

# General Database Infrastructure for Image Retrieval

Carlos Alvez<sup>1</sup>, Aldo Vecchietti<sup>2</sup>

<sup>1</sup> Facultad de Ciencias de la Administración, Universidad Nacional de Entre Ríos  
Concordia, 3200, Argentina

<sup>2</sup> INGAR – UTN, Facultad Regional Santa Fe  
Santa Fe, S3002GJC, Argentina

## Abstract

In this article, a general database infrastructure implemented in an Object-Relational Database Management System for image retrieval is proposed and describe. Semantic and content based image queries can be performed with this application. The infrastructure is structured into three levels: content-based, semantic data and an interface integrating them. It is complemented with a set of database User Defined Types (UDT) composed of attributes and operations. Set operations: union, intersection and difference are implemented for recovering images based on its attributes. In order to prove the capabilities of this approach a case study about vehicles is implemented. The results obtained by performing about 240 queries with a 10000 database image show an important improvement in image similarity search. The architecture can be easily adapted for specific field applications.

**Keywords:** *content – semantic – image retrieval - ORDBMS.*

## 1. Introduction

This paper is an extension and improved version of a previous one [1] where we introduced the software architecture to address image retrieval under an Object-Relational Database Management System (ORDBMS) [2]. Nowadays, the trend in image retrieval is the integration of both low-level and semantic data.

Object-relational database management system (ORDBMS) and its standards (ISO/IEC 9075-1, 2003; ISO/IEC 9075-2, 2003) emerged at the end of the 90's to incorporate the object technology into the relational databases, allowing the treatment of more complex data and relationships than its predecessors. One of these complex applications is Content-Based Image Retrieval (CBIR) which has received a great interest in the past decade. The interest has been driven by the need to efficiently manage and search large volumes of multimedia information, mostly due to the exponential growth of the World-Wide-Web (WWW). CBIR is performed based on abstract descriptions of the images that are extracted during the image analysis phase.

Most of the techniques proposed for CBIR are limited by the semantic gap separating the low level information from its metadata annotations. Semantic gap for image retrieval is the difference between low level data extracted and the interpretation the user has for the same picture [3].

The use of semantic models and techniques generated for the World Wide Web has exponentially grown in the last years for many applications beyond the use for Internet. In order to follow this trend, databases have included this technology to expand the nature of the applications to be performed thorough the DBMS. In this work, Oracle Semantic Technologies included in its 11g ORDBMS version has been employed.

Several approaches can be found in the literature about this topic. RETIN is a Search Engine developed by Gony et al. [4] with the goal of fulfilling the semantic gap. It uses an interactive process that allows the user to refine the query as much as necessary. The interaction with the user consists of binary levels to indicate if a document belongs to a category or not. SemRetriev proposed by Popescu et al. [5] is a prototype system which uses an ontology to structure an image repository in combination with CBIR techniques. The repository includes pictures gathered from Internet. Two methods are employed for image recovery based on keywords and visual similarities where the proposed ontology is used for both cases. Atnafu et al. [6] proposed several similarity-based operators for recovering images stored on relational databases tables. This is one of the first efforts to integrate the content-based and semantic for pictures in a DBMS. Tests were performed extending the prototype called EMIMS. The authors use a database as image repository and added several algorithms to EMIMS.

Research in image retrieval has been handled separately in database management systems and computer vision. In general systems related to image retrieval assume some sort of system containing complex modules for processing images stored in a database.

In this version, several issues regarding the combined low-level and semantic features image retrieval are added. A deeper explanation about the architecture and its implementation is provided such that it can be used as a reference guide for this type of application.

The article is outlined as follows: first the proposed architecture is presented and the levels composing it are explained with more detail. Then a vehicle case study is introduced where the reference ontology used for vehicle classification and MPEG-7 low-level descriptors selected for CBIR are described. After that, the considerations made to include semantic data to images are explained. Finally experiments performed, results obtained and conclusions are presented.

## 2. Architecture for image retrieval

The architecture for semantic and content based image retrieval in an ORDBMS is shown Fig. 1. Three main layers compose this model:

The low level tier is responsible of loading, extracting and management of images visual descriptors (color, texture, etc.). Combinations of user define type (UDT) and java functions and SQL queries are employed in this layer.

The semantic level is composed of database interfaces to introduce semantic ontology concepts and to recovery images related to these concepts in response to SQL semantic queries.

Between the previous layers exists the connection level which is the interface that links both levels. A special infrastructure of UDTs, functions and operators are provided in this section to relate the images low-level descriptors and its semantic concepts.

Fig. 2 shows the UML class diagram representing the collection of database UDTs to support images, matrices, arrays of low level descriptors and semantic information. Classes of Fig. 2 are transformed (via mapping functions) to UDT when defining the ORDBMS schema based on SQL:2003 standard [2].

