MIAOW

Marble Industry Advertising Over the World

MIAOW Multimedia Database: Revised Design and Implementation

Doc. No:	MIAOW-CNR-DES-001-007		
Issue :	1	Date:	02/12/1996
Revision:	1 --	Date:	02/12/1996
Author:	IEI - CNR		
Technical Board:		Date:	
Quality Repr.:		Date:	

TABLE OF CONTENTS

1. Introduction

This document contains a revised version of the design and implementation phases of the MIAOW database described in the Report "Design Document of the MIAOW Multimedia Database" [MIAOW-DD].

This review is a consequence of a better understanding of requirements and of a critical evaluation of the E-R approach applied to the design of an Illustra database.

This enhanced understanding of requirements has led to an enrichment of the conceptual schema in order to include information that previously had not been considered significant.

For the review, of both the conceptual and logical design, the Object Modeling Technique (OMT) [Rum91] was employed. This methodology was chosen to overcome the intrinsic restrictions of the entity-relationship modeling approach [Che76] used previously, which did not allow full exploitation of the Illustra modeling capabilities.

Using the OMT methodology has allowed the use of OMT mechanisms, like aggregation, that are not available in the entity-relationship model.

The schema, obtained after the revised database design phase, was implemented using the Illustra structures.

The remaining sections of this report are organized as follows:

Section 2 describes the modifications of requirements, which have been considered subsequently in the design phases;

Section 3 shows the MIAOW database revised conceptual schema;

Section 4 presents the revised logical design and introduces the mapping between OMT and Illustra objects models;

Section 5 describes the implementation phase of the Illustra schema.

Finally, Appendix A contains a brief description of the OMT Object Model and Appendix B presents the MIAOW database non-functional requirements on updates and queries.

2. Revising schema design

During the second semester of the MIAOW project a revision of the previous work was carried out as a result of a better comprehension of both the requirements and the selected software environment. This revision had two major effects on the previous design: i) the conceptual schema was enriched to incorporate new requirements; ii) an object representation of the schemas produced during the design replaced the previous entity-relationship to fully exploit the modeling capabilities of the object-relational DBMS Illustra. The next two sections deal with these two points in detail.

2.1 Enhancing requirements

The better understanding of requirements led to an enrichment of the conceptual schema.

In particular, through a deeper analysis of the requirements the necessity emerged to maintain, for each material, also the available formats for sale, the shipping constraints and the quantity of material available from each supplier, and the total production per month of a given material. Moreover, it was necessary, for each material, to know not only whether it was suitable for interior or exterior, but also whether it was suitable for ground, wall and whether it is pedestrian and suitable for vehicles (heavy or light traffic). Finally for each material, in addition to the indicative price, the lowest and the highest price were required. A currency was associated with each of these three prices.

Information about currency was also introduced, like the name of the moneys in circulation, the exchange factor and a description of the money. This information was inserted to give users of the MIAOW database to know the prices of materials in the preferred currency.

Additional information about the suppliers, such as the address, phone and fax numbers, e-mail and a presentation, with a photo of the society, were also judged to be useful.

Moreover, more detailed information about the quarry was required: its name, address, phone and fax numbers, e-mail and the area in which the quarry is placed, with a map of this area.

The only information, among those modeled previously, that was considered unnecessary after having made a review of the requirements, were the names of the

nearest towns to the quarry, since this information was deemed to be too detailed to be meaningful for the MIAOW user.

Going deeper into the knowledge of the domain of the marbles and the stones used in building, it was realized that images of materials stored in the MIAOW database need not represent materials carved in different ways and rough materials instead, for each material, the images still represent the same photograph, only with a different definition. As a consequence, the requirement of storing five images of each material was replaced by that of storing four images, with an associated description of them. From one of these images, then, a predominant colour hue of the material is calculated and stored in RGB components.

Finally, an important improvement carried out was to display a "show room" made by a collection of photos and descriptions of works realized with the materials provided by a given supplier.

2.2 Object representation

As illustrated in the MIAOW Report on Telematic Technologies [MIAOW-TT], the Illustra database management system offers modeling capabilities of both relational and object models, i.e. it adds to the relational modeling features the possibility of representing object identity, composite objects, multivalue attributes, etc..

As a result of a better experimentation of the Illustra system, we realized that the ER modeling approach, suggested by some authors as the best for supporting the design of Illustra schemas, if not used carefully, could lead the designer to produce a schema in a form that does not fully exploit the Illustra modeling capabilities. For example, the ER model does not allow the representation of an object as a composite element made of subparts, whereas Illustra supports this facility. Thus, if we represent the requirements using an ER approach we lose immediately, at the conceptual level, information which can instead be usefully exploited at the lower level of detail to generate a more efficient object storing structuring.

In order to overcome this problem during the second phase of the MIAOW project it was decided to reconstruct the schema design using an object model. The model employed for this new version is the one introduced within the Object Modeling Technique (OMT) [Rum91]. This model, described briefly in Appendix A, is actually a sort of extension of the ER model, with abstraction mechanisms like aggregation and inheritance. In building the new version of the design, therefore, we have replaced the previous design, applying the new available mechanisms when appropriate.

3. Revised conceptual schema

A graphical representation of the revised MIAOW conceptual schema, given in terms of the OMT Object Model graphical notation, follows[1]. As OMT allows one to decompose the specification at different levels of detail, the schema is shown on two levels: Level 1 shows the main objects of the schema; whereas Level 2 represents the subparts of the main objects. Thus, the MIAOW conceptual schema is presented as decomposed into two parts. Figure 3.1 describes the main MIAOW database objects. Figure 3.2[2] shows the explicit constraints of the main objects (namely, the constraints that cannot be represented explicitly with the OMT Object Model mechanisms). Figure 3.3 shows the object "material" as an aggregate of parts. Each part is an object which represents a particular characteristic of "material". Explicit constraints of it are shown in Figure 3.4.

As the OMT Object Model does not provide any notation to represent the explicit constraints, a semiformal notation has been used in the pictures showing the explicit schema constraints.

For simplicity, we have not listed the objects' methods. We assume implicitly, however, each object to have methods for the creation, deletion and modification of its instances. These methods are defined in such a way as to always preserve the given constraints.

In what follows we suppose that the following types are given:

1. basic types:
    - *character*
    - *desc_t*
    - *image_t*
    - *integer*
    - *real*
    - *string*

[1] See Appendix A for a brief description of this notation.
[2] In this figure, and the ones that follow, the dot notation is used: oo.aa indicates the attribute 'aa' of the object 'oo'.

2. enumerated types:
   - *background_tt* ≡ {'Uniform', 'Crystalline', 'Clouded', 'Fine Grain', 'Gross Grain', 'Fine & Gross Grain', 'Salt & Pepper', 'Spotting'}
   - *finish_tt* ≡ {'Polished', 'Flamed', 'Sand Blasy', 'Hammered', 'Split Face', 'Jet Liquid Honed', 'Honed', 'Acid Finishing'}
   - *format_tt* ≡ {'Blocks', 'Slabs', 'Tiles', 'Cut to Size', 'Base or Skirt', 'Artwork'}
   - *material_type_tt* ≡ {'Marble', 'Granite', 'Stone', 'Quartzite', 'Travertine'}
   - *quality_tt* ≡ {'Limestone', 'Sandstone', 'Slate'}
   - *use_mode_tt* ≡ {'Interior', 'Exterior', 'Ground', 'Wall', 'Pedestrian', 'Carriage-heavy-traffic', 'Carriage-light-traffic'}
   - *vein_pattern_tt* ≡ {'Rust', 'Spot', 'Parallel to the bed', 'Against the bed', 'Flowery', '45° Vein', 'Clouded', 'Streak', 'Spotting', 'Arabesque'}
   - *vein_type_tt* ≡ {'No Vein', 'Light Vein', 'Regular Vein', 'Strong Vein', 'Irregular Vein'}

3. composed types:
   - *address_t* ≡ street: string,
     number: string,
     country: string,
     city: string
     zipcode: integer,
     state: string
   - array[integer][*type_t*] (array of elements of type *type_t* of fixed length)
   - *price_t* ≡ price: real,
     currency: array[3][*character*]
   - *profile_t* ≡ description: descr_t,
     society_image: image_t
   - set[*type_t*] (set of elements of type *type_t*)

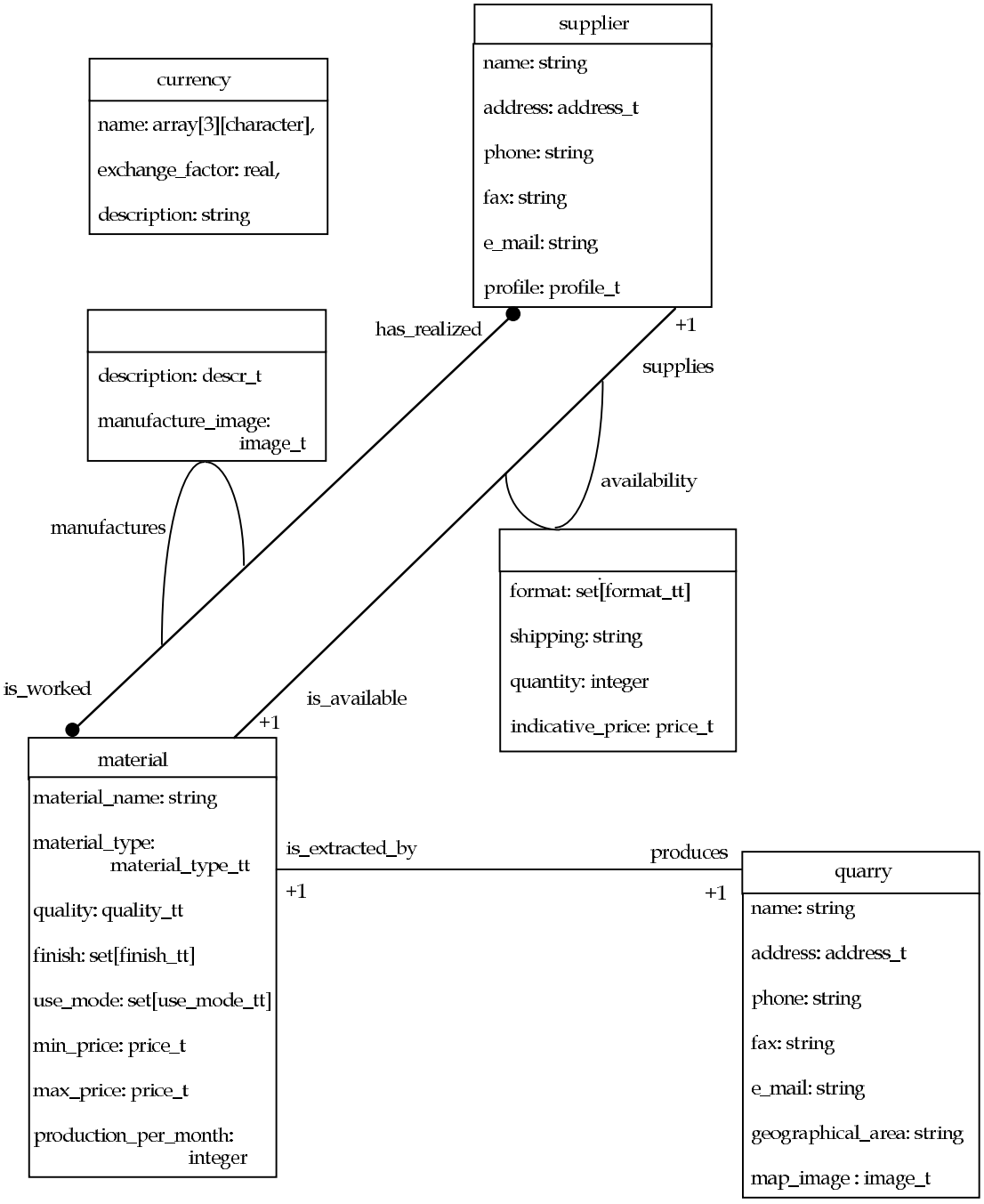

Figure 3.1.- *Conceptual Schema: Level 1 Object Model*

availability.quantity > 0

availability.indicative_price.price > 0

availability.indicative_price.currency appears in currency.name

(availability.supplies, availability.is_available) is unique

currency.name is the primary key of currency

currency.name is not null

currency.exchange_factor > 0

manufactures.manufacture_image is not null

manufactures.description is not null

material.material_name is the primary key of material

material.material_name is not null

material.quality not null if material.material_type = 'stone'

