration. In Proceedings of the International Conference on Systems Integration, ICSI'94, pages 476--485, São Paulo, Brazil, August 1994. IEEE Comp. Society Press.

[4] Boudjlida, N., Bouneffa, M. A., and Perrin, O. (1995) Data Integration in a Legacy-Systems Migration Process. Research Report 95-R-394, CRIN-UHP Nancy 1, January 1995

[5] Elmagarmid, A., Rusinkiewicz, M., Sheth, A. (1999) Management of Heterogeneous and Autonomous Database Systems, Morgan Kaufmann Publishers, Inc.

[6] Elmasri R. and Navathe, S.(2000) Fundamentals of Database Systems, Benjamin/Cummings pub.

[7] Fong, J,(1992) Methodology for schema translation from hierarchical or network into relational, Information and Software Technology, p159-174, Vol 34, No 3, 1992.

[8] Fong, J. and Kwan I. (1994) A Re-engineering Approach for Object-Oriented Database Design Proceeding of First IFIP/SQI International Conference on Software Quality and Productivity pp139-147 Chapman and Hall 1994

[9] Fong, J. (1995) Mapping Extended Entity Relationship Model to Object Modeling Technique. SIGMOD Record 24(3): 18-22 (1995)

[10] Fong J. and Huang, S.M. (1997) Information Systems Reenineering, Springer 1997

[11] Fong, J., Karlapalem, K., Li, Q., and Kwan, I.,(1999) Methodology of Schema Integration for New Database Applications: A Practitioner's Approach, Journal of Database Management, Vol. 10, No. 1, pp. 3-18.

[12] Fong, J. and Huang, S. (1999b) Architecture of a Universal Database: A Frame Model Approach, International Journal of Cooperative Information Systems, Vol. 8, No. 1, pp.47-82.

[13] Gray, P.M.D., Kulkarni, K.G., and Paton, N.W. (1992) Object-Oriented Databases: A Semantic Data Model Approach, Prentice Hall International, UK 1992

[14] Hughes, John G. (1991) Object-Oriented Databases (1991) Prentice Hall International, UK 1991

[15] Koh, J-L., Chen, Arbee (1993) Integration of Heterogeneous Object Schemas, on the Proceedings of 12th International Conference on the ER Approach Arlington, Texas USA, December 1993 Springer-Verlag

[16] Korth, H. and Silberschatz, A.(1997)Database System Concepts (3rd edition), McGraw-Hill.

[17] Kwan, I. and Fong, J.,(1999) Schema integration methodology and its verification by use of information capacity, Information Systems, Vol 24, No 5, pp. 355-376.

[18] McLoed, D. and Heimbigner, D.(1980) A Federated Architecture for Database Systems, Proceedings of the AFIPS National Computer Conference, vol 39, AFIPS Press, Arlington, VA.

[19] R.J.Miller, Y.E.Ioannidis, and R. Ramakrishnan (1993) The use of Information capacity in schema integration and translation, Proceedings of the 19th International Conference on Very Large Data Base, Dublin, Ireland, pp. 120-133, Morgan Kaufmann.

[20] Navathe, S., Sashidhar, T., and Elmasri, R. (1984) Relationship merging in schema integration. Proceedings of the 10th International Conference on Very Large DataBase, Singapore, pp. 27-90, Morgan Kaufmann.

[21] Narasimhan, B., Navathe, S. and Jayaraman, S. (1993) On Mapping ER and Relational Models into OO Schemas, Proceedings of the 12th International Conference on the ER Approach, pp402-413, Springer-Verlag

[22] Özsu, M. and Valduriez, P. (1991) Principles of Distributed Database Systems, Prentice-Hall International Edition, pp.428-430.

[23] Rosenthal, A. and Reiner, D., (1987) Theoretically sound transformations for practical database design. Proceedings of the International Conference on the Entity-Relationship Approach, New York, pp. 115-131.

[24] Sheth A. and Larson, J.(1990)Federated Database Systems for Managing Distributed Heterogeneous, and Autonomous Databases, ACM Computing Survey, Vol 22, No 3.

[25] Smith, J.M., and Smith, D.C.P. (1977) Database Abstractions: Aggregation and Generalization, ACM Trans. On Database Systems 2, No. 2

[26] UNISQL, (1999) http://www.unisql.com/kcompub/new/en_sqlover.htm