On top of the hierarchy there is an interface called *SetOperations* where the signature of set operators are defined. Those operators can be used to query low-level or semantic features separately or combined.

*Image* is a class defined to store the image and its properties which inherits the operations declared in *SetOperations*. *Image* has a composition relationship with the generic *LowLevelDescriptor* abstract class from which

special classes to implement specific low-level descriptors can be derived. *Image* attributes are:

- a BLOB (Binary Large Object) to store pictures,
- low-level descriptors UDTs type as a result of mapping the composition relationship with *LowLevelDescriptor*,
- some other attributes to keep images characteristics like size, width, height, etc.

Methods *importImage* and *exportImage* are responsible to import and export images from files respectively. The method *similarScore* implements a particular descriptor which can be used to obtain a distance between a reference image and another one stored in the database; *similar* method in *Image* class calls *similarScore* to compare the descriptor values of a particular image against the remaining in the database. Low-level UDTs are implemented according to the application domain. For example, for general images can be used descriptors proposed by the MPEG-7 standard, for more specific applications like biometry and medicine explicit domain descriptors must be employed. Set operators inherited from the interface can be redefined in *Image* UDT for special purposes. These UDTs and its operators facilitate CBIR queries by using SQL sentences which are familiar in the IT community.

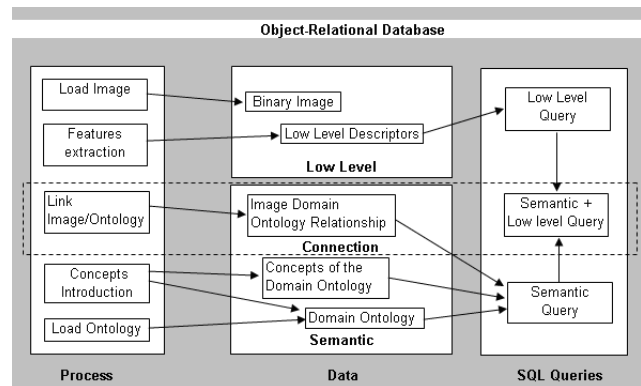


Fig. 1 Architecture proposed for image retrieval.

*SemanticData* class is included to support domain ontology. It inherits methods from interface *SetOperations*. *SemResultSet* is the method to return a set of images with same semantic values.

Semantic data are not loaded automatically and several alternatives are possible to work with them: metadata generated by the user, structured concepts extracted from Internet, ontology driven concepts, etc. Since semantic model must be structured in order to query them by its content, it is convenient to count with a domain ontology like the one proposed by MPEG-7 group [7].

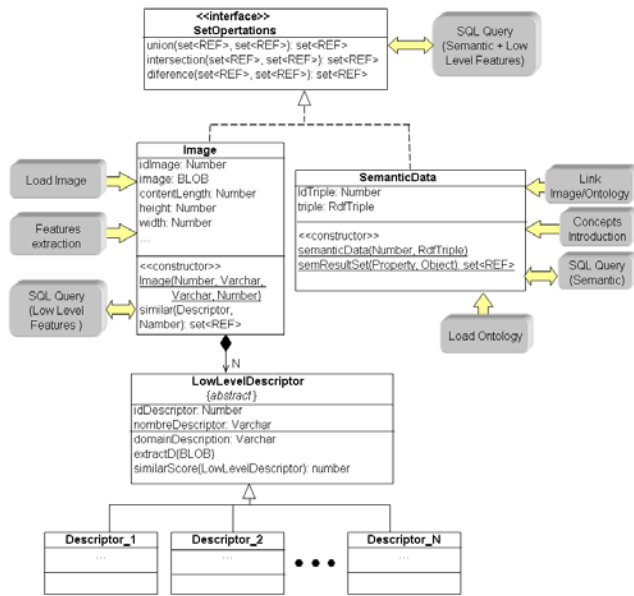


Fig. 2 UML class diagram model of the object-relational database defined for the architecture.

In this work, semantic is introduced for every image in the database. Concepts taken from the domain ontology or literals  $x$  are associated with a specific UDT *Image* instance  $y$  by means of the following triple:

$\langle \text{subject property object} \rangle$

Links can be performed by asserted triples ( $\langle y \text{ rfd:type } x \rangle$ ) or can be inferred by means of some other properties like “subclass of”, “range”, “subproperty of”, etc. [8][9].

The previous levels can be queried individually; in cases that low-level and semantic data are needed together the interface *setOperations* which is inherited by both is used to integrate and get results of combined queries.

