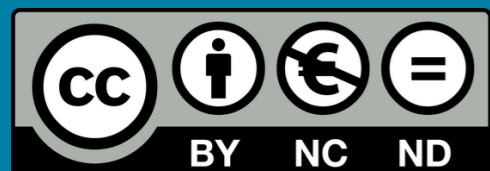


DATUM ACADEMY



MBDS course :

« From data bases to big data »

(7 lectures)

Professor Serge Miranda

Dept of Computer Science

University of Nice Sophia Antipolis (member of UCA)

Director of MBDS Master degree

(www.mbds-fr.org)



www.mbds-fr.org



Membre de UNIVERSITÉ CÔTE D'AZUR

Introduction to ODMG

(lecture 5)

Professor Serge Miranda

Departement of Computer Science

University of Nice Sophia Antipolis

Director of MBDS Master (www.mbds-fr.org)



www.mbds-fr.org



Membre de UNIVERSITÉ CÔTE D'AZUR

French version of this course since 2015

➤ « BD*2 : Des Bases de Données à Big Data »
sur la plateforme de Mooc FUN (www.fun-mooc.org) :

trailer du cours à
[http://www.canal-u.tv/video/universite de nice sophia antipolis/
mooc bd 2 des bases de donnees a big data le trailer.15548](http://www.canal-u.tv/video/universite_de_nice_sophia_antipolis/mooc_bd_2_des_bases_de_donnees_a_big_data_le_trailer.15548)

