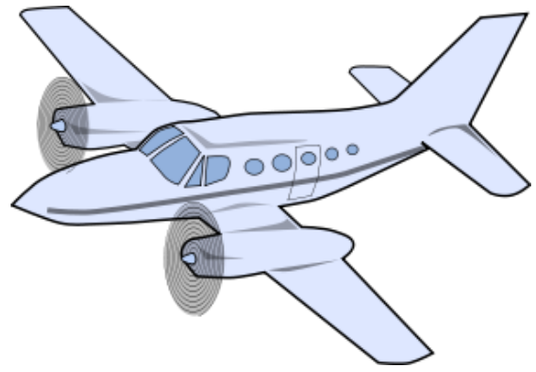


Introduction à l'ODMG

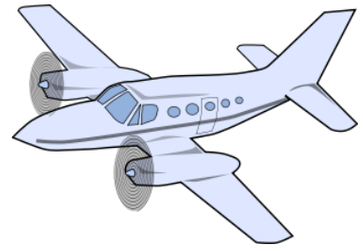
Professeur Serge Miranda
Département Informatique
Université de Nice Sophia Antipolis
Directeur du Master MBDS (www.mbds-fr.org)

Extrait du MOOC 2014-2015 « BD*2 : Des Bases de Données à Big Data » sur plateforme FUN :

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)

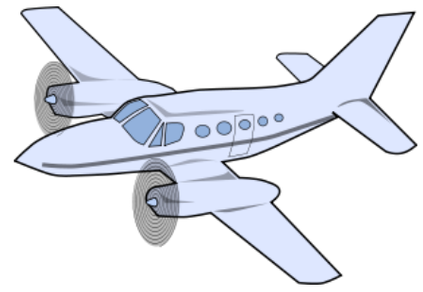
Consortium ODMG



- **Créé en 1991 par Rick Cattell (SUN, Javasoft) comme sous groupe de l' OMG (**Object Management Group) avec les éditeurs suivants: 02 Tech., Objectivity, Object Design, Ontos, Versant
- OMG : CORBA,…

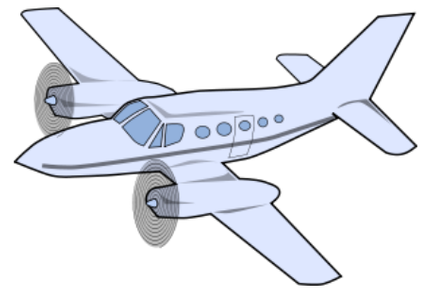
- plus de 50 organisations de l'industrie et de l'informatique en 2000 (Lucent, Lockheed, CA, Microsoft, Baan, ...)

- **Objectifs**:
  - Promouvoir les OO DBMS, *FUD* , au delà des niches et standard avant …SQL3 !
  - Créer un standard pour garantir l'indépendance entre OODBMS (*Portability)*



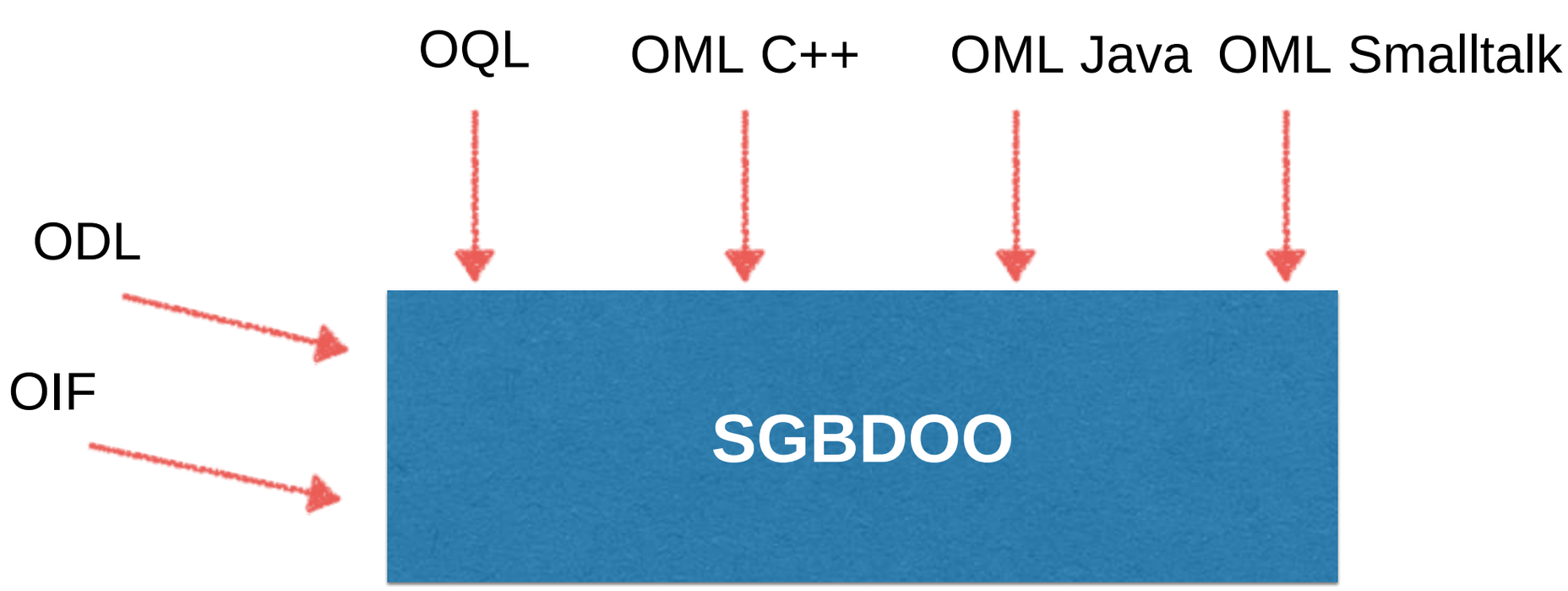
Historique du standard « OO »
Objectif « portabilité Code »