*SetOperations* lets the definition of combined union, intersection and difference set operations, which calls *similarScore* and *semResultSet* functions of *LowLevelDescriptor* and *SemanticData* UDT respectively. By these methods it is also possible to get results from two low-level queries or two semantic queries. Both *similarScore* and *semResultSet* functions returns a set of *Image* references (set<REF>). Since both methods return a set<REF> they can be easily employed in SQL queries.

### 3. Model Implementation

The model implementation was made using Oracle 11g database. It was chosen because is one the most advanced

ORDBMS and to get benefit of its Semantic Technologies Tools [10].

#### 3.1 Low Level Implementation

The implementation on this tier implies the creation of the UDTs with its attributes and methods. The low-level metadata for image recovery are represented as a collection of descriptors, precisely the *LowLevelDescriptor* UDT is created (Fig.3 – Part A) where *extractD* and *similarScore* methods are declared. The implementation of these operators for a specific descriptor is defined in the subclasses (see section 4.1).

The composition relationship shown in Fig. 2 between image and *LowLevelDescriptor* is made through a collection, an array for this case because the collection multiplicity can be estimated. In Fig. 3 – Part B, it is shown the *descriptors\_t* UDT creation as an array of *LowLevelDescriptor* and in Fig. 3 – Part C, in the definition of *Image\_t* UDT the *descriptors* attribute is defined as of *descriptors\_t* type.

```
-- Part A: LowLevelDescriptor Definition
-----
create or replace
TYPE LowLevelDescriptor AS OBJECT (
    domain Varchar(32),
    descriptorName Varchar(128),
    Member Procedure extractd(fileName Varchar),
    Member Procedure extractd(img Blob),
    Member Function similarScore(descr LowLevelDescriptor)
    RETURN number
) Not Instantiable not final;

-- Part B: descriptors_t is a array of LowLevelDescriptor
-----
create or replace TYPE descriptors_t
AS VARRAY(10) OF LowLevelDescriptor;

-- Part C: Image_t definition. attributes
-----
create or replace TYPE Image_t AS OBJECT (
    idImage number,
    image BLOB,
    fileName Varchar(255),
    filePath Varchar(255),
    descriptors descriptors_t,
    ...
);
```

Fig. 3 PL/SQL code. Part A: abstract UDT *LowLevelDescriptor* definition. Part B: UDT *descriptors\_t* definition as array of *LowLevelDescriptor*. Part C: UDT *Image\_t* attributes.

Two alternatives are possible to define the descriptors amount and type to be used in a particular application. The first one is by defining the *descriptors* array inside *Image\_t* constructor; and the second one is by defining an additional table to store the descriptors. The first

alternative was chosen for this case (Fig. 4 – Part A and B).

In Fig. 4 – Part A can be seen the signature of the constructor method *Image\_t* where *fileN*, is a parameter which carries out the file name containing the image and *fileP* corresponds to the file path and *dirOb* is an Oracle *Directory* object needed to load the image.

Fig. 4 – Part B shows a code portion of the constructor method *Image\_t*. In this method the descriptors array is created, then its particular instances and finally the arrays are loaded with data. In section 4.1 is shown the implementation of some of those descriptors.

In the constructor method is also loaded the picture into a BLOB attribute. (Fig. 4 – Part C).

```
-- Part A: CONSTRUCTOR FUNCTION Image_t Signature
-----
...
CONSTRUCTOR FUNCTION Image_t(SELF IN OUT NOCOPY Image_t,
                             idImg number, fileN Varchar,
                             fileP Varchar dirOb Varchar)
RETURN SELF AS RESULT,
-----

-- Part B: CONSTRUCTOR FUNCTION in the Image_t's Body
-----
...
-- descriptors array is initialized
descr1 Descriptor1_t;
descr2 Descriptor2_t;
...
--- array of descriptors is created
SELF.descriptor := new descriptor_t();

--- Descriptor extraction
descr1 := new Descriptor1_t(...);
descr1.extractD(fileP);
SELF.descriptor.extend;
SELF.descriptor(1) := descr1;
...
descr2 := new Descriptor2_t(...);
descr2.extractD(fileP);
SELF.descriptor.extend;
SELF.descriptor(2) := descr2;
...

--- Part C: Image Load
-----
DBMS_LOB.CREATETEMPORARY(l_blob, TRUE);
l_bfile := BFILENAME(dirOb, fileN);
DBMS_LOB.fileopen(l_bfile, Dbms_Lob.File_ReadOnly);
DBMS_LOB.loadfromfile(l_blob, l_bfile,
                    DBMS_LOB.getlength(l_bfile));

SELF.Image := l_blob;
...

```

Fig. 4 PL/SQL code. Part A: Constructor *Image\_t* Signature. Part B: Constructor *Image\_t* in *Image\_t*'s body - Descriptors Extraction. Part C: Constructor *Image\_t* in *Image\_t*'s body - Load Image.