material.min_price.price > 0

material.min_price.currency appears in currency.name

material.max_price.price > 0

material.max_price.currency appears in currency.name

material.production_per_month > 0

quarry.name is the primary key of quarry

quarry.name is not null

supplier.name is the primary key of supplier

supplier.name is not null

supplier.address is not null

Figure 3.2.- *Conceptual Schema: Level 1 Object Model - Explicit Constraints*

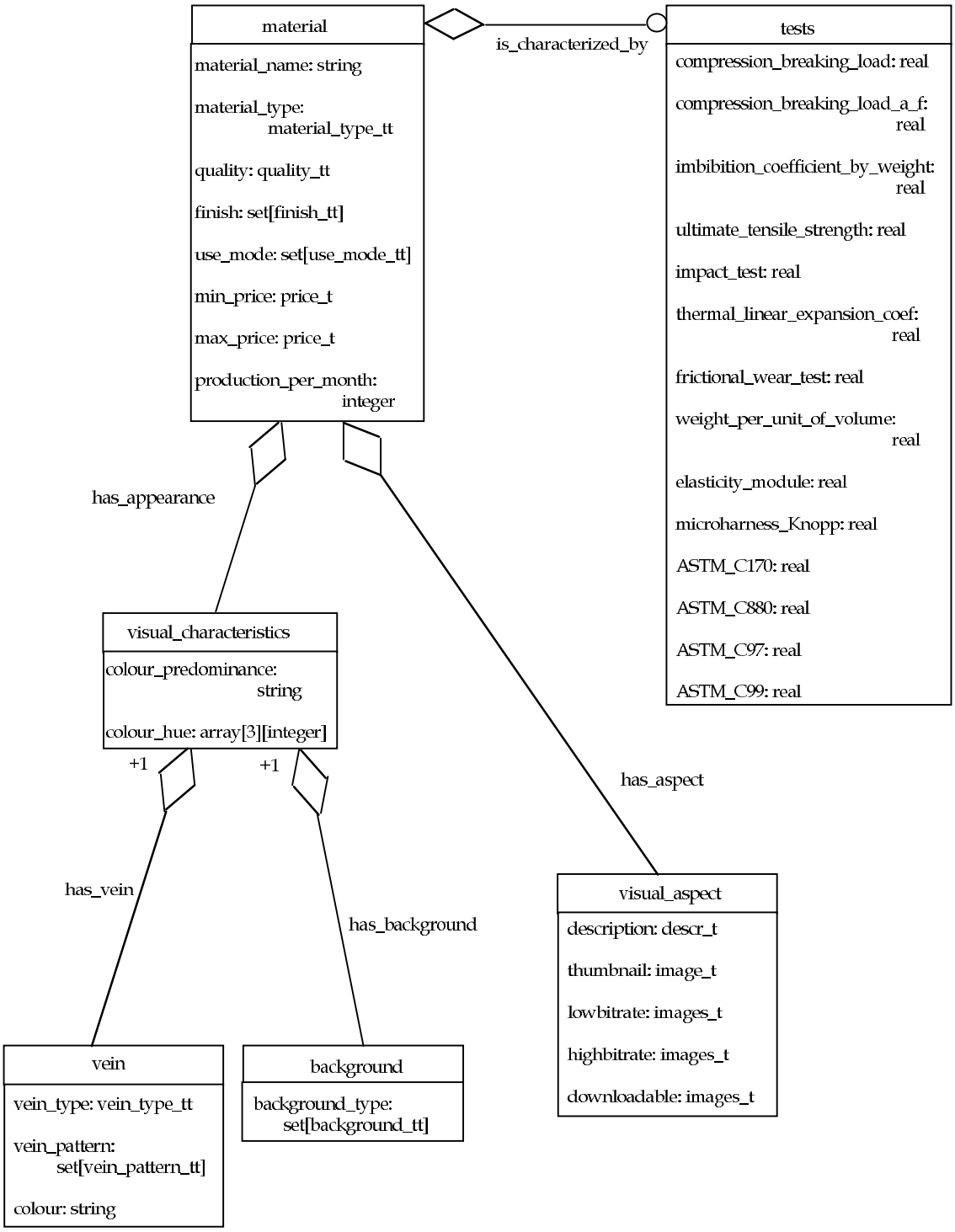

Figure 3.3.- *Conceptual Schema: Level 2 Object Model - object material*

background.background_type is unique

background.background_type is not null

material.material_name is the primary key of material

material.material_name is not null

material.quality not null if material.material_type = 'stone'

material.min_price.price > 0

material.min_price.currency appears in currency.name

material.max_price.price > 0

material.max_price.currency appears in currency.name

material.production_per_month > 0

tests.compression_breaking_load ∈ [0,..,9999.99]

tests.compression_breaking_load_a_f ∈ [0,..,9999.99]

tests.imbibition_coefficient_by_weight ∈ [0.01,..,9.999]

tests.ultimate_tensile_strength ∈ [0,..,999.99]

tests.impact_test ∈ [0,..,99.99]

tests.thermal_linear_expansion_coef ∈ [0.0001,..,9.9999]

tests.frictional_wear_test ∈ [0.01,..,9.99]

tests.weight_per_unit_of_volume ∈ [0,..,9999.99]

tests.elasticity_module ∈ [0,..,999.99]

tests.microharness_Knopp ∈ [0,..,999]

tests.ASTM_C170 ∈ [0,..,99999.99]

tests.ASTM_C880 ∈ [0,..,9999.99].

tests.ASTM_C97 ∈ [0.01,..,9.99]

tests.ASTM_C99 ∈ [0,..,9999.99]

vein.vein_type is not null

vein.vein_pattern is not null

(vein.vein_type, vein.vein_pattern, vein.colour) is unique

visual_characteristics.colour_hue[1] ∈ [0,..,255]

visual_characteristics.colour_hue[2] ∈ [0,..,255]

visual_characteristics.colour_hue[3] ∈ [0,..,255]

Figure 3.4.- *Conceptual Schema: Level 2 Object Model - Explicit Constraints of object material*

In order to better understand the link between the requirements and the given above conceptual schema, two tables are presented: the first (pp. 12 to 18) describes, for each object of the conceptual schema, what the object wants to model; the second (pp. 19 to 25) describes the integrity constraints in a more detailed way.

Name	***Description***
Availability	It is a many-to-many relationship which links the materials to their suppliers. Each instance of "availability" models the availability of a supplier S for a given material M.
Attributes	
format	It describes the material M formats for sale of the supplier S. It takes as value a subset of {'Blocks', 'Slabs', 'Tiles', 'Cut to Size', 'Base or Skirt', 'Artwork'}.
shipping	It describes the shipping constraints that the supplier S has for the material M.
quantity	It describes the quantity of the material M that a supplier S has available.
indicative_price	It denotes the indicative price at which the material M is sold by the supplier S. It also indicates the currency used in the specification of the indicative price. The currency can take only the values contained in the field "name" of the object "currency".
Relationships	
is_characterized_by	It describes the aggregation relationship between an instance of "availability" and an instance of "tests".
supplies	It describes the role of "supplier" in the association relationship "availability" between material M and supplier S.
is_available	It defines the role of "material" in the association relationship "availability" between supplier S and material M.

Name	***Description***
Background	It models all the possible materials background types.
Attributes	
background_type	It represents the background types that the materials can have. The possible values are 'Uniform', 'Crystalline', 'Clouded', 'Fine Grain', 'Gross Grain', 'Fine & Gross Grain', 'Salt & Pepper', 'Spotting'.
Relationships	
has_background	It defines the association relationship between "visual_characteristics" of the material and its background.
Currency	It models the currency by which the prices of materials can be given.
Attributes	
name	It contains the name of the currency.
exchange_factor	It contains the exchange factor of the currency with respect to the US dollar.
description	It describes the currency.
Manufactures	It is a many-to-many relationship which links the materials to their suppliers. Each instance of "manufactures" corresponds to a work made of the material M provided by the supplier S.
Attributes	
manufacture_image	It stores an image of a work made of the material M provided by the supplier S.
description	It contains a description of the work stored in "manufacture_image".
Relationships	
has_realized	It defines the "supplier" role in the association relationship "manufactures" between material M and supplier S.
is_worked	It defines the "material" role in the association relationship "manufactures" between supplier S and material M.

Name	*Description*
Material	It models the materials which will be stored in the MIAOW database.
Attributes	
material_name	It contains the name of the material.
material_type	It contains the type of material. It takes one of the following values: 'Marble', 'Granite', 'Stone', 'Quartzite' and 'Travertine'.
quality	It describes the type of stone. It can take one of the following values: 'Limestone', 'Sandstone' and 'Slate'. It is defined only if the material_type is 'Stone'.
finish	It indicates the possible material workings. It can take as a value a subset of {'Polished', 'Flamed', 'Sand Blasy', 'Hammered', 'Split Face', 'Jet Liquid Honed', 'Honed', 'Acid Finishing'}.
use_mode	It shows the uses that can be made of the material. It can take as a value a subset of {'Interior', 'Exterior', 'Ground', 'Wall', 'Pedestrian', 'Carriage-heavy-traffic', 'Carriage-light-traffic'}.
min_price	It models the lowest price at which the material is sold and its currency. The currency can take only the values contained in the field "name" of the class "currency".
max_price	It contains the top price at which the material is and its currency. The currency can take only the values contained in the field "name" of the class "currency".
production_per_ month	It represents the material total production per month.
Relationships	
is_worked	It models the role of "material" in the association relationship "manufactures" between "material" and "supplier".
is_available	It represents the role of "material" in the association relationship "availability" between "material" and "supplier".
is_extracted_by	It represents the association relationship between each material and the quarry from which it is extracted.
has_apparence	It represents the aggregation relationship between "material" and its "visual_characteristics".
has_aspect	It represents the aggregation relationship between each material and its "visual_aspect".
is_characterized_by	It describes the aggregation relationship between an instance of "material" and an instance of "tests".

Name	*Description*
Quarry	It models the quarries where the materials are extracted.
Attributes	
name	It contains the name of the quarry.
address	It contains the address of the quarry.
phone	It contains the telephone number of the quarry.
fax	It contains the fax number of the quarry.
e_mail	It contains the e-mail of the quarry.
geographical_area	It defines the geographical area where the quarry is located.
map_image	It contains a photo of the area where the quarry is located.
Relationships	
produces	It represents the association relationship between quarries and materials extracted from them.
Supplier	It models the suppliers of the materials stored in the MIAOW database.
Attributes	
name	It contains the name of the supplier.
address	It contains the address of the supplier.
phone	It contains the telephone number of the supplier.
fax	It contains the fax number of the supplier.
e_mail	It contains the e-mail of the supplier.
profile	It provides a presentation and a photo of the society.
Relationships	
has_realized	It models the role of “supplier” in the association relationship “manufactures” between “supplier” and “material”.
supplies	It models the role of “supplier” in the association relationship “availability” between suppliers and materials that they provide.