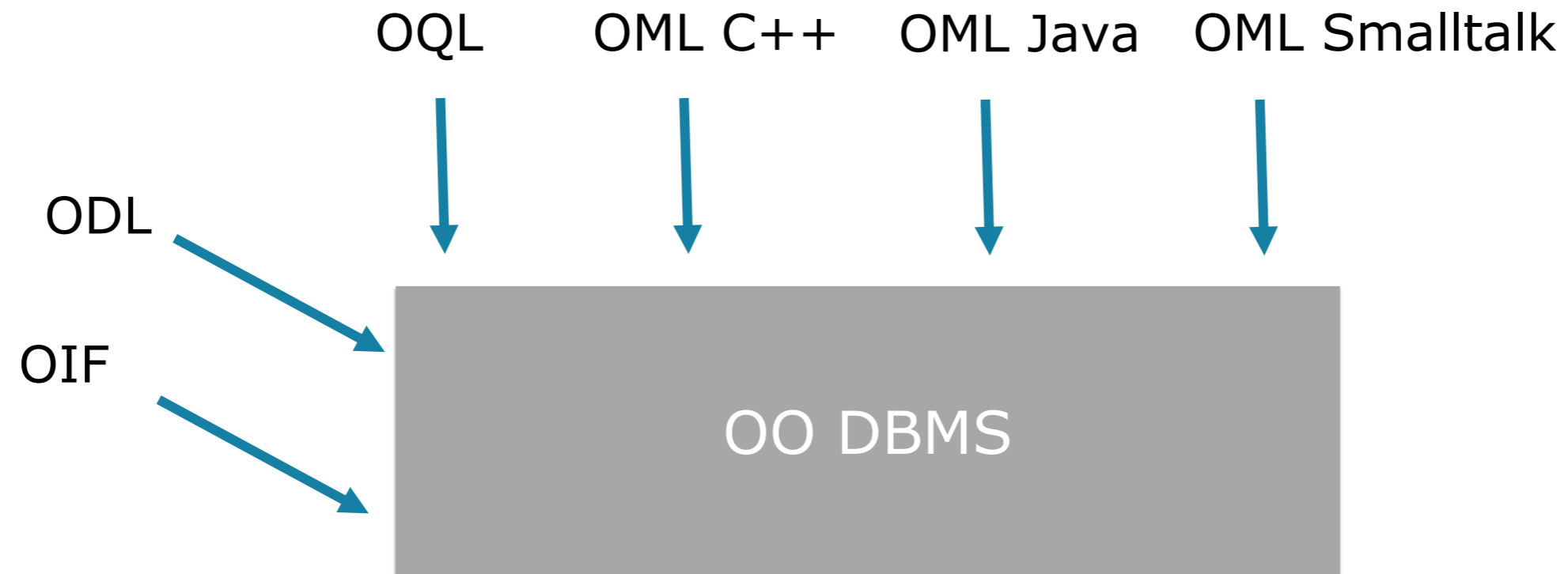
ODMG consortium

- Created in 1991 by Rick Cattell (SUN, Javasoft) as sub group of OMG (Object Management Group) with :
02 Tech., Objectivity, Object Design, Ontos, Versant
 - OMG : UML, CORBA, ...
- More than 50 companies in 2000
(Lucent, Lockheed, CA, Microsoft, Baan, ...)
- Bancilhon's Manifesto on Data Base of the future (1st Manifesto)
- **GOALS :**
 - Promoting OO (Object-Oriented) DBMS beyond FUD, before ...SQL3 !
 - Creating an Object standard for application portability

ODMG History

- 1st meeting at SUN with Rick Catell in Sept 91
- **ODMG 1.0 (1993)** : 5 Editors
 - ODL, OQL, C++ Interface, Smalltalk
- **ODMG 2.0 (1996)** : 10 Editors
(Poet, Lucent, Windward, American Man, Barry)
 - Java interface (Java Binding), Meta Model, OIF
- **ODMG 3.0 (2000)**
 - Java interface enrichment
 - Integration with OMG and X3H2 (SQL3)

ODMG 3.0 components



ODMG 3.0 components

- Object data model derived from OMG
- Specification language
 - ODL : Object Definition Language
 - OIF (object Interchange Format) : import/export
- Object Query Language : OQL (SQL like)
- OML (with links)
 - C++, SMALLTALK et JAVA

« Object-value » paradigm

➤ OBJECT CLASS ?

- OID for object (and PK for Class)
- « VALUE » for SET of OBJECTS (extension)
- ATTRIBUTES
- METHODS
- Bi-directional pointers among classes
(REF, INVERSE)
- Manipulation : NAVIGATION between classes with POINTERS
- Inheritance