The *Image\_t* UDT, has also the *similar* method (Fig. 5 Part A), which is a function that allows the similarity query of one image against the other stored in the database. This method returns a set of references to *Image\_t* objects which are stored in its corresponding Type Table which is created by the following SQL sentence:

```
CREATE TABLE Image OF Image _t;
```

In order to get the image references it is necessary to define *SetRef* UDT as a set of Reference type to *Image\_t*. Because the number of references that can be returned by the *similar* method is not known, the best way to implement *SetRef* UDT is by using a multiset collection which corresponds to nested tables in Oracle 11g (Fig. 5 – Part B). *similar* has two parameters: the descriptor to be used (which is an index inside the descriptors array) and its threshold. For the image comparison all images in the database are invoked by calling the descriptor's *similarScore* method for each one (Fig. 5 – Part C).

```
-- Part A: UDT Image_t. Function similar signature.
-----
create or replace TYPE Image_t AS OBJECT (
...
MEMBER FUNCTION similar(idDesc Number, threshold Number)
RETURN SetRef ,
...
--- Part B: UDT Set Ref.
-----
create or replace type SetRef
as table of ref Image_t;

--- Part C: UDT Image_t. Function similar in Image_t's body.
-----
...
MEMBER FUNCTION similar(idDesc number, threshold number)
RETURN setRef AS
refImg setRef;
BEGIN
SELECT REF(f) BULK COLLECT INTO refImg
FROM image f, image b
WHERE b.idimage= Self.idImage and
b.descriptor(idDesc).similarScore(
f.descriptor(idDesc)) < threshold
ORDER BY b.descriptor(idDesc).similarScore(
f.descriptor(idDesc));
RETURN refImg;
END;
...

```

Fig. 5 PL/SQL code. Part A: Function *similar* Signature. Part B: UDT *SetRef* definition. Part C: Function *similar* *Image\_t* in *Image\_t*'s body.

### 3.2 Semantic Implementation

In order to store and query the semantic information *SemanticData* class (Fig. 2) must be responsible for the following tasks:

- Import, load and store the application domain ontology.
- Load new concepts related to the ontology.
- Establish relationships between images and domain ontology concepts.
- Allow semantic queries to search images in the database.

For *SemanticData* UDT implementations we employed the facilities provided by Oracle Semantic Technologies (OST). From Fig. 6, it can be seen that *SemanticData\_t* UDT has two attributes: *idTriple* which corresponds to a triple identifier and the *triple* attribute which is of *SDO\_RDF\_TRIPLE\_S* type predefined in OST. Given *SemanticData\_t* UDT its corresponding Type Table is created as follows:

```
CREATE TABLE SemanticData OF SemanticData_t;
```

and then a semantic model must be created by means of OST:

```
EXECUTE SEM_APIS.CREATE_RDF_MODEL('nom_model',  
                                'SemanticTable',  
                                'triple');
```

The first parameter corresponds to the model name, the second to the table name where semantic data are stored and the third is the attribute name where the triple must be stored

OST also provides a Java API to import the ontology which was used for this case. For loading the semantic concepts and relate them to images instances we employed a constructor of *SemanticData* UDT, as follows:

```
Insert into SemanticData Values(  
  
SemanticData_t( idTr, SDO_RDF_TRIPLE_S(  
                strIdImg, property, object),  
SemanticData_t( idTr, SDO_RDF_TRIPLE_S(  
                strIdImg, Rdf:type, :Image)  
  
);
```

The first inserted tuple associates the image id with a property and object of the domain ontology. The second inserted tuple indicates that the inserted id is of image type. This is important because when recovering images responding to the following query pattern  $\langle ?I \text{ p } O \rangle$  it must also satisfy the triple  $\langle ?I \text{ p } :Image \rangle$ .

```
-----  
-- UDT SemanticData_t definition  
  
create or replace TYPE SemanticData_t AS OBJECT (  
    idTriple NUMBER,  
    triple SDO_RDF_TRIPLE_S,  
  
    CONSTRUCTOR FUNCTION SemanticData_t(  
        SELF IN OUT NOCOPY SemanticData_t,  
        IdTriple number,  
        triple SDO_RDF_TRIPLE_S)  
        RETURN SELF AS RESULT,  
  
    STATIC FUNCTION semResultSet (tproperty varchar,  
        object varchar)  
        RETURN SetRef,  
    ...
```

Fig. 6 PL/SQL code. UDT *SemanticData\_t* definition.

The static function *semResultSet* is defined to query images using semantic data. This function takes as input parameters a property and an object and returns a set of references to *Image*. To perform this task this function employs the *SEM\_MATCH* method defined by OST.

In the same way than *similar* of *Image\_t*, this function returns an object of type *SetRef* which can be used as an input to union/intersection/difference functions (Fig. 2).