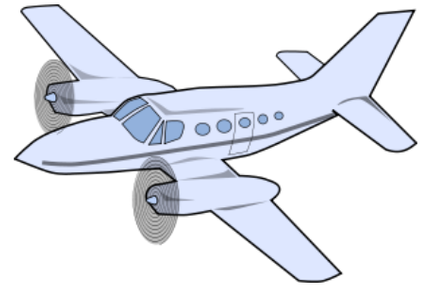
- Première réunion à l'initiative de Rick Catell chez Sun en Sept 91
- **ODMG 1.0 (1993) :** 5 Editeurs
  - ODL, OQL, Interface C++, Smalltalk
- **ODMG 2.0 (1996**) : 10 Editeurs (Poet, Lucent, Windward, American Man, Barry)
  - Interface Java (Java Binding) , Meta Model, OIF
- **ODMG 3.0 (2000)**
  - enrichissement interface Java
  - accent sur intégration avec OMG et X3H2 (SQL3)



Composants du standard ODMG 3.0

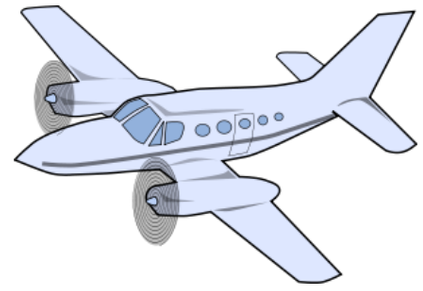
- Adaptation du modèle objet de l'OMG
- Interfaces d'accès à un SGBD OO





Composants du standard ODMG 3.0

- **Modèle de données Objet dérivé de l'OMG**
- Langages de spécifications
  - ODL : langage de définition d'objets
  - OIF (object Interchange Format) : import/export
- **Langage de requête : OQL** (avec retour sur base SQL)
- OML (avec Liaisons)
  - C++, SMALLTALK et JAVA



Modèle ODMG : modèle « *OBJET-VALEUR* »

CLASSE d'OBJETS (OBJECT CLASS) ?

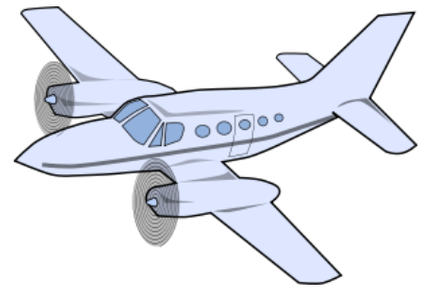
- OID invariant (pour « objet ») et **CP pour classe**
- « *VALUE* » de l ensemble des objets (*extension*)
- ATTRIBUTES
- METHODS
- **pointeurs <u>bidirectionnels</u> binaires entre classes**
  (REF, INVERSE)

Manipulation : NAVIGATION via des pointeurs
Héritage : d'état et de comportement

Modèle ODMG : modèle "**OBJET-VALEUR"**

Extension du modèle de l'OMG

- l'OMG a proposé un modèle standard pour les objets
- le modèle est supporté par le langage IDL (def. interface)
- les BD objets nécessitent des adaptations/extensions TIPS
  - instances de classes, collections, associations
  - persistance, transactions