Name	***Description***
Tests	It represents the possible tests to which materials can be submitted.
Attributes	
compression_breaking_load	It contains the values for the compression breaking load test in Kg/cm2. It takes values between 0 and 9999.99.
compression_breaking_load_a_f	It contains the values for the compression breaking load after freezing test in Kg/cm2. It takes values between 0 and 9999.99.
imbibition_coefficient_by_weight	It contains the values for the imbibition coefficient by weight test. It takes values between 0.001% and 9.999%.
ultimate_tensile_strength	It contains the values for the ultimate tensile strength test in Kg/cm2. It takes values between 0 and 999.99.
impact_test	It contains the values for the impact test in cm. It takes values between 0 and 99.99.
thermal_linear_expansion_coef	It contains the values for the thermal linear expansion coef test in mm/mC°. It takes values between 0.0001 and 9.9999.
frictional_wear_test	It contains the values for the frictional wear test (relative abrasion coefficient) in mm. It takes values between 0.01 and 9.99.
weight_per_unit_of_volume	It contains the values for the weight per unit of volume test in Kg/m3. It takes values between 0 and 9999.99.
elasticity_module	It contains the values for the elasticity module test in Kg/cm2. It takes values between 0 and 999.999.
microhardness_Knopp	It contains the values for the microhardness Knopp test in Kg/mm2. It takes values between 0 and 999.
ASTM_C170	It contains the values for the compression strength test. It takes values between 0 and 99999.99.
ASTM_C880	It contains the values for the flexural strength test. It takes values between 0 and 9999.99.
ASTM_C97	It contains the values for the water absorption test in percentage by weight. It takes values between 0.01 and 9.99.
ASTM_C99	It contains the values for the modules of rupture test. It takes values between 0 and 9999.99.
Relationships	
is_characterized_by	It models the aggregation relationship between each instance of “material” and its technical tests.

Name	*Description*
Vein	It models the possible veins of materials.
Attributes	
vein_type	It contains the vein types of materials. It takes one of the following values: 'No Vein', 'Light Vein', 'Regular Vein', 'Strong Vein', 'Irregular Vein'.
vein_pattern	It contains the vein characteristics of materials. It takes a subset of {'Rust', 'Spot', 'Parallel to the bed', 'Against the bed', 'Flowery', '45 Vein', 'Clouded', 'Streak', 'Spotting', 'Arabesque'}.
colour	It represents the colour predominance of the material vein.
Relationships	
has_vein	It models the aggregation relationship between “vein” and the “visual_characteristics” of the materials that have that particular vein.
Visual_aspect	It contains four material photos of different definitions.
Attributes	
description	It is a description of the photo that is represented with different definitions in the four attributes that follow.
thumbnail	It represents a photo of the material in GIF 48X48.
lowbitrate	It represents a photo of the material in JPEG 250X188.
highbitrate	It represents a photo of the material in JPEG 400X300.
downloadable	It represents a photo of the material in TGA 400X300.
Relationships	
has_aspect	It models the aggregation relationship between “visual_aspect” and the material that is represented in the photo.

Name	***Description***
Visual_ characteristics	It contains information about the colour of the materials.
Attributes	
colour_predominance	It represents the colour predominance of the material.
colour_hue	It is the colour predominance of the material computed from a photo of it. It has three components: each of them is an integer between 0 and 255.
Relationships	
has_appearance	It represents the aggregation relationship between each instance of "material" and its colour characteristics.
has_vein	It models the aggregation relationship between each instance of "material" and its characteristics regarding vein.
has_background	It models the aggregation relationship between each instance of "material" and its characteristics regarding background.

The following table lists the constraints on objects and their attributes and relationships that emerged from the requirements analysis. These constraints are modeled in the conceptual schema: part of them are modeled in an explicit way, others with the mechanisms offered by the OMT model.

Name	*Constraints*
Availability	Each instance of “availability” is relative to exactly one material and one supplier. Two instance of “availability” with the same values of (supplies, is_available) must not exist.
Attributes	
format	Subset of {'Blocks', 'Slabs', 'Tiles', 'Cut to Size', 'Base or Skirt', 'Artwork'}.
shipping	
quantity	Integer greater than zero or null.
indicative_price	The subfield “price” must be a real greater than zero. The value of the subfield “currency” must be contained in the field “name” of the object “currency”.
Relationships	
supplies	Each element of “supplies” must be an element of “supplier”. For each instance of “availability” the field “supplies” must be defined.
is_available	Each element of “is_available” must be an element of “material”. For each instance of “availability” the field “is_available” must be defined.
Background	Each instance of “background” must be pointed by at least an instance of “visual_characteristics”. Two instances of “background” with the same value of the attribute “background_type” must not exist.
Attributes	
background_type	Subset of {‘Uniform’, ‘Crystalline’, ‘Clouded’, ‘Fine Grain’, ‘Gross Grain’, ‘Fine & Gross Grain’, ‘Salt & Pepper’, ‘Spotting’}. For each instance of background the field “background_type” must be defined.
Relationships	
has_background	Each element in “has_background” must be an element of “visual_characteristics”. For each instance of “background” the field “has_background” must be defined.

Name	*Constraints*
Currency	
Attributes	
name	This field is the primary key for “currency”. For each instance of “currency” the field “name” must be defined.
exchange_factor	Real greater than zero.
description	
Manufactures	Each object in “manufactures” is relative to exactly one instance of “material” and “supplier”.
Attributes	
manufacture_ image	For each instance of “manufactures” the field “manufacture_image” must be defined.
description	For each instance of “manufactures” the field “description” must be defined.
Relationships	
has_realized	Each element in “has_realized” must be an element of “supplier”. For each instance of “manufactures” the field “has_realized” must be defined.
is_worked	Each element in “is_worked” must be an element of “material”. For each instance of “manufactures” the field “is_worked” must be defined.

Name	***Constraints***
Material	Each instance of "material" must have at least one "availability", one "quarry" from which is extracted, exactly one "visual_characteristics" and at most one "visual_aspect".
Attributes	
material_name	This field is the primary key for "material". For each element in "material" the field "material_name" must be defined.
material_type	It takes a value from {'Marble', 'Granite', 'Stone', 'Quartzite' and 'Travertine'}. For each instance of "material" the field "material_type" must be defined.
quality	It is defined only if "material_type" is 'Stone' and takes a value from: {'Limestone', 'Slate' and 'Sandstone'}.
finish	Subset of {'Polished', 'Acid Finishing', 'Jet Liquid Honed', 'Hammered', 'Flamed', 'Sand Blasy', 'Split Face', 'Honed'}.
use_mode	Subset of {'Interior', 'Exterior', 'Ground', 'Pedestrian', Wall', 'Carriage-heavy-traffic', 'Carriage-light-traffic'}.
min_price	The subfield "price" must be a real greater than zero. The subfield "currency" must be contained in the field "name" of the object "currency".
max_price	The subfield "price" must be a real greater than zero. The subfield "currency" must be contained in the field "name" of the object "currency".
production_per_month	Integer greater than zero.
Relationships	
is_worked	Each value in "is_worked" must be an element of "manufactures".
is_available	Each value in "is_available" must be an element of "availability". For each element in "material" the field "is_available" must be defined.
is_extracted_by	Each value in "is_extracted_by" must be an element of "quarry". For each element in "material" the field "is_extracted_by" must be defined.
has_appearance	Each value in "has_appearance" must be an element of "visual_characteristics". For each instance of "material" the field "has_appearance" must be defined.
has_aspect	Each value in "has_aspect" must be an element of "visual_aspect". For each element in "material" the field "has_aspect" must be defined.
is_characterized_by	Each value in "is_characterized_by" must be an element of "tests".

Name	***Constraints***
Quarry	Each instance of “quarry” must provide at least one “material”.
Attributes	
name	This field is the primary key for “quarry”. For each element in “quarry” the field “name” must be defined.
address	
phone	
fax	
e_mail	
geographical_area	
map_image	
Relationships	
produces	Each element in “produces” must be an element of “material”. For each instance of “quarry” the field “produces” must be defined.
Supplier	Each instance of “supplier” has at least one “availability”.
Attributes	
name	This field is the primary key for “supplier”. For each element in “supplier” the field “name” must be defined.
address	For each instance of “supplier” the field “address” must be defined.
phone	
fax	
e_mail	
profile	
Relationships	
has_realized	Each element in “has_realized” must be an element of “manufactures”.
supplies	Each element of “supplies” must be an element of “availability”. For each instance of “supplier” the field “supplies” must be defined.

Name	*Constraints*
Tests	Each instance of "tests" must be pointed by at least one "availability".
Attributes	
compression_breaking_load	Real between 0 and 9999.99.
compression_breaking_load_a_f	Real between 0 and 9999.99.
imbibition_coefficient_by_weight	Real between 0.001% and 9.999%.
ultimate_tensile_strength	Real between 0 and 999.99.
impact_test	Real between 0 and 99.99.
thermal_linear_expansion_coef	Real between 0.0001 and 9.9999.
frictional_wear_test	Real between 0.01 and 9.99.
weight_per_unit_of_volume	Real between 0 and 9999.99.
elasticity_module	Real between 0 and 999.999.
microhardness_ Knopp	Real between 0 and 999.
ASTM_C170	Real between 0 and 99999.99.
ASTM_C880	Real between 0 and 9999.99.
ASTM_C97	Real between 0.01 and 9.99.
ASTM_C99	Real between 0 and 9999.99.
Relations	
is_characterized_by	Each element in "is_characterized_by" must be an element of "material". For each instance of "tests" the field "is_characterized_by" must be defined.

Name	***Constraints***
Vein	Each element in “vein” must be pointed by at least one “visual_characteristics”. Two instances of “vein” with the same values of its attributes must not exist.
Attributes	
vein_type	It takes one of the following values: 'No Vein', 'Light Vein', 'Regular Vein', 'Strong Vein', 'Irregular Vein'. For each instance of “vein” the field “vein_type” must be defined.
vein_pattern	It takes a subset of {'Rust', 'Spot', 'Parallel to the bed', 'Against the bed', 'Flowery', '45 Vein', 'Clouded', 'Streak', 'Spotting', 'Arabesque'}. For each instance of “vein” the field “vein_pattern” must be defined.
colour	
Relations	
has_vein	Each element in “has_vein” must be an element of “visual_characteristics”. For each instance of “vein” the field “has_vein” must be defined.
Visual_aspect	Each instance of “visual_aspect” corresponds to exactly one element of “material”.
Attributes	
description	
thumbnail	
lowbitrate	
highbitrate	
downloadable	
Relationships	
has_aspect	Each element in “has_aspect” must be an element of “material”. For each instance of “visual_aspect” the field “has_aspect” must be defined.

Name	***Constraints***
Visual_ characteristics	Each instance of "visual_characteristics" corresponds to exactly one element of "material".
Attributes	
colour_predominance	
colour_hue	Each element in "colour_hue" is a tern of integer between 0 and 255.
Relationships	
has_appearance	Each element in "has_appearance" must be an element of "material". For each instance of "visual_characteristics" the field "has_appearance" must be defined.
has_vein	Each element in "has_vein" must be an element in "vein". For each instance of "visual_characteristics" the field "has_vein" must be defined.
has_background	Each element in "has_background" must be an element in "background". For each instance of "visual_characteristics" the field "has_background" must be defined.