### 3.3 Connection Level Implementation

Having the structure of *Image\_t* and *SemanticData\_t* UDTs, the combined queries integrating both aspects is now very simple to do employing the set operators mentioned before. Anyone of the set operators has the form of:  $Op(SetRef, SetRef): SetRef$ .

Due to *similar* and *semResultSet* methods return a *SetRef* type, then in the set operators any of the following combinations is valid:

```
Op(similar, similar): SetRef  
Op(semResultSet, similar): SetRef  
Op(semResultSet, semResultSet): SetRef
```

The meaning of it is that it is possible to combine not only semantic data queries with low-level data but also low-level queries with different descriptors or semantic queries with different patterns.

Since the operators have been defined as static functions they do not depend on a particular instance. They can be used independently of a particular application.

Fig. 7 shows the implementation of *set\_union* function, which is done in *Image\_t* and also in *SemanticData\_t*, the

other functions *set\_intersect* and *set\_diference* are done in a similar way.

```

STATIC FUNCTION set_union(sr1 SetRef, sr2 SetRef) RETURN SetRef AS
    refImage SetRef ;
BEGIN
    SELECT * BULK COLLECT INTO refImage from
        (SELECT t1.column_value from TABLE(sr1) t1
        UNION
        SELECT t2.column_value from TABLE(sr2) t2) ;
    RETURN refImage ;
END ;
    
```

Fig. 7 PL/SQL code. Set\_union function implementation.

## 4. Case Study

For the case study vehicle images are used. To provide low-level descriptors for those images MPEG-7 standard descriptors: Dominant Color (DCD), Color Layout (CLD), Scalable Color (SCD) and Edge Histogram (EHD) are selected. In the experiment, vehicle images gathered from Flickr by means of Downloadr [11] are stored in the database. Tags and texts are used to obtain pictures from Flickr site. In section 4.1 there is an explanation about low-level descriptors selected and how data are obtained. Fig. 8 presents an excerpt of the vehicle ontology used in this example, images tags and texts have been introduced as concepts in the database like RDF triples. Section 4.2 describes with more details this issue.

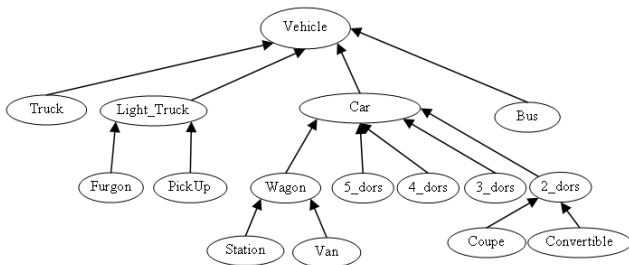


Fig. 8. Excerpt of vehicle reference ontology.

### 4.1 Low Level. Mpeg-7 descriptors

Four UDTs subclasses of *LowLevelDescriptor* are implemented in Oracle to load the following MPEG-7 standard descriptors: Dominant Color (DCD), Color Layout (CLD), Scalable Color (SCD) and Edge Histogram (EHD). *Extract* methods were defined to obtain descriptors data and *similarScore* for content based search.

*similarScore* method was written in PL/SQL, while *extract* is a Java wrapper code linked to LIRE library defined by Caliph&Emir[12]. *Extract* are static methods performing the task of creating objects, obtain descriptors data and store the information in the corresponding arrays defined in the UDTs [13]. For example, *extractionEHD* creates an instance of *EdgeHistogramImplementation* calls method *getStringRepresentation* and returns the 80 bins of this descriptor. The code is compiled and load in the database by means of *Loadjava* utility, by means of the following sentences:

```
Loadjava -u usuario/password JavaWrapper.class
```

In order to use *ExtractionEHD* function in Oracle another wrapper written in PL/SQL is needed for each method in *JavaWrapper.class* as follows:

```

CREATE OR REPLACE FUNCTION
    extractEHD(imagen VARCHAR2)
    RETURN VARCHAR2 AS
    LANGUAGE JAVA
    NAME
        'JavaWrapper.extractionEHD(java.lang.String)
        return java.lang.String';
    
```

Fig. 9 represents the implementation model of low-level data descriptors. In Fig. 10 it is shown a detailed implementation of the UDT *EdgeHistogram\_t* as a subtype of *LowLevelDescriptor*. In Fig. 10 - Part A, *BinCounts* is defined as an array of 80 positions which contains the attribute *binCounts* of *EdgeHistogram\_t* UDT (Fig. 10 - Part B). The description of the body of *EdgeHistogram\_t* and its methods *extractD* and *similarScore* are shown in Fig. 10 - Part C. The first one calls *extractEHD* function defined in the PL/SQL Wrapper previously presented. Function *similarScore* calculates the difference in absolute values of the *binCounts* array between the object calling the function and the array of the input parameter.