ODMG et propriétés RICE

R * HERITAGE MULTIPLE (C++)

I * OID

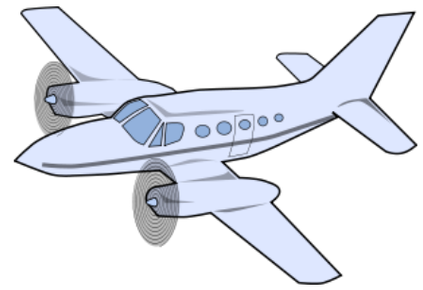
C * STRUCTURE

* COLLECTIONS :

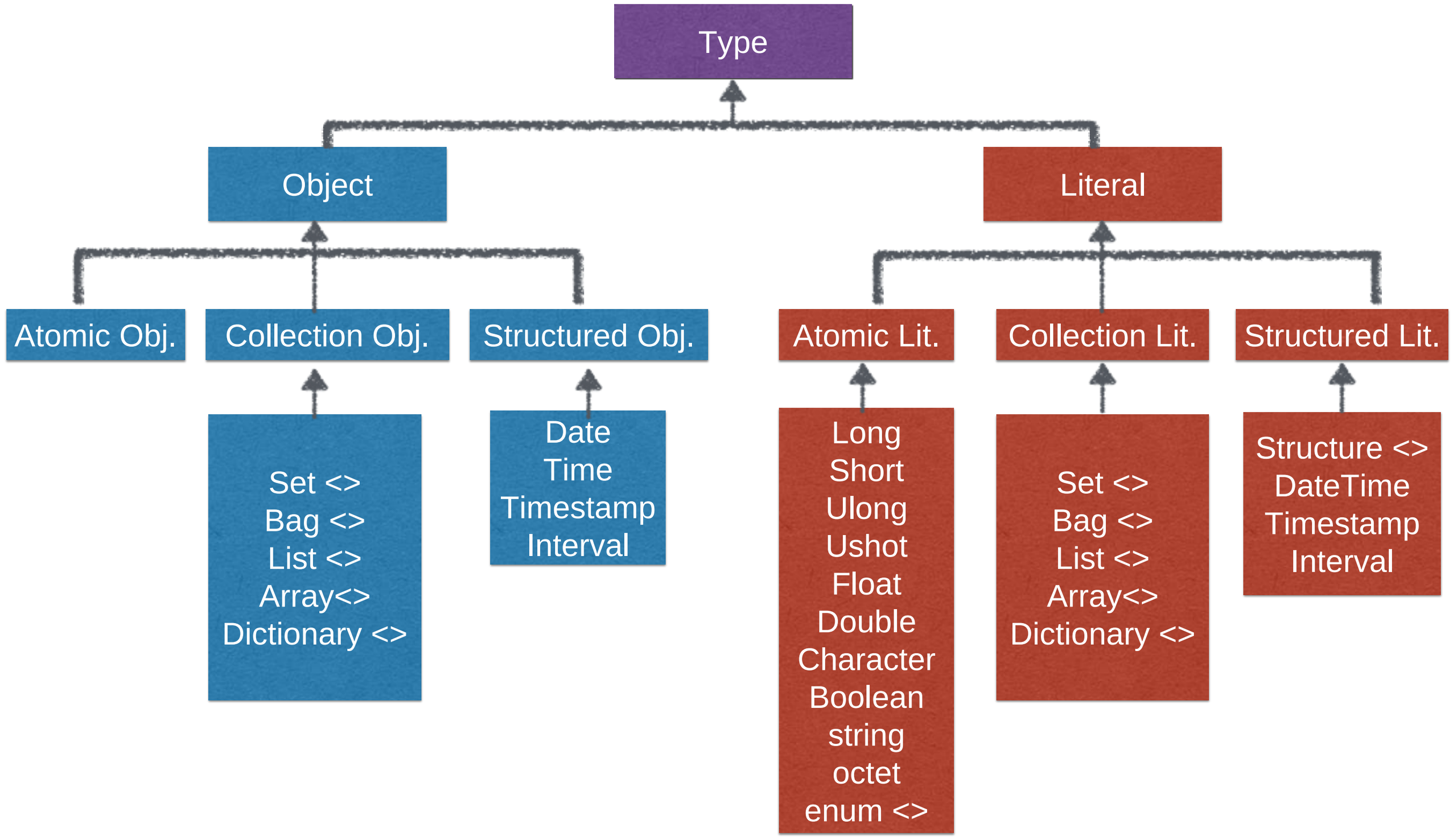
SET, BAG, LIST, ARRAY, Dictionary

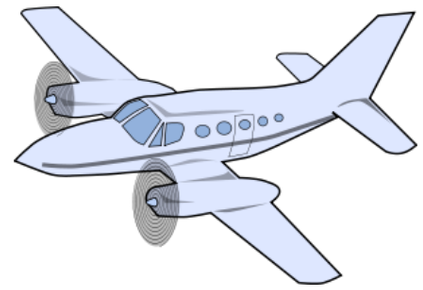
* pointeurs bidirect. *REF* et *INVERSE*