« Object-value » paradigm

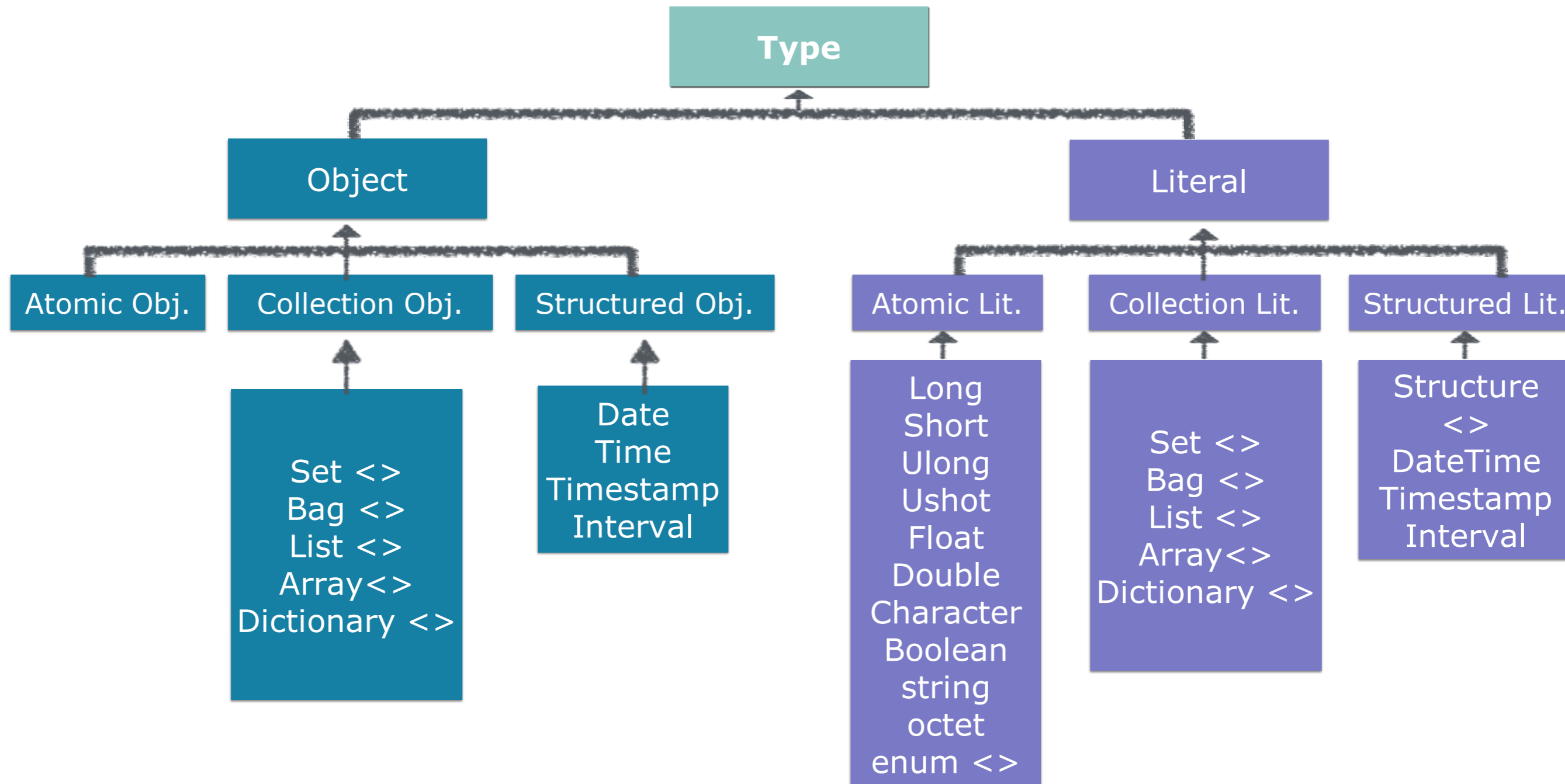
OMG Extension

- Standardized object model by OMG (« *Everything is OBJECT* »)
- Object data model supported by IDL (Interface Definition Language)
- Object-oriented DBs require TIPS extensions

ODMG and RICE properties

- R** * Multiple Inheritance (C++)
- I** * OID
- C** * STRUCTURE
- * COLLECTIONS :
SET, BAG, LIST, ARRAY, Dictionary
- * Bidirectional pointers . REF & INVERSE
- E** * « Types » with methods