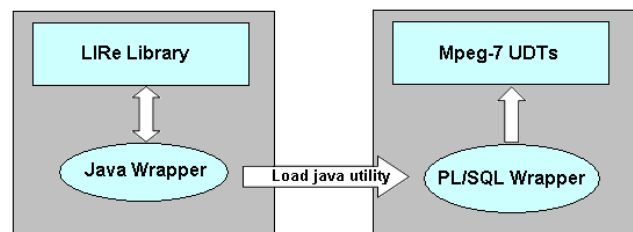


Fig. 9. Low-level description implementation model.

```

-- Part A: BinCounts_t type.
-----
create or replace TYPE BinCounts_t
    AS VARRAY(80) OF NUMBER(1,0);
...

-- Part B: UDT EdgeHistogram_t definition
-----
create or replace TYPE EdgeHistogram_t
    UNDER LowLevelDescriptor (
        binCounts BinCounts_t,
        MEMBER PROCEDURE extractD(img VARCHAR2),
        MEMBER FUNCTION similarScore(eh EdgeHistogram_t)
            RETURN Number
        ...
    );

-- Part C: Part B: UDT EdgeHistogram_t body
-----
create or replace TYPE BODY EdgeHistogram_t AS
    MEMBER PROCEDURE extractD(img VARCHAR2) IS
        binArray ArrayNumber;
    BEGIN
        binArray := new ArrayNumber();
        SELF.BinCounts := new BinCounts_t();
        binArray := Var2ToArray(mmuser.extractEHDStr(img));
        FOR i IN 1 .. 80
            LOOP
                SELF.BinCounts.extend;
                SELF.BinCounts(i) := binArray(i);
            END LOOP;
        END;

    MEMBER FUNCTION similarScore(eh EdgeHistogram_t)
        RETURN Number IS
        sumAbs NUMBER;
    BEGIN
        sumAbs := 0;
        FOR i IN eh.binCounts.FIRST .. eh.binCounts.LAST
            LOOP
                sumAbs := sumAbs + Abs(SELF.binCounts(i)
                    - eh.binCounts(i));
            END LOOP;
        RETURN sumAbs/640;
    END;
    
```

Fig. 10 PL/SQL code. Part A: BinCounts\_t type. Part B: UDT EdgeHistogram\_t definition. Part C: EdgeHistogram\_t body.

Database type table Image is defined to store objects of Image\_t UDT. Each table row contains data items of low-level descriptor UDTs such that once an image is loaded the appropriated *extractD* method is invoked to initialize the objects.

#### 4.2 Semantic. Vehicle model reference

The reference ontology was generated having in mind tags used to download the images from Flickr. It categorizes different vehicle types. By means of Oracle Semantic Technologies a “semantic data network” was created and also a RDF *Vehicle* model. These tasks are completed by implementing *SemanticData* UDT and *semResultSet*

function. Besides the triples of Fig. 8, some other are defined to relate concepts with class instances, domains, etc. Every image loaded in database table *Image* has a link to the reference ontology by adding a triple in the form of  $\langle I \text{ p } O \rangle$ , which associates rows of *Image* table with the hierarchy shown in Fig. 8. Some other functions provided by Oracle Semantic Technologies can be employed after this step, for example *SEM\_APIS.CREATE\_ENTAILMENT* creates a rule index to perform RDFS, OWLPRIME, user rules inference, or *SEM\_MATCH* to query semantic data directly.

#### 4.3 Low-level and Semantic Integration Implementation

Results returned by *semResultSet* y *similarScore* are set of image references that can be combined and manipulated by defining and implementing functions of the interface *SetOperations*. These operations can be overloaded and/or overridden for special purposes giving several options and flexibility to develop an application.

### 5 Experiments and Results

A database of medium size containing 10000 images gathered from Flickr has been generated to perform the analysis, where 30 of them are proposed like descriptors query patterns. For each one between 15 and 20 images visually similar are proponed as the ground truth set (*gts*) [14]. The recovery rate (RR) can be obtained as follows:

$$RR(q) = \frac{NF(\alpha, q)}{NG(q)} \quad (1)$$

where  $NG(q)$  is the *gts* size for query  $q$  and  $NF(\alpha, q)$  is the amount of images of the *gts* obtained between the first  $\alpha NG(q)$  recovered.

$RR(q)$  takes values between 0 and 1, if the value is 0 no images have been found from the *gts* while 1 indicates that all images of the *gts* were recovered. The  $\alpha$  factor must be  $\geq 1$ , tolerance is larger for higher  $\alpha$  and therefore it increases the likely of obtaining images from the *gts*. For the experiment we set  $\alpha = 1.6$  y  $NG(q) = 5$  meaning that  $NF(\alpha, q)$  is the amount of *gts* images recovered from the first 8. Figure 11 presents an example and the *gts* proposed.