4. Revised logical design

Starting from the above revised conceptual schema, the logical design has been modified taking into account both the characteristics of the Illustra model and the requirements about the estimated frequency of updates and queries on the MIAOW database.

A first step of refinement was done to optimize the information retrieval. Then, the mapping between OM and Illustra mechanisms was defined and applied to schema obtained after the logical refinement step. Finally, a second step of refinement was executed to remove redundant or useless information.

4.1 First refinement step

The first step of refinement was carried out to transform the conceptual schema into a form that is suitable for a faster evaluation of the most frequent queries. According to the non-functional requirements[3] these queries regard the retrieval of the following information:

- a material with a given colour predominance;
- a material whose calculated colour belongs to a particular class of colours;
- a material with a given vein type, pattern and colour;
- a material with a given background type.

To optimize these queries the object "visual_characteristics" and its relationships with "material", "vein" and "background" have been removed, the attributes "colour_predominance" and "colour_hue" have been added to the object "material" and the objects "vein" and "background" have been linked directly with "material".

Notice that this refinement step reduces the complexity of the schema, reducing the distance between "material" and "vein" and between "material" and "background". This has positive effects on the management of the database and on the handling of the insertion operations.

Moreover, a new object, called "palette", and the appropriate relationship with "material" that groups the materials into classes with respect to the calculated colour on a photo of materials, have been introduced.

[3] See Appendix B for the list of these non-functional requirements.

Figures 4.1 and 4.3 show respectively the Level 1 and the object material Level 2, which are produced after this refinement step. Figures 4.2 and 4.4 present instead the explicit constraints of them, after the refinement step.

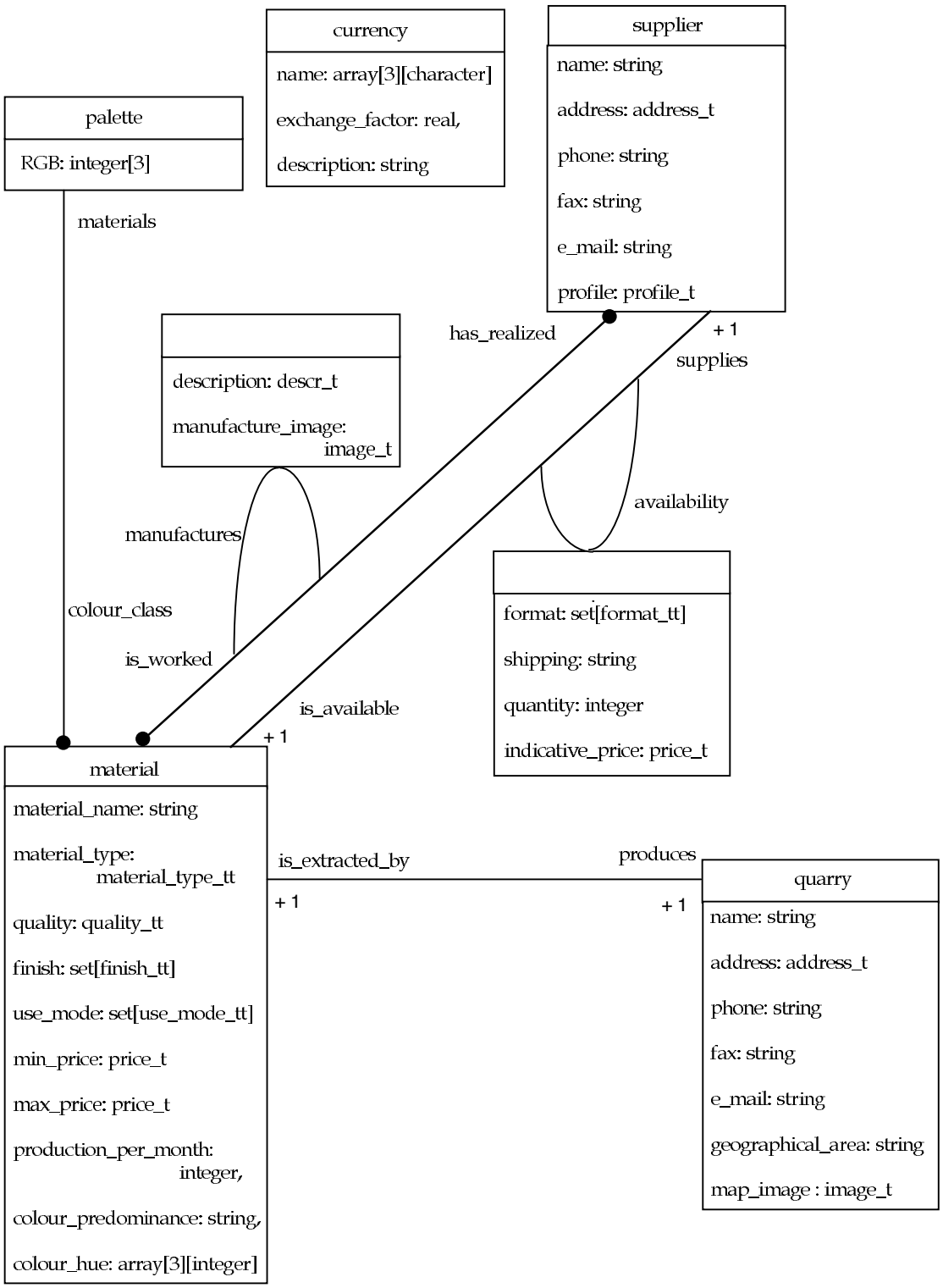

Figure 4.1.- *Refinement 1: Level 1 Object Model*

availability.quantity > 0

availability.indicative_price.price > 0

availability.indicative_price.currency appears in currency.name

(availability.supplies, availability.is_available) is unique

currency.name is the primary key of currency

currency.name is not null

currency.exchange_factor > 0

manufactures.manufacture_image is not null

manufactures.description is not null

material.material_name is the primary key of material

material.material_name is not null

material.quality not null if material.material_type = 'stone'

material.min_price.price > 0

material.min_price.currency appears in currency.name

material.max_price.price > 0

material.max_price.currency appears in currency.name

material.production_per_month > 0

material.colour_hue[1]∈ [0,..,255]

material.colour_hue[2]∈ [0,..,255]

material.colour_hue[3]∈ [0,..,255]

palette.RGB[1]∈ [0,..,255]

palette.RGB[2]∈ [0,..,255]

palette.RGB[3]∈ [0,..,255]

quarry.name is the primary key of quarry

quarry.name is not null

supplier.name is the primary key of supplier

supplier.name is not null

supplier.address is not null

Figure 4.2.- *Refinement 1: Level 1 Object Model - Explicit Constraints*

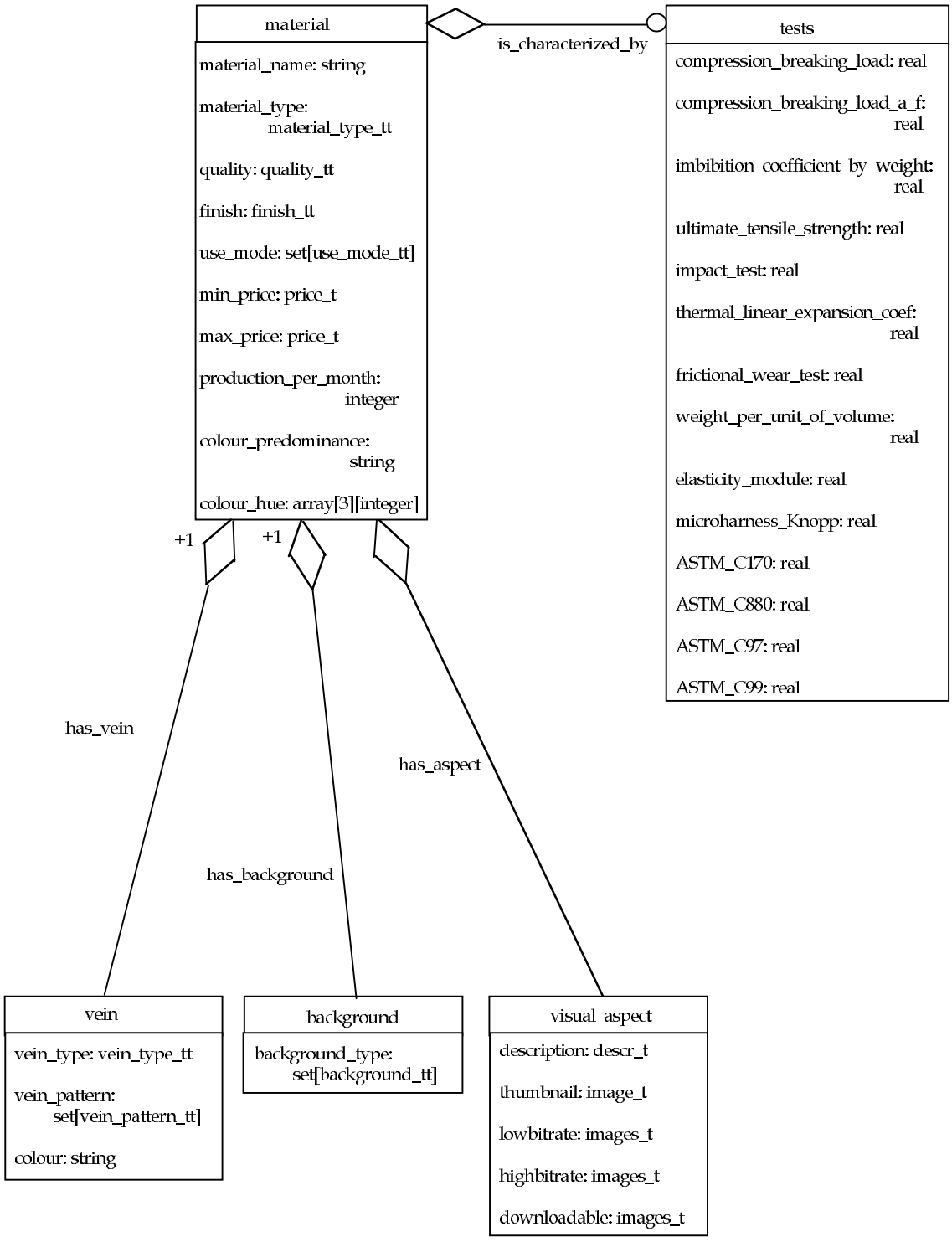

Figure 4.3.- *Refinement 1: Level 2 Object Model - object material*

background.background_type is unique
background.background_type is not null
material.material_name is the primary key of material
material.material_name is not null
material.quality not null if material.material_type = 'stone'
material.min_price.price > 0
material.min_price.currency appears in currency.name
material.max_price.price > 0
material.max_price.currency appears in currency.name
material.production_per_month > 0
material.colour_hue[1] ∈ [0,..,255]
material.colour_hue[2] ∈ [0,..,255]
material.colour_hue[3] ∈ [0,..,255]
tests.compression_breaking_load ∈ [0,..,9999.99]
tests.compression_breaking_load_a_f ∈ [0,..,9999.99]
tests.imbibition_coefficient_by_weight ∈ [0.01,..,9.999]
tests.ultimate_tensile_strength ∈ [0,..,999.99]
tests.impact_test ∈ [0,..,99.99]
tests.thermal_linear_expansion_coef ∈ [0.0001,..,9.9999]
tests.frictional_wear_test ∈ [0.01,..,9.99]
tests.weight_per_unit_of_volume ∈ [0,..,9999.99]
tests.elasticity_module ∈ [0,..,999.99]
tests.microharness_Knopp ∈ [0,..,999]
tests.ASTM_C170 ∈ [0,..,99999.99]
tests.ASTM_C880 ∈ [0,..,9999.99].
tests.ASTM_C97 ∈ [0.01,..,9.99]
tests.ASTM_C99 ∈ [0,..,9999.99]
vein.vein_type is not null
vein.vein_pattern is not null
(vein.vein_type, vein.vein_pattern, vein.colour) is unique

Figure 4.4.- *Refinement 1: Level 2 Object Model - Explicit Constraints of object material*

4.2 Mapping between OM and Illustra object models

Now, we can establish a one-to-one correspondence between the OMT object modeling mechanisms and the set of Illustra data types. Among the various kinds of mapping, one was defined and employed[4], described informally below:

a) Each OM type corresponds to an Illustra type.