E * « Types » avec méthodes



Hiérarchie de « Types »

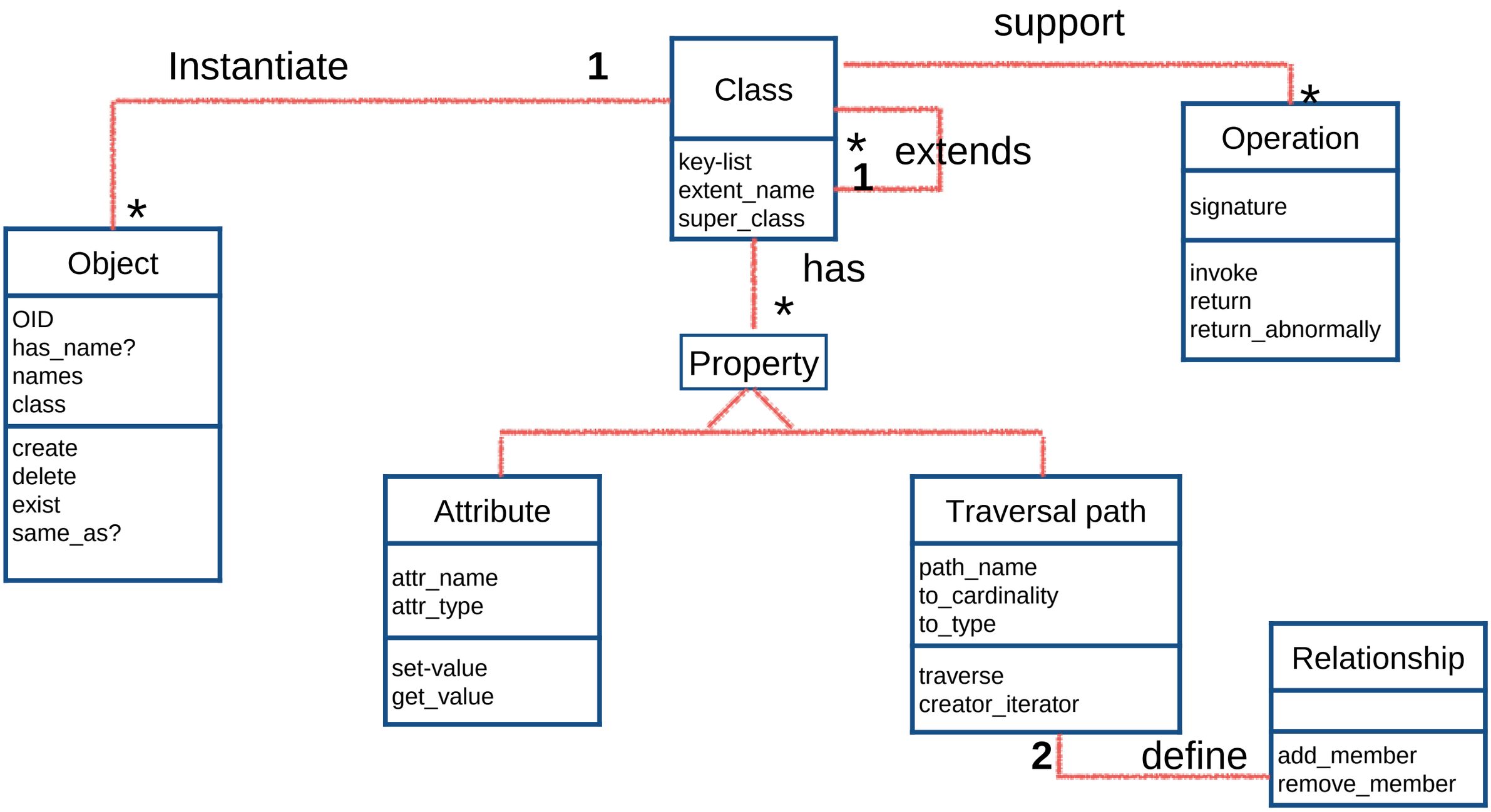




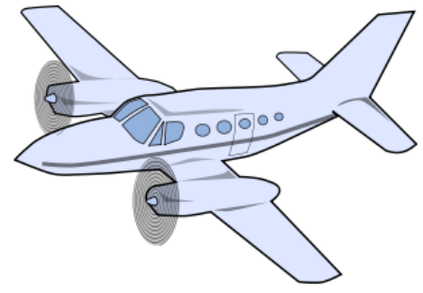
« Meta MODELE » de l'ODMG

- Principe de Base
  - Description du modèle Objet
- Les Meta Objets
  - Modules, Operations, Exceptions, Constants,
  - **Properties (relationship et Attributes),**
  - TypeDefinitions, **Interfaces, Classes,** Collections,
  - Specifiers, Operands

Méta-modèle du modèle ODMG



+ Type, littéraux, interface ...



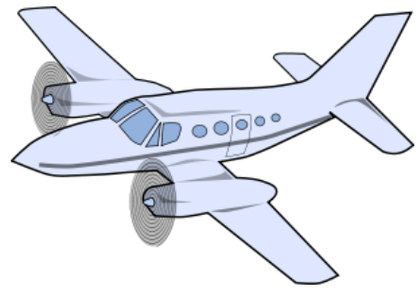
OQL

(F.Bancilhon, Dasfaa 95)

requêtes ad-hoc interactives type « SQL »

- simplification de programmation via des requêtes imbriquées
- optimisation de requêtes
- indépendance de données
- opérateurs d'enrichissement des données
- Support TIPS

Navigation et .. "Surf" via des expressions de chemin et de suivi de pointeurs (tout en préservant les interrogations associatives)



OQL
(nouveautés « SQL »)

1.Expression de chemin mono-valuée *dans SELECT/WHERE*

- Séquence d'attributs ou associations (« relationship ») mono-valués de la forme X1.X2…Xn telle que chaque Xi à l'exception du dernier contienne une référence à un objet ou un litéral unique sur lequel le suivant s'applique.