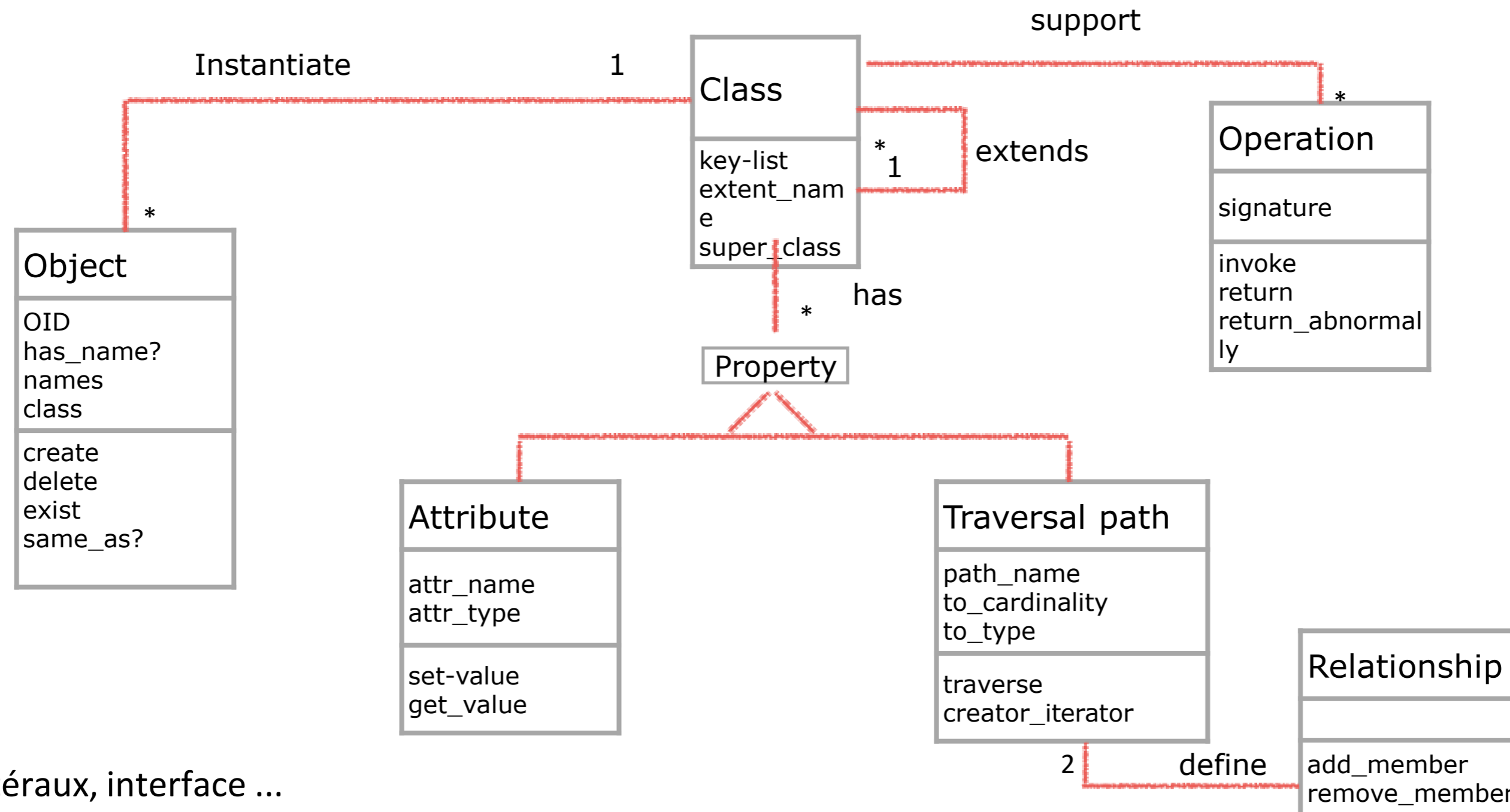
TYPE Hierarchy



« ODMG Meta model (*model on model*)

- Fundamentals
 - Object model description
- Meta Objects
 - Modules, Operations, Exceptions, values,
 - Properties (relationship & Attributes),
 - TypeDefinitions, Interfaces, Classes, Collections,
 - Specifiers, Operands

ODMG meta-model



+ Type, littéraux, interface ...

OQL (F.Bancilhon, Dasfaa 1995)

- **OQL : SQL- like interactive ad-hoc queries**
 - query optimization
 - Data independence
 - Embedded queries
 - TIPS support
 - data enrichment operators

- *Navigation and... « Surf » ☺*

1. Single-valued expression in **SELECT/WHERE** clauses

- *EXPRESSION* : X1.X2.X3...Xi with Xi attribute or pointer
- Such single-valued expression could be used to replace an attribute in a SQL SELECT/ WHERE clause

2. **Dependent Multivalued Collection in FROM clause**

- COLLECTION embedding in FROM clause

OQL query format

- **Select** [<type résultat>] (<**expression**> [, <expression>] ...)
From
 x in <**collection**>
 [, y in <collection>]...
Where <**expression**>

OQL by example

- Object schema:
 - Either in ODL (IDL extension),
 - Or C++/Smalltalk/Java

- With LINKS:
 - REF : persistent C++ pointer
 - INVERSE : pointer for referential integrity

ODL example

C++ binding

```
CLASS employee {  
    E# INT,  
    Name STRING,  
    Adress ADDRESS <other class>  
    //methods...}
```

ODL example

```

CLASS pilot : employee {
  NumberFH # INT,
  salary FLOAT,
  SET REF <flight> insures INVERSE is-insured-by
  //methods...}