- The OM *basic* types correspond to the Illustra types:

1. character ------> **char**
2. descr_t ------> **large_text**
3. image_t ------> **external_file**
4. integer ------> **int**
5. real ------> **real**
6. string ------> **text**

- Each OM *enumerated* type T_tt = {‘val_1’, ‘val_2’, ... , ‘val_n’} corresponds to an Illustra table like

```
create table T of new type T_t
(
val      text    check(val in
(‘val_1’, ‘val_2’, ..... , ‘val_n’))
)
```

that will be pre-loaded with the values ‘val_1’, ‘val_2’, ... , ‘val_n’. Moreover, in the tables in those columns where the type T_tt is used a control is made to assume that the column assumes the right values.

The choice to implement the enumerated types as shown above can, at first sight, appear uselessly complicated. The column C of enumerated type T_tt can be implemented directly in the following way:

```
C    text   check(C in (‘val_1’, ‘val_2’, ... , ‘val_n’))
```

[4] In what follows OM stands for "OMT Object Model".

However, the solution chosen allows a greater flexibility in that, if the enumerated type T_tt should assume values different from the present values, the only thing to do would be to load the table with the new values. Instead, in the second case, given the Illustra characteristics, the deleting of the table, including the column C and the definition and the loading of a new table with the correct new control for the column C, would be necessary. This operation has unfavorable repercussions throughout the database, in that we must maintain the consistency of references.

Leaving open the possibility of modifying the values of the enumerated type is, in this particular context, very important, because a standardization of the nomenclature used in the field of the building materials was not yet given.

- The OM *composed* types without constraints on the components correspond to the Illustra composed types.

- The OM *composed* types with constraints on components are decomposed into Illustra elementary types. For example, if an OM object O has an attribute C of composed type

  T_t ≡ c1: t1 with constraint c,
  c2: t2

  then the Illustra table O, corresponding to the OM object O, does not have the column C, but has the columns C.c1 of type t1 with constraints c and C.c2 of type t2.

- The OM *structured* types correspond to the Illustra structured types:

  1. array[integer][type_t] ------> type_t[**integer**]
  2. set[type_t] ------> **setof**(**text**)
     check (**all** .... **in** .....)

b) Each OM object corresponds to an Illustra table.

c) Each attribute of an OM object corresponds to a column in the Illustra table which implements that object.

d) Each type constraint on an OM attribute corresponds to a control, implemented through the construct check, on the corresponding attribute in the Illustra Schema.

e) Each key constraint on an OM attributes' composition corresponds to a constraint *unique* to the corresponding Illustra attribute, if the OM attributes are not of type *set*. The constraints about the uniqueness of OM attributes of type *set* cannot be checked with the constraint *unique* in Illustra, because it is not defined on *sets*. In this case, the attributes will not be constrained and the operation that inserts the values in these attributes must check that the new value inserted is not already present.

f) Each constraint *not null* on an OM attribute corresponds to a constraint *not null* on the corresponding Illustra attribute.

g) Each OM binary association relationship R, with attributes A1,..., An between the classes C1 and C2 (C1 has role *rel* in R and C2 has role *rel'* in R) is transformed into a table R with attributes A1, ..., An and the two following attributes: *rel_C1_ref* of type *ref* (identifier), with constraint *not null*, and *rel'_C2_ref* of type *ref* (identifier)), with constraint *not null* .

Moreover,

1. If the OM binary association relationship R is a many-to-many relationship, then the attributes *rel_R_ref* of type *setof* (*ref* (identifier)) and *rel'_R_ref* of type *setof* (*ref* (identifier)) become part of the tables which implement C1 and C2 respectively;

2. If the OM binary association relationship R is a one-to-many relationship, then the attributes *rel*_R_*ref* of type *setof* (*ref* (identifier)) and *rel'*_R_*ref* of type *ref* (identifier) become part of the tables which implement C1 and C2 respectively;

3. If the OM binary association relationship R is a many-to-one relationship, then the attributes *rel*_R_*ref* of type *ref* (identifier) and *rel'*_R_*ref* of type *setof* (*ref* (identifier)) become part of the tables which implement C1 and C2 respectively;

4. If the OM binary association relationship R is a one-to-one relationship, then the attributes *rel*_R_*ref* of type *ref* (identifier) and *rel'*_R_*ref* of type *ref* (identifier) become part of the tables which implement C1 and C2 respectively;

In any case, what must happen is that

- if the OM binary association relationship R is total then the insert operation in C1 must be implemented in such a way as to ensure the introduction of a value in R linked with the new value inserted in C1;
- if the inverse of the OM binary association relationship R is total then the insert operation in C2 must be implemented in such a way as to ensure the introduction of a value in R linked with the new value inserted in C2;

Moreover, the insert operation must be implemented in such a way as to insert the right link between C1 and R and between C2 and R when a new value is inserted in R.

The cancel and update operations must also be implemented in such a way as to ensure consistency of the database.

h) Each OM binary association relationship *rel*, without attributes, between the classes C1 and C2, with inverse *rel'*, is transformed into two Illustra reference attributes:

1. If the OM binary association relationship *rel* is a many-to-many relationship, then the attributes *rel_C2_ref* of type *setof* (*ref* (identifier)) and *rel'_C1_ref* of type *setof* (*ref* (identifier)) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total, then the insert operation must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;
   - if the inverse of the OM binary association relationship *rel* is total, then the insert operation in C2 must be implemented in such a way as to ensure the updating of the attribute *rel'_C1_ref* in C2 with the right link between C2 and C1 when a new value is inserted in C2 (with Illustra it sis not at present possible to insert a value in an attribute of type set directly with an insert operation);

2. If the OM binary association relationship *rel* is a one-to-many relationship, then the attributes *rel_C2_ref* of type *setof* (*ref* (identifier)) and *rel'_C1_ref* of type *ref* (identifier) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total, then the insert operation in C1 must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in

C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;

- if the inverse of the OM binary association relationship *rel* is total, then the reference attribute *rel'_C1_ref* will be a *not null* attribute;

3. If the OM binary association relationship *rel* is a many-to-one relationship, then the attributes *rel_C2_ref* of type *ref* (identifier) and *rel'_C1_ref* of type *setof* (*ref* (identifier)) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total, then the insert operation in C1 must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;
   - if the inverse of the OM binary association relationship *rel* is total, then the insert operation in C2 must be implemented in such a way as to ensure the updating of the attribute *rel'_C1_ref* in C2 with the right link between C2 and C1 when a new value is inserted in C2 (with Illustra it is not possible at present to insert a value in an attribute of type set directly with an insert operation);

4. If the OM binary association relationship *rel* is a one-to-one relationship, then the attributes *rel_C2_ref* of type *ref* (identifier) and *rel'_C1_ref* of type *ref* (identifier) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total, then the insert operation in C1 must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;
   - if the inverse of the OM binary association relationship *rel* is total, then the reference attribute *rel'_C1_ref* will be a *not null* attribute;

In any case, the insert operation must be implemented in such a way as to insert the inverse link when a link from Ci to Cj is inserted. Moreover, the cancel and update operations must be implemented in such a way as to ensure consistency of the database.

i) Each OM aggregation relationship *rel* between the classes C1 and C2 is transformed in two Illustra attributes reference:

1. If the OM aggregation relationship *rel* is a many-to-many relationship, then the attributes *rel_C2_ref* of type *setof* (*ref* (identifier)) and *rel_C1_ref* of type *setof* (*ref* (identifier)) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total, then the insert operation must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;
   - if the inverse of the OM binary association relationship *rel* is total, then the insert operation in C2 must be implemented in such a way as to ensure the updating of the attribute *rel_C1_ref* in C2 with the right link between C2 and C1 when a new value is inserted in C2 (with Illustra it is not at present possible to insert a value in an attribute of type set directly with an insert operation);

2. If the OM aggregation relationship *rel* is a one-to-many relationship, then the attributes *rel_C2_ref* of type *setof* (*ref* (identifier)) and *rel_C1_ref* of type *ref* (identifier) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total, then the insert operation in C1 must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;
   - if the inverse of the OM binary association relationship *rel* is total, then the reference attribute *rel_C1_ref* will be a *not null* attribute;

3. If the OM aggregation relationship *rel* is a many-to-one relationship, then the attributes *rel_C2_ref* of type *ref* (identifier) and *rel_C1_ref* of type *setof* (*ref* (identifier)) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total then the insert operation in C1 must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;

- if the inverse of the OM binary association relationship *rel* is total, then the insert operation in C2 must be implemented in such a way as to ensure the updating of the attribute *rel_C1_ref* in C2 with the right link between C2 and C1 when a new value is inserted in C2 (with Illustra it is not possible at present to insert a value in an attribute of type set directly with an insert operation);

4. If the OM aggregation relationship *rel* is a one-to-one relationship, then the attributes *rel_C2_ref* of type *ref* (identifier) and *rel_C1_ref* of type *ref* (identifier) become part of the tables which implement C1 and C2 respectively.

   Moreover,

   - if the OM binary association relationship *rel* is total, then the insert operation in C1 must be implemented in such a way as to ensure the introduction of a value in C2 linked with the new value inserted in C1 and to insert the right link between C1 and C2 when a new value is inserted in C2;
   - if the inverse of the OM binary association relationship *rel* is total, then the reference attribute *rel_C1_ref* will be a *not null* attribute;

In any case, the insert operation must be implemented in such a way as to insert the inverse link when a link from Ci to Cj is inserted. Moreover, the cancel and update operations must be implemented in such a way as to ensure consistency of the database.

4.3 Mapping OM schema into Illustra schema

Applying the above mapping to the object schema, obtained after the first step of refinement, the following Illustra schema is generated.

The first table shows the Illustra composed types, whereas the second presents the Illustra tables.