The performance of selected descriptors for all queries ( $NQ=30$  for our study) is evaluated by the average recovery rate (ARR) [14] given by:

$$ARR = \frac{1}{NQ} \sum_{q=1}^{NQ} RR(q) \quad (2)$$













		$NQ$			
		$q_1$	$q_2$	$q_3$	...
ground-truth set (gts)	$q_i$				...
				...	
				...	
				...	
	...	...	...	...	

Fig. 11. Examples of ground truth set.

Different levels of the hierarchy tree of Fig. 8 are selected as query objectives in order to compare the efficiency and precision of the answers. Results obtained taking into account with and without the semantic models are shown in Fig. 12. For example for two doors sedan cars the comparison is made considering the whole set of car images against 2\_doors cars set. The inference made by using specified domain and property *rdfs:subClassOf* can contain any of these values: 2\_door, convertible, coupé. Images qualified by 2\_doors concept are visited and also those belonging to *coupe* and *convertible*.

The experiment is performed as follows:

- One query for every image in the database using each one of the descriptors presented in section 3.1.
- One query for every image satisfying the condition of the tree node selected in a direct or inferred manner.

In total 240 queries have been executed.

In Fig. 12 can be found the average recovered rate obtained for each one of the descriptors selected for this experiment, results without and including the semantic are

shown in the first and second series of Fig. 12, respectively.

From Fig. 12 can be concluded that SCD descriptor has the best behavior when recovering images by similarity. It can be also observed that there is an important improvement getting similar images when semantic inference is employed together low level descriptors data. In Figure 13 it can be seen the images obtained with queries involving SCD descriptor integrated with semantic data.

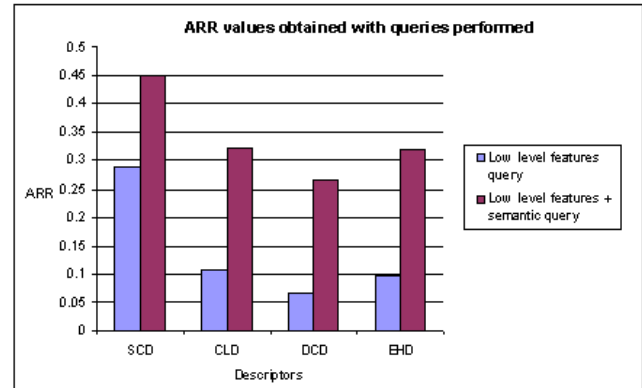


Fig. 12. ARR values obtained with queries performed.

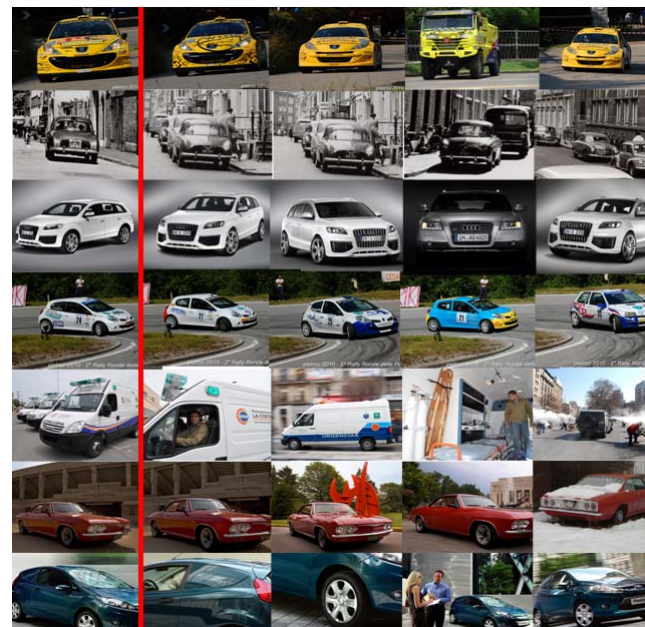


Fig. 13. Results obtained with SCD descriptor. The left most image of each row shows the query image, the other four on the right show the most relevant retrieved.



## 6 Conclusions

Architecture for image retrieval in an Object-Relational Database Management System is proposed and implemented in Oracle 11g database by using its semantic technologies framework. The architecture is structured in three levels, one for the semantic content of the images, another one for the low-level content and the third tier is the connection between the previous. For each layer, UDTs containing attributes and operator functions for images are proposed and defined. These UDTs are the fundamental infrastructure in order to support images, its characteristics and perform queries based on its low-level contents and semantic information. A detailed explanation about the implementation is made through this article to provide and insight about the way to carry out this application type.