```

```

CLASS flight {
  F# STRING,...
  REF <pilot> is-insured-by INVERSE insures,
  REF <plane> uses INVERSE is-used-in,.
  DC...}

```

ODL example (cont.)

```
CLASS plane : {  
p# INT,  
pname STRING,  
...
```

```
SET REF flight is-used-in INVERSE uses  
...}
```

Note :

- Links among classes (« *joins* ») are done EXCLUSIVELY by chasing pointers explicitly defined in the schema (generally in the FROM clause except if there are single-valued (and then could be in a WHERE expression))

OQL query example

- *Q1: What are the flight numbers ensured by a pilot whose name is SERGE?*

OQL example

- *Q1: What are the flight numbers ensured by a pilot whose name is SERGE?*

```

SELECT f.f#
FROM
    f IN flight    <IN collection>

    pl IN f.is-insured-by

WHERE pl.Name = 'Serge' ;

```


OQL Example

- *Q1: What are the flight numbers ensured by a pilot whose name is SERGE?*

```
Select f.f.#  
from f IN flight  
where f.is-insured-by.name='Serge';
```

- **<Note : Single-valued expression : X1.X2.X3>**

OQL example (cont')

- Q2 : *What are the names of planes driven by a pilot from NICE ?*

OQL example (cont')

- Q2 : *What are the names of planes driven by a pilot from NICE ?*

```
SELECT  p.pname
FROM    p in plane <in COLLECTION>
        f in p.is-used-by <in COLLECTION>
        pl in f.is-insured-by
WHERE   pl.adr = ` Nice ` ;
```

Exemple OQL

- Q2 : *What are the names of planes driven by a pilot from NICE ?*

```
SELECT  p.pname
FROM    p in plane <in COLLECTION>
        f in p.is-used-in<in COLLECTION>
WHERE   f.is-insured-by.adr = 'Nice'
```

Following EXERCICE
with solution next week



Exercices

- *Q3 : What are the names of pilots from NICE who insure flights from Nice with a plane located in Nice ?*

- *Q4 : What are the names of pilots who live in a city where an Airbus ('A300') is located ?*

Answers

➤ Q3 : *What are the names of pilots from NICE who insure flights from Nice with a plane located in Nice ?*

```
SELECT    pl.Name
From      pl in PILOT
          f in pl.insure
          p in f.use
```

Where *pl.adr = 'Nice' and p.loc = 'Nice' and f.DC= 'Nice';*

or

Where pl.adr= 'Nice' and f.use.loc= 'Nice ' and f.DC = 'Nice' (replacing p in f.use)

Answers

➤ Q4: *What are the names of pilots who live in a city where an Airbus ('A300') is located ?*

1. Impossible query with the existing schema (no ad hoc queries)
2. First : creation of pointers between PLANE and PILOT : LIVE pointer and its reverse IS-LOCATED-IN
3. Then

```

SELECT      pl.Name
From        pl in PILOT
            p in pl.LIVE <collection>
Where       p.pname = 'A300' ;
    
```


OQL

- Quantifier possible within FROM
 - universal (for all x in collection: predicate)
 - Example : « for all p in plane : a.cap<350 »
 - existence (exists x in collection..)
- GROUP BY allowed

Example with universal quantifier

- Q5 : *What are the names of the pilots who are driving **ALL** the A300 located in Nice ?*

Universal quantifier

- Q5 : *What are the names of the pilots who are driving **ALL** the A300 located in Nice ?*

```
Select pl.Name
```

```
From pl in pilot
```

```
    f in pl.insure
```

```
    For all p in f.use : p.pname = 'A300 ' and p.loc = 'Nice';
```

GROUP BY example

```
Select      f
from        f in flight
group by    (early : f.DT<9am, late : f.AT>6pm)
```

< 2 partitions for flights >

OQL summary

- Join EXCLUSIVELY expressed by pointer chasing (within an expression in WHERE clause if single-valued or within a collection of FROM clause if multi-valued)
- Complex language (with difficult implementation)
- note : *SQL improvement with ODMG 2.0*