Type name	*Components*	*Components type*
address_t		
	street	**text**
	number	**text**
	country	**text**
	city	**text**
	zipcode	**text**
	state	**text**
profile_t		
	society_image	**external_file**
	description	**large_text**
identifier		
	id	**text**

Table name	*Column name*	*Column type*	*Columns constraints*
background_ type			
	val	**text**	**check**(val **in** ('Uniform', 'Crystalline', 'Clouded', 'Fine Grain', 'Gross Grain', 'Fine & Gross Grain', 'Salt & Pepper', 'Spotting'))
finish			
	val	**text**	**check**(val **in** ('Polished', 'Flamed', 'Sand Blast', 'Hammered', 'Split Face', 'Jet Liquid Honed', 'Honed', 'Acid Finishing'))

Table name	*Column name*	*Column type*	*Columns constraints*
format			
	val	**text**	**check**(val **in** ('Blocks', 'Slabs', 'Tiles', 'Cut to Size', 'Base or Skirt', 'Artwork'))
material_ type			
	val	**text**	**check**(val **in** ('Marble', 'Granite', 'Stone', 'Quartzite', 'Travertine'))
quality			
	val	**text**	**check**(val **in** ('Limestone', 'Sandstone', 'Slate'))
use_mode			
	val	**text**	**check**(val **in** ('Interior', 'Exterior', 'Ground', 'Wall', 'Pedestrian', 'Carriage-heavy-traffic', 'Carriage-light-traffic'))
vein_type			
	val	**text**	**check**(val **in** ('No Vein', 'Light Vein', 'Regular Vein', 'Strong Vein', 'Irregular Vein'))
vein_pattern			
	val	**text**	**check**(val **in** ('Rust', 'Spot', 'Parallel to the bed', 'Against the bed', 'Flowery', '45 Vein', 'Clouded', 'Streak', 'Spotting', 'Arabesque'))

Table name	*Column name*	*Column type*	*Columns constraints*
availability			**unique** (is_available_MATERIAL_ ref, supplies_SUPPLIER_ref)
	format	**setof**(**text**)	**check**(**all** format **in** (**select** val **from** format))
	shipping	**text**	
	quantity	**int**	**check**(quantity **>** 0)
	indicative_price. price	**real**	**check**(indicative_price.price **>** 0)
	indicative_price. currency	**char**(3)	**check**(indicative_price. currency **in** (**select** name **from** currency))
	is_available_ MATERIAL_ref	**ref**(identifier)	**not null**
	supplies_ SUPPLIER_ref	**ref**(identifier)	**not null**
background			
	background_ type	**setof**(**text**)	**check**(**all** background_type **in** (**select** val **from** background_type)) **not null**
	has_background_ MATERIAL_ ref	**setof**(**ref** (identifier))	
currency			
	name	**char**(3)	**not null unique**
	exchange_factor	**real**	**check**(exchange_factor **>** 0)
	description	**text**	

Table name	*Column name*	*Column type*	*Columns constraints*
manufactures			
	description	**large_text**	**not null**
	manufacture_ image	**external_ file**	**not null**
	has_realized_ SUPPLIER_ref	**ref**(identifier)	**not null**
	is_worked_ MATERIAL_ref	**ref**(identifier)	**not null**
material			
	material_name	**text**	**not null unique**
	material_type	**text**	**check**(material_type **in** (**select** val **from** material_type)) **not null**
	quality	**text**	**check**(quality **in** (**select** val **from** quality))
	use_mode	**setof**(**text**)	**check**(**all** use_mode **in** (**select** val **from** use_mode))
	min_price.price	**real**	**check**(min_price.price **>** 0)
	min_price. currency	**char**(3)	**check**(min_price.currency **in** (**select** name **from** currency))
	max_price.price	**real**	**check**(min_price.price **>** 0)
	max_price. currency	**char**(3)	**check**(max_price.currency **in** (**select** name **from** currency))
	production_per_ month	**int**	**check**(production_per_ month **>** 0)
	colour_ predominance	**text**	

Table name	*Column name*	*Column type*	*Columns constraints*
material			
	colour_hue	**int**[3]	**check**(colour_hue[1] **between** 0 **and** 255 **and** colour_hue[2] **between** 0 **and** 255 **and** colour_hue[3] **between** 0 **and** 255)
	has_aspect_ VISUAL_ ASPECT_ref	**ref**(identifier)	
	has_background_ BACKGROUND_ref	**ref**(identifier)	
	has_vein_VEIN_ ref	**ref**(identifier)	
	is_extracted_by_ QUARRY_ref	**setof**(**ref** (identifier))	
	is_available_ AVAILABILITY_ref	**setof**(**ref** (identifier)),	
	is_worked_ MANUFACTURES_ ref	**setof**(**ref** (identifier))	
	colour_class_ PALETTE_ref	**ref**(identifier)	
	is_characterized_ by_TESTS_ref	**ref**(identifier)	
palette			
	RGB	**int**[3]	**check**(RGB[1] **between** 0 **and** 255 **and** RGB[2] **between** 0 **and** 255 **and** RGB[3] **between** 0 **and** 255),
	materials	**setof**(**ref** (identifier))	

Table name	*Column name*	*Column type*	*Columns constraints*
quarry			
	name	**text**	**not null unique**
	address	address_t	
	phone	**text**	
	fax	**text**	
	e_mail	**text**	
	geographical_area	**text**	
	map_image	**external_ file**	
	produces_ MATERIAL_ref	**setof**(**ref** (identifier))	
supplier			
	name	**text**	**not null unique**
	address	address_t	**not null**
	phone	**text**	
	fax	**text**	
	e_mail	**text**	
	profile	profile_t,	
	has_realized_ MANUFACTURES_ ref	**setof**(**ref** (identifier)),	
	supplies_ AVAILABILITY_ref	**setof**(**ref** (identifier))	

Table name	*Column name*	*Column type*	*Columns constraints*
tests			
	compression_ breaking_load	**real**	**check**(compression_breaking_load **between** 0 **and** 9999.99)
	compression_ breaking_load_a_f	**real**	**check**(compression_breaking_load_ a_f **between** 0 **and** 9999.99)
	imbibition_ coefficient_by_ weight	**real**	**check**(imbibition_coefficient_by_ weight **between** 0.001 **and** 9.999)
	ultimate_tensile_ strength	**real**	**check**(ultimate_tensile_strength **between** 0 **and** 999.99)
	impact_test	**real**	**check**(impact_test **between** 0 **and** 99.99)
	thermal_linear_ expansion_coef	**real**	**check**(thermal_linear_expansion_ coef **between** 0.0001 **and** 9.9999)
	frictional_wear_test	**real**	**check**(frictional_wear_test **between** 0.01 **and** 9.99)
	weight_per_unit_of _volume	**real**	**check**(weight_per_unit_of_volume **between** 0 **and** 9999.99)
	elasticity_module	**real**	**check**(elasticity_module **between** 0 **and** 999.999)
	microhardness_ Knopp	**real**	**check**(microhardness_Knopp **between** 0 **and** 999)
	ASTM_C170	**real**	**check**(ASTM_C170 **between** 0 **and** 99999.99)
	ASTM_C880	**real**	**check**(ASTM_C880 **between** 0 **and** 9999.99)
	ASTM_C97	**real**	**check**(ASTM_C97 **between** 0.01 **and** 9.99)
	ASTM_C99	**real**	**check**(ASTM_C99 **between** 0 **and** 9999.99)
	is_characterized_by _MATERIAL_ref	**ref**(identifier)	**not null unique**

Table name	*Column name*	*Column type*	*Columns constraints*
vein			
	vein_type	**text**	**check**(vein_type **in** (**select** val **from** vein_type)) **not null**
	vein_pattern	**setof**(**text**)	**check**(**all** vein_pattern **in** (**select** val **from** vein_pattern)) **not null**
	colour	**text**	
	has_vein_ MATERIAL_ref	**setof**(**ref** (identifier))	
visual_aspect			
	description	**large_text**	
	thumbnail	**external_ file**	
	lowbitrate	**external_ file**	
	highbitrate	**external_ file**	
	downloadable	**external_ file**	
	has_aspect_ MATERIAL_ref	**ref**(identifier)	**not null**

As shown in the previous section, some constraints of the OMT Object Model cannot be expressed using the Illustra mechanisms. For example, the constraints about the uniqueness of the attribute “background_type” of the table “background” cannot be checked with the constraint *unique*.

A similar consideration can be done for the uniqueness of the triple (vein_type, vein_pattern, colour) in vein.

In these and in other cases underlined in the section “Mapping between OM and Illustra object model”, the operations that insert, delete and update these values must check that the new values inserted verify the constraints.

4.4 Second step of refinement

A second step of refinement was applied to substitute into the object “material” the attributes “max_price.currency” and “min_price.currency” with the attribute “currency”. The two attributes contain in fact the same information (the current money with which the minimum and maximum prices are expressed).

To make things easier, the attributes “indicative_price.price” and “indicative_price.currency” of “availability” and “max_price.price” and “min_price.price” of “material” were also renamed respectively “indicative_price”, “currency”, “max_price” and “min_price”.

As a last modification, the relation “class_colour_PALETTE_ref” between tables “material” and “palette” was deleted to optimize the database management. Moreover, this deleting has not reduced the performance because it was not interesting to know to which class of colour a given material belongs. So, the relation between “material” and “palette” would never be traveled from “material” to “palette”.

The schema obtained after these modifications is then equal to the previous one, but with the table “material” and “availability” modified in the following way:

Table name	*Column name*	*Column type*	*Columns constraints*
availability			**unique** (is_available_MATERIAL_ref, supplies_SUPPLIER_ref)
	format	**setof**(**text**)	**check**(**all** format **in** (**select** val **from** format))
	shipping	**text**	
	quantity	**int**	**check**(quantity **>** 0)
	indicative_price	**real**	**check**(indicative_price **>** 0)
	currency	**char**(3)	**check**(currency **in** (**select** name **from** currency))
	is_available_ MATERIAL_ref	**ref**(identifier)	**not null**
	supplies_ SUPPLIER_ref	**ref**(identifier)	**not null**
material			
	material_name	**text**	**not null unique**
	material_type	**text**	**check**(material_type **in** (**select** val **from** material_type)) **not null**
	quality	**text**	**check**(quality **in** (**select** val **from** quality))
	use_mode	**setof**(**text**)	**check**(**all** use_mode **in** (**select** val **from** use_mode))
	min_price	**real**	**check**(min_price **>** 0)
	max_price	**real**	**check**(min_price **>** 0)
	currency	**char**(3)	**check**(currency **in** (**select** name **from** currency))
	production_per_ month	**int**	**check**(production_per_month **>** 0)
	colour_ predominance	**text**	

Table name	*Column name*	*Column type*	*Columns constraints*
material			
	colour_hue	**int**[3]	**check**(colour_hue[1] **between** 0 **and** 255 **and** colour_hue[2] **between** 0 **and** 255 **and** colour_hue[3] **between** 0 **and** 255)
	has_aspect_VISUAL_ASPECT_ref	**ref**(identifier)	
	has_background_BACKGROUND_ref	**ref**(identifier)	
	has_vein_VEIN_ref	**ref**(identifier)	
	is_extracted_by_QUARRY_ref	**setof**(**ref** (identifier))	
	is_available_AVAILABILITY_ref	**setof**(**ref** (identifier)),	
	is_worked_MANUFACTURES_ref	**setof**(**ref** (identifier))	
	is_characterized_by_TESTS_ref	**ref**(identifier)	
	colour_class_PALETTE_ref	**ref**(identifier)	

5. Implementing the Illustra schema

The implementation phase consisted in writing a sequence of Illustra instructions to create the appropriate structures. The process of creating the structures provided by the Illustra system itself was very similar to that employed to create a relational schema.

```
create schema miaow authorization miaow;

set schema miaow;

-- Types definitions.

create type address_t
(
        street          text,
        number          text,
        country         text,
        city            text,
        zipcode         text,
        state           text
);

create type profile_t
(
        society_image   external_file,
        description     large_text
);

create type identifier
(
        id      text
);
```