The architecture functionality has been studied by means of a case study based on a vehicle ontology for semantic annotations and MPEG 7 low-level descriptors for low-level contents. Descriptors employed are: Dominant Color (DCD), Color Layout (CLD), Scalable Color (SCD) and Edge Histogram (EHD). Extraction and loading data about those descriptors is made via Java and PL/SQL wrappers linked to LIRe library. Semantic data is included by means of RDF triples using Oracle tools. Ground truth sets of images and the average recovery rate (ARR) property are defined to drive the experiments. Semantic data of images have been loaded in the database using reference ontology concepts and properties. A total of 240 queries have been executed. Results obtained from the scheme proposed demonstrate an important improvement in images similarity search.

This example belongs to a broad image domain, having an unlimited and unpredictable variability of the image's content. On the other hand, a narrow image domain can be easily implemented because the architecture proposed is generic such that it can be adapted to specific field applications.

## Acknowledgments

Authors thanks to Universidad Tecnológica Nacional (UTN) and Universidad Nacional Entre Ríos (UNER) for the financial support received.

## References

[1] Carlos E. Alvez, Aldo R. Vecchiatti. Combining Semantic and Content Based Image Retrieval in ORDBMS. Knowledge-Based and Intelligent Information and Engineering Systems Lecture Notes in Computer Science,

- 2010, Volume 6277/2010, 44-53. Editors Rossitza Setchi, Ivan Jordanov, Robert J. Howlett, Lakhmi C. Jain. Springer-Verlag Berlin Heidelberg (2010).
- [2] Melton Jim, "(ISO-ANSI Working Draft) Foundation (SQL/Foundation)", ISO/IEC 9075-2:2003 (E), United States of America (ANSI), 2003.
- [3] Neumamm D. and Gegenfurtner K. "Image Retrieval and Perceptual Similarity" ACM Transactions on Applied Perception, Vol. 3, No. 1, January 2006, Pages 31-47.
- [4] Gony J., Cord M., Philipp-Foliguet S. and Philippe H. "RETIN: a Smart Interactive Digital Media Retrieval System". ACM Sixth International Conference on Image and Video Retrieval - CIVR 2007 - July 9-11, 2007, Amsterdam, The Netherlands, pp. 93-96, (2007).
- [5] Popescu A., Moellic P.A. and Millet C. "SemRetriev - an Ontology Driven Image Retrieval System". ACM Sixth International Conference on Image and Video Retrieval - CIVR 2007 - July 9-11, Amsterdam, The Netherlands, pp. 113-116, (2007).
- [6] Atnafu S., Chbeir R., Coquil D. and Brunie L. "Integrating Similarity-Based Queries in Image DBMSs". 2004 ACM Symposium on Applied Computing, March 14-17, 2004, Nicosia, Cyprus, pp. 735-739 (2004).
- [7] J. Hunter, "Adding Multimedia to the Semantic Web - Building and Applying an MPEG-7 Ontology", chapter of "Multimedia Content and the Semantic Web: Methods, Standards and Tools", Wiley, 2006.
- [8] Dan Brickley and R.V. Guha. "RDF Vocabulary Description Language 1.0: RDF Schema". W3C Recommendation 10 February 2004, <http://www.w3.org/TR/rdf-schema/>.
- [9] Allemang D. and Hendler J., "Semantic Web for the working Ontologist. Effective Modeling in RDFS and OWL". Morgan Kaufman(2008).
- [10] Chuck Murray, Oracle Database Semantic Technologies Developer's Guide. 11g Release 1 (11.1) Part B28397-02. September 2007.
- [11] Flickr Downloadr 2.0.4, <http://flickrdownloadr.codeplex.com>.
- [12] Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval - An Extensible Java CBIR Library. In proceedings of the 16th ACM International Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008.
- [13] C. Alvez, A. Vecchiatti, "A model for similarity image search based on object-relational database", IV Congresso da Academia Trinacional de Ciências, 7 a 9 de Outubro de 2009 - Foz do Iguaçu - Paraná / Brasil (2009).
- [14] Manjunath B., Ohm J. R., Vasudevan V. and A. Yamada. "Color and texture descriptors". IEEE Trans. on Circuits and Systems for Video Technology, vol. 11, no. 6, 703 -715, June 2001.

**C. Alvez** obtained his M.Sc degree in computer science in 2003, Departamento de Ciencias e Ingeniería de la Computación - Universidad Nacional del Sur. Bahía Blanca, Argentina. Professor and researcher at the College of Administration Sciences, UNER.

**A Vecchiatti.** Obtained his PhD degree in 2000 in Chemical Engineering at Facultad de Ingeniería Química Universidad Nacional del Litoral, Argentina. He is a researcher of CONICET and Professor at Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Argentina.