"from c IN class1" is replaced by "from class1 c"

THESAURUS EXAMPLE

ODL

```
INTERFACE document  
(extent documents key doc#) : persistent  
{attribute integer doc#,  
  attribute string title ;  
  attribute string editor ;  
  attribute date Publirate);
```

Thesaurus example (cont')

- **relationship** LIST <author>
is-written-by **INVERSE** wrote;

- **relationship** SET <key-word>
includes **INVERSE** is-in;

Thesaurus example (cont')

➤ INTERFACE THESAURUS

- Relationship FATHERof <thesaurus> inverse SONof
- Relationship CLOSE...
- Relationship SYNONYMY...

➤ INTERFACE author : person (extent authors)

- relationship LIST<document> wrote
- INVERSE is-written-by

➤ INTERFACE person...

Thesaurus example

C++ binding

CLASS document

```
{ integer doc#,  
  string title;  
  string editor ;  
  date   publirate;  
  integer nb_pages;  
}
```

Thesaurus example

```
list Ref <author> is-written-by  
inverse wrote;  
set Ref <keyword>include  
inverse is-in;
```

```
CLASS author : person  
List Ref<document>wrote  
inverse is-written-by;
```

Thesaurus example

<thesaurus ; C++ binding >

CLASS person,...

CLASS keyword

{ string ident } ;

Set Ref <document> is-in

inverse include ;

Thesaurus example

List REF<keyword>is-synonymous-to
inverse is-synonymous-to;
Set REF< keyword>is-father-of
inverse is-son-of;
Set REF<keyword>is-son-of
inverse is-father-of;
Set REF<keyword>is-close-to
inverse is-close-to;

Thesaurus example (OQL)

Documents referred by « software »?

```
Select D
from
  D in documents
  K in D. include
Where K.ident = « software »;
```

Thesaurus example (OQL)

- *Documents concerning « software » whose title starts with « concepts »?*

Select d

From

 k in keywords

 ?...
 ?



SOLUTION NEXT

WEEK

Thesaurus (OQL)

Documents concerning « software » whose title starts with « concepts »?

Select d

From

k in keywords

c in k.is-close-to

s in k.is-synonymous-of

f in k.is-son-of

d in set (k.is-in, c.is-in, s.is-in, f.is-in)

Where k.ident = 'software" and d.title = Concepts%;

Thesaurus example (with Poet)

```
import COM.POET.odmg.*;
import COM.POET.odmg.collection.*;

class Document {
    int num_doc;
    String titre;
    String editeur;
    java.util.Date date_de_publication;
    int nb_pages;
    ListOfObject est_ecrit_par; // liste des auteurs
    SetOfObject contient;     // ensemble des mots clés
```

```
public Document(int num_doc, String titre, String editeur,
                java.util.Date date_de_publication, int
nb_pages){
    this.num_doc = num_doc;
    this.titre = titre;
    this.editeur = editeur;
    this.date_de_publication = date_de_publication;
    this.nb_pages = nb_pages;
}

public void ajouteAuteur(Auteur auteur){
    est_ecrit_par.add(auteur);
}

public void ajouteMotCle(MotCle motCle){
    contient.add(motCle);
}
}
```

Thesaurus example(Poet DBMS)

```
import COM.POET.odmg.*;
import COM.POET.odmg.collection.*;

class MotCle{
    String ident;
    SetOfObject est_dans;          // liste des documents qui
contient ce mot clé
    ListOfObject est_synonyme_de; // liste des synonymes de
ce mot clé
    SetOfObject est_pere_de;       // ensemble des mots clé
père
    SetOfObject est_fils_de;       // ensemble des mots clé
fils
    SetOfObject est_voisin_de;     // ensemble des mots clé
voisin
```

```
public MotCle(String ident){
    this.ident = ident;
}
public void ajouteSynonyme(MotCle synonyme){
    est_synonyme_de.add(synonyme);
}
public void ajouteVoisin(MotCle voisin){
    est_voisin_de.add(voisin);
}
public void ajoutePere(MotCle pere){
    est_pere_de.add(pere);
}
public void ajouteFils(MotCle fils){
    est_fils_de.add(fils);
}
public void ajouteDocument(Document document){
    est_dans.add(document);
}
}
```