-- **Enumerate set definitions.**

```
create table background_type of new type background_type_t
(
        val     text     check(val in ('Uniform', 'Crystalline', 'Clouded',  'Fine
                                Grain', 'Gross Grain', 'Fine & Gross Grain', 'Salt
                                & Pepper', 'Spotting'))
);

create table finish of new type finish_t
(
        val     text     check(val in ('Polished', 'Flamed', 'Sand Blast',
                                'Hammered', 'Split Face', 'Jet Liquid Honed',
                                'Honed', 'Acid Finishing'))
);

create table format of new type format_t
(
        val     text     check(val in ('Blocks', 'Slabs', 'Tiles', 'Cut to Size',
                                'Base or Skirt', 'Artwork'))
);

create table material_type of new type material_type_t
(
        val     text     check(val in ('Marble', 'Granite', 'Stone',
                                'Quartzite', 'Travertine'))
);

create table quality of new type quality_t
(
        val     text     check(val in ('Limestone', 'Sandstone', 'Slate'))
);
```

```
create table use_mode of new type use_mode_t
(
        val     text    check(val in ('Interior', 'Exterior', 'Ground', 'Wall',
                                'Pedestrian', 'Carriage-heavy-traffic', 'Carriage-
                                light-traffic'))
);

create table vein_type of new type vein_type_t
(
        val     text    check(val in ('No Vein', 'Light Vein', 'Regular
                                Vein', 'Strong Vein', 'Irregular Vein'))
);

create table vein_pattern of new type vein_pattern_t
(
        val    text    check(val in ('Rust', 'Spot', 'Parallel to the bed', 'Against
                                the bed', 'Flowery', '45 Vein', 'Clouded', 'Streak',
                                'Spotting', 'Arabesque'))
);
```

-- **Tables definitions.**

```
create table availability of new type availability_t
(
        format              setof(text) check(all format in
                                                (select val from format)),
        shipping            text,
        quantity            int check(quantity > 0),
        indicative_price    real check(indicative_price > 0),
        currency            char(3) check(currency in
                                                (select name from currency)),

-- Association relations of the Object Model.
-- This column implements the relationship between availability and material.

        is_available_MATERIAL_ref       ref(identifier) not null,

-- This column implements the relationship between availability and supplier.

        supplies_SUPPLIER_ref           ref(identifier) not null,

        unique (is_available_MATERIAL_ref, supplies_SUPPLIER_ref)
);
```

```
create table background of new type background_t
(
        background_type    setof(text)     check(all background_type in
                                    (select val from background_type)) not null,

-- Aggregation relations of the Object Model.
-- This column implements the relationship between background and material.

        has_background_MATERIAL_ref         setof(ref(identifier))
);

create table currency of new type currency_t
(
        name                    char(3) not null unique,
        exchange_factor         real check(exchange_factor > 0),
        description              text
);

create table manufactures of new type manufactures_t
(
        description                     large_text not null,
        manufacture_image               external_file not null,

-- Association relations of the Object Model.
-- This column implements the relationship between manufactures and supplier.

        has_realized_SUPPLIER_ref           ref(identifier) not null,

-- This column implements the relationship between manufactures and material.

        is_worked_MATERIAL_ref              ref(identifier) not null
);
```

```
create table material of new type material_t
(
        material_name           text not null unique,
        material_type           text check(material_type in
                                    (select val from material_type)) not null,
        quality                 text check(quality in
                                            (select val from quality)),
        finish                  setof(text) check(all finish in
                                                    (select val from finish)),
        use_mode                setof(text) check(all use_mode in
                                                    (select val from use_mode)),
        min_price               real check(min_price > 0),
        max_price               real check(max_price > 0),
        currency                char(3) check(currency in
                                                    (select name from currency)),
        production_per_month    int check(production_per_month > 0),
        colour_predominance     text,
        colour_hue              int[3]
                                    check(colour_hue[1] between 0  and 255
                                        and colour_hue[2] between 0  and 255
                                        and colour_hue[3] between 0  and 255),

-- Aggregation relations of the Object Model.
-- This column implements the relationship between material and images.

        has_aspect_VISUAL_ASPECT_ref    ref(identifier),

-- This column implements the relationship between material and background.

        has_background_BACKGROUND_ref   ref(identifier),

-- This column implements the relationship between material and vein.
```

```
        has_vein_VEIN_ref               ref(identifier),

-- This column implements the relationship between availability and tests.

        is_characterized_by_TESTS_ref   ref(identifier),

-- Association relations of the Object Model.
-- This column implements the relationship between material and quarry.

        is_extracted_by_QUARRY_ref      setof(ref(identifier)),

-- This column implements the relationship between material and availability.

        is_available_AVAILABILITY_ref   setof(ref(identifier)),

-- This column implements the relationship between material and manufactures.

        is_worked_MANUFACTURES_ref      setof(ref(identifier))

);
```

```
create table palette of new type palette_t
(
        RGB             int[3]    check(RGB[1] between 0  and 255
                                        and RGB[2] between 0  and 255
                                        and RGB[3] between 0  and 255),

-- This column identifies the materials that belong to a class of colour.

        materials       setof(ref(identifier))
);

create table quarry of new type quarry_t
(
        name                    text not null unique,
        address                 address_t,
        phone                   text,
        fax                     text,
        e_mail                  text,
        geographical_area       text,
        map_image               external_file,

-- Association relations of the Object Model.
-- This column implements the relationship between quarry and material.

        produces_MATERIAL_ref           setof(ref(identifier))
);
```

```
create table supplier of new type supplier_t
(
        name            text not null unique,
        address         address_t not null,
        phone           text,
        fax             text,
        e_mail          text,
        profile         profile_t,

-- Association relations of the Object Model.
-- This column implements the relationship between supplier and manufactures.

        has_realized_MANUFACTURES_ref       setof(ref(identifier)),

-- This column implements the relationship between supplier and availability.

        supplies_AVAILABILITY_ref           setof(ref(identifier))
);

create table tests of new type tests_t
(
        compression_breaking_load           real
                                check(compression_breaking_load between 0 and 9999.99),
        compression_breaking_load_a_f       real
                                check(compression_breaking_load_a_f between 0 and 9999.99),
        imbibition_coefficient_by_weight    real
                        check(imbibition_coefficient_by_weight between 0.001 and 9.999),
        ultimate_tensile_strength           real
                                check(ultimate_tensile_strength between 0 and 999.99),
        impact_test                         real
                                check(impact_test between 0 and 99.99),
        thermal_linear_expansion_coef       real
                        check(thermal_linear_expansion_coef between 0.0001 and 9.9999),
```

```
        frictional_wear_test                    real
                                    check(frictional_wear_test between 0.01 and 9.99),
        weight_per_unit_of_volume               real
                                check(weight_per_unit_of_volume between 0 and 9999.99),
        elasticity_module                       real
                                    check(elasticity_module between 0 and 999.999),
        microhardness_Knopp                     real
                                    check(microhardness_Knopp between 0 and 999),
        ASTM_C170                               real
                                    check(ASTM_C170 between 0 and 99999.99),
        ASTM_C880                               real
                                    check(ASTM_C880 between 0 and 9999.99),
        ASTM_C97                                real
                                    check(ASTM_C97 between 0.01 and 9.99),
        ASTM_C99                                real
                                    check(ASTM_C99 between 0 and 9999.99),

-- Aggregation relations of the Object Model.
-- This column implements the relationship between tests and material.

        is_characterized_by_MATERIAL_ref        ref(identifier)
                                                    not null unique
);
```

```
create table vein of new type vein_t
(
        vein_type           text     check(vein_type in
                                          (select val from vein_type)) not null,
        vein_pattern        setof(text)   check(all vein_pattern in
                                        (select val from vein_pattern)) not null,
        colour              text,

-- Aggregation relations of the Object Model.
-- This column implements the relationship between vein and material.

        has_vein_MATERIAL_ref   setof(ref(identifier))
);

create table visual_aspect of new type visual_aspect_t
(
        description         large_text,
        thumbnail           external_file,
        lowbitrate          external_file,
        highbitrate         external_file,
        downloadable        external_file,

-- Aggregation relations of the Object Model.
-- This column implements the relationship between images and material.

        has_aspect_MATERIAL_ref         ref(identifier)     not null
);
```

-- **Grant privileges.**

grant all privileges on table availability **to public**;
grant all privileges on table background **to public**;
grant all privileges on table background_type **to public**;
grant all privileges on table finish **to public**;
grant all privileges on table format **to public**;
grant all privileges on table manufactures **to public**;
grant all privileges on table material **to public**;
grant all privileges on table material_type **to public**;
grant all privileges on table palette **to public**;
grant all privileges on table quality **to public**;
grant all privileges on table quarry **to public**;
grant all privileges on table supplier **to public**;
grant all privileges on table tests **to public**;
grant all privileges on table use_mode **to public**;
grant all privileges on table vein **to public**;
grant all privileges on table vein_pattern **to public**;
grant all privileges on table vein_type **to public**;
grant all privileges on table visual_aspect **to public**;

-- Initialization of pre-defined tables.

insert into background_type **values** ('Fine & Gross Grain');
insert into background_type **values** ('Salt & Pepper');
insert into background_type **values** ('Gross Grain');
insert into background_type **values** ('Crystalline');
insert into background_type **values** ('Fine Grain');
insert into background_type **values** ('Spotting');
insert into background_type **values** ('Uniform');
insert into background_type **values** ('Clouded');

```
insert into finish values ('Jet Liquid Honed');
insert into finish values ('Acid Finishing');
insert into finish values ('Hammered');
insert into finish values ('Sand Blast');
insert into finish values ('Split Face');
insert into finish values ('Polished');
insert into finish values ('Flamed');
insert into finish values ('Honed');

insert into format values ('Base or Skirt');
insert into format values ('Cut to Size');
insert into format values ('Artwork');
insert into format values ('Blocks');
insert into format values ('Slabs');
insert into format values ('Tiles');

insert into material_type values ('Travertine');
insert into material_type values ('Quartzite');
insert into material_type values ('Granite');
insert into material_type values ('Marble');
insert into material_type values ('Stone');

insert into quality values ('Limestone');
insert into quality values ('Sandstone');
insert into quality values ('Slate');

insert into use_mode values ('Carriage-heavy-traffic');
insert into use_mode values ('Carriage-light-traffic');
insert into use_mode values ('Pedestrian');
insert into use_mode values ('Interior');
insert into use_mode values ('Exterior');
insert into use_mode values ('Ground');
insert into use_mode values ('Wall');
```

insert into vein_type **values** ('Irregular Vein');
insert into vein_type **values** ('Regular Vein');
insert into vein_type **values** ('Strong Vein');
insert into vein_type **values** ('Light Vein');
insert into vein_type **values** ('No Vein');

insert into vein_pattern **values** ('Parallel to the bed');
insert into vein_pattern **values** ('Against the bed');
insert into vein_pattern **values** ('Arabesque');
insert into vein_pattern **values** ('Spotting');
insert into vein_pattern **values** ('Flowery');
insert into vein_pattern **values** ('Clouded');
insert into vein_pattern **values** ('45 Vein');
insert into vein_pattern **values** ('Streak');
insert into vein_pattern **values** ('Rust');
insert into vein_pattern **values** ('Spot');

-- Initialization of table currency.

-- We initialized the table currency with the following values, but subsequently other
-- values can be inserted in the table if required.
insert into currency **values** (‘XEU’, 1.2838, ‘European Currency Units’);
insert into currency **values** (‘GRD’, 236.02, ‘Greek Drachmas’);
insert into currency **values** (‘DEM’, 1.5256, ‘German Marks’);
insert into currency **values** (‘SEK’, 6.6235, ‘Swedish Krona’);
insert into currency **values** (‘FRF’, 5.1643, ‘French Francs’);
insert into currency **values** (‘USD’, 1, ‘American Dollars’);
insert into currency **values** (‘JPY’, 111.37, ‘Japanese Yen’);
insert into currency **values** (‘ITL’, 1522.10, ‘Italian Lira’);

6. Bibliography

[Che76] Chen, P. P., The entity-relationship model - toward a unified view of data. ACM Transactions on Database Systems, 1 (1), pp.9-36, 1976.

[ILL-IMAGE] Illustra Image DataBlade Release 1.2, June 1994.

[ILL-SERVER] Illustra Server Release 3.2, 1995.

[ILL-VIR] Illustra Visual Information Retrieval (VIR) DataBlade Release 1.1, January 1996.

[ILL-WEB] Illustra Web DataBlade Module Release 2.2, July 1996.

[MIAOW-DA] MIAOW Application Data Analysis Report, MIAOW-IME-REP-001-005, 1996.

[MIAOW-DD] MIAOW Design Document of the MIAOW Multimedia Database, 1996.

[MIAOW-RQ] MIAOW User Requirement Document, MIAOW-ITC-RQ-001-002, 1996.

[MIAOW-TT] MIAOW Report on Telematic Technologies, MIAOW-ITC-REP-001-001, 1995.

[Rum91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. Object-Oriented Modeling and Design. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1991.

[Sto96] Michael Stonebraker and Doroty Moore, Object-Relational DBMSs: the Next Great Wave, The Morgan Kaufmann Series in Data Management Systems, Series Editor, Jim Gray, 1996.

7. Acronyms

DBMS	DataBase Management System
ER	Entity-Relationship
GIF	Graphical Interchange Format
JPEG	Joint Photographic Expert Group
OM	OMT Object Model
OMT	Object Modeling Technique
RGB	Red, Green and Blue
TGA	Targa

8. Appendix A: OMT Object Model

Object Modeling Technique (OMT) [Rum91] is an object-oriented methodology which supports the entire software life cycle. It requires the building of a starting model of the application domain and, subsequently, refines it during the design phase. This methodology comprises the following stages:

- *Analysis*. This stage is concerned with understanding and modeling the application domain. The output from analysis is a formal model that captures the essential aspects of the system: the objects and their relationships, the dynamic behavior of each object and the functional transformations of the data.
- *System Design* The overall architecture of the system is established during this stage. Using the object model as a guide, the system is organized into subsystems.
- *Object Design*. During this stage, there is a shift in emphasis from application concepts towards computer concepts. The analysis models are elaborated, refined, and then optimized to produce a practical design. First, the basic algorithms are chosen to implement each major function (method) of the system. Based on these algorithms, the structure of the object model is then optimized for efficient implementation.
- *Implementation*. The object classes and relationships developed during object design are finally translated into a particular programming language.

During all these stages modeling is the main activity. To appropriately support this task three models are provided; each model represents related but different viewpoints, each capturing important system aspects.

- *Object model*. It captures the static structure of a system by showing the system objects, the relationships between these objects, and the attributes and operations that characterize each class of objects.
- *Dynamic model*: It describes those aspects of a system which deal with flow of control, interactions and sequences of operations.
- *Functional model*. It describes the computations within a system by specifying how output values in a computation are derived from input values, without regard to the order in which the values are computed.

Each model is equipped with an intuitive representation which greatly facilitate its understanding by and communication with the customer.

The three above models are not completely independent, but each model can be examined and understood by itself to a large extent since the interconnections between the three models are limited and explicit.

In this report our interest is particularly addressed to the *Object Model* described in the following section.

8.1 Object Model

The Object-Oriented approach to the development of a system focuses first on identifying objects from the application domain, then fitting procedures around them. As a consequence of this approach the object model is considered the most important of the three OMT models.

The object is the main modeling mechanism of the object model. An object is characterized by an identity which distinguishes it from the others.

A class describes a group of objects with similar *attributes*, common *operations*, and common *relationships* to other objects.

An attribute is characterized by a name which is unique within a class. Each attribute has a value for each object instance, this value may vary with time. The values admitted are pure data value, i.e. they cannot be objects.

An operation is a function or transformation that may be applied to or by objects in a class. The same operation may have different implementations (*methods*) in different classes. An object "knows" its class and hence the right implementation of the operation. When an operation has methods associated with several classes it is important that the methods all have the same signature, i.e. the number and types of arguments and the type of the result value must be the same.

A set of *constraints* restricts the possible values of attributes and operations. Each constraint is expressed under the form of a condition to be satisfied.

A graphical notation is provided to express object models. Figure A.1 shows an *object diagram* composed of a *class diagram*. A class diagram is drawn as a box which may have as many as three regions. The regions contain, from top to bottom: class name, list of attributes, and list of operations. Attributes and operations may or may not be shown; it depends on the level of detail desired.

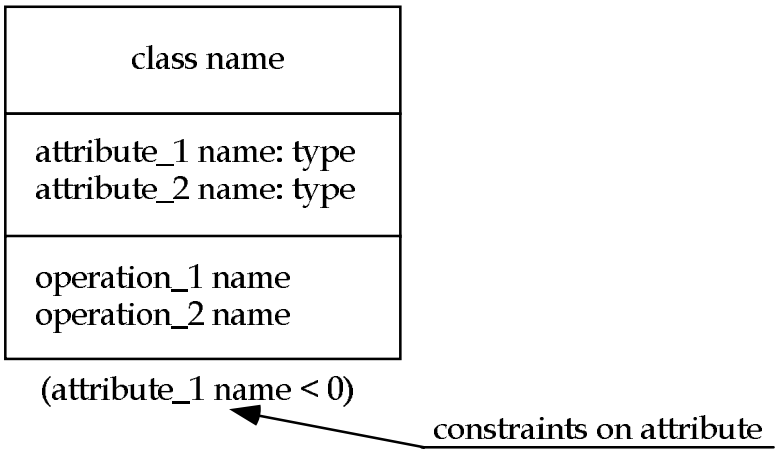

Figure A.1.- *An Object Diagram*

Multiplicity constraints, which specify how many instances of a class may relate to a single instance of an associated class, can be expressed graphically by using appropriate line terminators. The possible line terminators are shown in figure A.2.

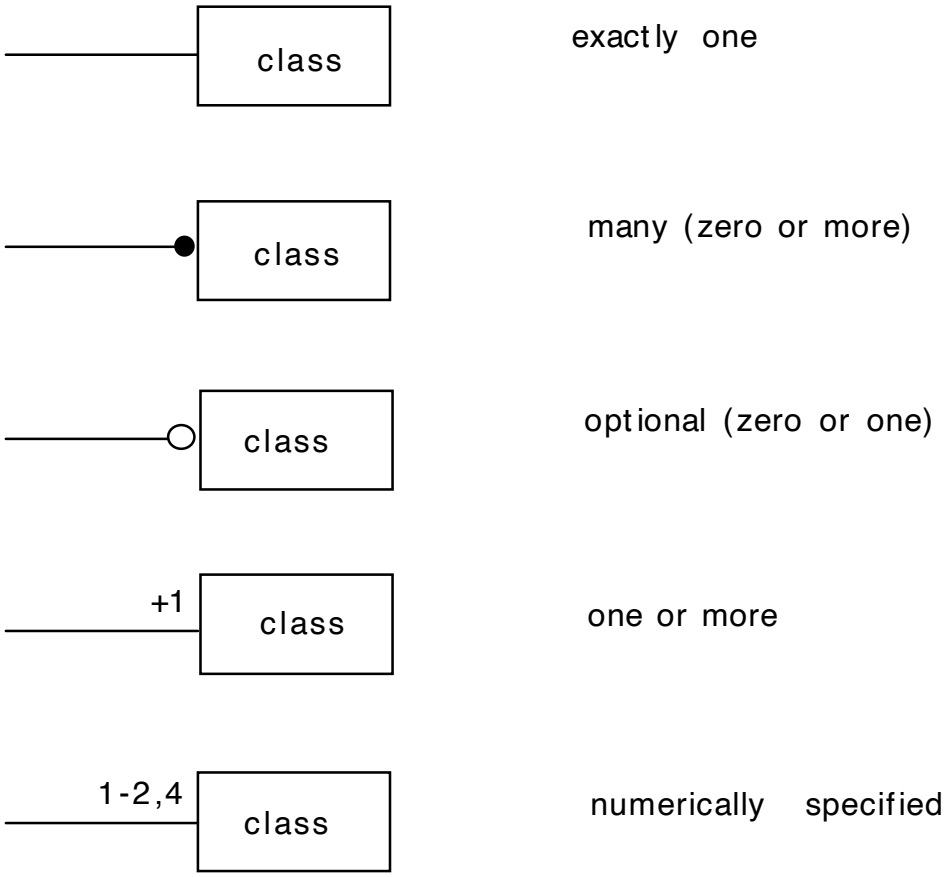

Figure A..2 - *Multiplicity constraints*

Association is a modeling mechanism which describes a group of links between objects. All the links in an association connect objects from the same classes. Associations may be binary, ternary, or higher order; independently from the arity they are inherently bi-directional; this means that, for example, even if the name of a binary association reads in a particular direction, the binary association can be traversed in either direction. The link in an association can have some properties described by attributes, called a *link attribute*. Figure A.3 shows how binary associations are represented in a class diagram.

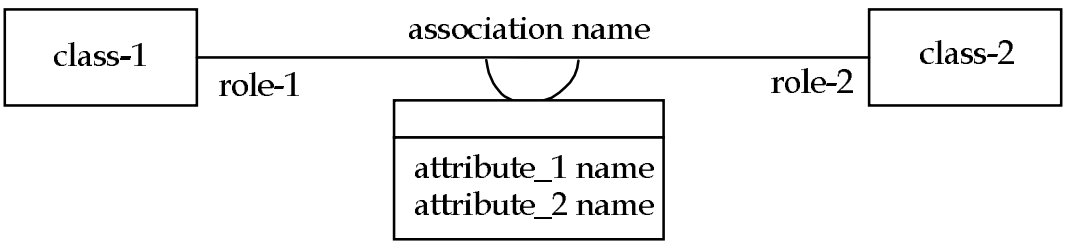

Figure A.3.- *Binary association*

Several modeling mechanisms are provided to enrich the semantics of an association. Roles are one of these mechanisms.

Role can be attached to a binary association. Each role identifies uniquely an object, or a set of objects, associated with an object at the other end.

Aggregation is a particular case of association relationship. This relationship, often called *part-of*, links an aggregate object to its component objects. The component objects may or may not exist apart from the aggregate, in addition they can appear in multiple aggregates.

Figure A.4 shows the graphical representation of an aggregation relationship.

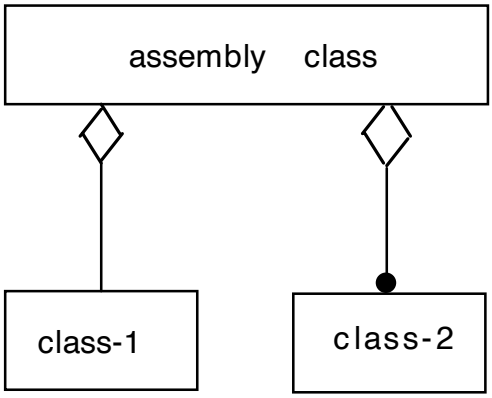

Figure A.4.- *Aggregation*

9. Appendix B: Non-functional requirements on updates and queries

The information in the MIAOW database is quite stable.

Very rarely is it necessary to add or remove information about a material.

Suppliers will accede to the database approximately every three months to make updates on the quantity available for material that they provide, whereas the indicative price of the materials will be updated more or less once per year.

The queries on the MIAOW database will be mainly devoted to kinds of selections such as an object of material or an object of supplier. Once the object has been retrieved, information about it will be selected by a successive query.

The most frequent information about a material required will be colour predominance and type of material. Other information quite often required will be name of material, use mode, place of origin, physical and mechanical tests and price range.

The availability of a certain material that a given supplier has (quantity, format available for sale) is another type of information that will be queried often. In most cases, the selection of this information will be done through a simple query on material.

Another very frequent kind of query will be the search for a material which is similar to the one retrieved previously and the retrieval of a material by colour similarity.