Thesaurus example with POET (OQL)

➤ // document whose author is Serge Miranda

```
SELECT d
FROM
  k IN MocleExtent,
  c IN m.est_voisin_de,
  s IN m.est_synonyme_de,
  f IN m.est_fils_de,
  d IN SET(m.est_dans, c.est_dans, s.est_dans, f.est_dans),
  aut IN d.est_ecrit_par
WHERE m.ident = 'software' AND aut.nom = 'Miranda'
      AND aut.prenom='Serge';
```

Extra slides

OIF

- Specification language for
 - Importing
 - Exporting

➔ Exchanging data between different DBMS

OIF Exemple

Prenons par exemple la définition ODL :

```
Interface Person {
    attribute string Name;
    relationship Employer
        inverse Company : : Employees ;
    relationship Property
        inverse Company : : Owner ;
};

Interface Company {
    relationship set<Person> Employees
        inverse Person : : Employer ;
    relationship Person Owner
        inverse Person : : Property ;
};
```

Dans le fichier OIF les objets seront créés ainsi :

```
Personne1 Person{Name «Julio»}
Personne2 Person{«Pedro»}
Entreprise1 Company {Employees
{Personne1,Personne2}}
```

OIF (commands)

- For exporting objects :
odbdump <database name>

- For importing objects from several OIF files:
odblog <database name> <file 1>...<file n>

Transaction management

- Object Transaction created by Factory
 - begin() to open a transaction ;
 - commit() to validate transaction update ;
 - abort() to undo transaction updates;
 - checkpoint() = commit() + begin(), without releasing locks
 - join() to get transaction into thread ;
 - leave() to discard transaction from thread ;
 - To embed transactions
 - Concurrency control at object grain level;

Locking

- Implicit locking (object graph traversal) or Explicit locking (LOCK or TRY_LOCK in object interface) with level-3 SQL isolation level and ACID transactions
- Interface TransactionFactory
 - Transaction...
 - Transaction.....
- LOCK
 - Read
 - Write
 - Upgrade (before Read or Write to avoid livelock)

Other 2D Example with POET Dbms

2D Example

```
//classe qui représente un point
import COM.POET.odmg.*;
import COM.POET.odmg.collection.*;

class Point2D{
    int x;
    int y;
    Point2D(){
    }
    Point2D(int pi_x, int pi_y){
        x = pi_x;
        y = pi_y;
    }
    void move(int pi_x, int pi_y){
        x = pi_x;
        y = pi_y;
    }
    void moveRelative(int pi_x, int pi_y){
        x += pi_x;
        y += pi_y;
    }
}
```

2D Example

```
// Classe qui représente un polygone 2D
import COM.POET.odmg.*;
import COM.POET.odmg.collection.*;
class Polygone2D {
    SetOfObject points; //les points du polygone
    public Polygone2D() { //Constructeur
        points = new SetOfObject();
    }
    public void ajoutePoint(Point2D pr_point){ //rajouter un Point2D
        points.add(pr_point);
    }
    public int nombreDeCote() throws PolygoneException {
        if (points.size() > 2)
            return points.size();
        else
            throw new PolygoneException("Ce n'est pas un polygone");
    }
}
```

2D Example (OQL)

```
//combien d'hexagone existent-ils dans notre base
SELECT COUNT(*)
FROM Polygone2DExtent h
WHERE h.nombreDeCote = 6
```

```
//tous les points qui forment des octagones
SELECT p
FROM
    pol IN Polygone2DExtent,
    p IN pol.points
WHERE pol.nombreDeCote = 8
```

```
//tous les points qui sont dans des cercles d'un rayon supérieure à 10
SELECT p
FROM (SELECT c FROM Cercle2DExtent c WHERE c.rayon > 10) AS x,
    Point2DExtent p
WHERE
    x.contient(p)
```

CERTIFICATION of the course on

<https://www.DATUMACADEMY.COM>