- Utilisable en place d'un attribut SQL dans SELECT ou WHERE

2. Collection dépendante <Multivaluée> dans FROM

- Collection obtenue à partir d'un objet, soit parce qu'elle est imbriquée dans l'objet ou pointée par l'objet.
- Imbrication de COLLECTIONS dans FROM

Format des Requêtes

Forme générale d'une requête

- Bloc « SELECT » étendu :

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

Type résultat

- automatiquement inféré par le SGBD
- toute collection est possible (bag par défaut)
- création possible d’ objets en résultats

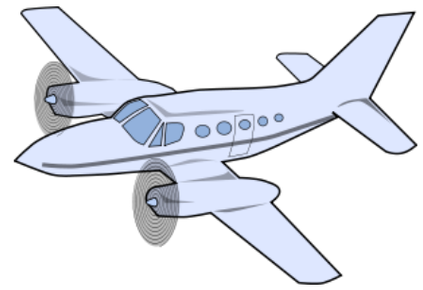
Syntaxe très libre, fort contrôle de type

Concepts de base par l'exemple

Définition du schéma Objet :

- **soit en ODL** (Extension IDL), soit OIF
- **soit en C++/Smalltalk/java**

- **avec les** LIENS :
- REF : pointeur C++ persistant
- INVERSE : extension pour intégrité référentielle



Exemple ODL

Liaison C++

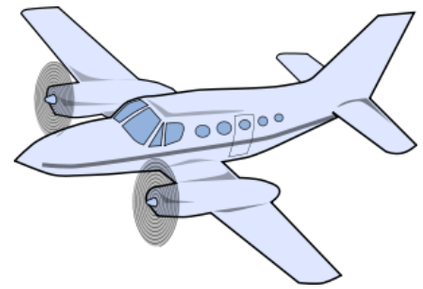
CLASS employe {

E# INT,

Nom STRING,

Adresse ADDRESS <*autre classe*>

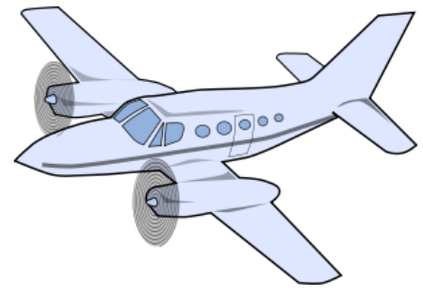
//méthodes...}



Exemple ODL

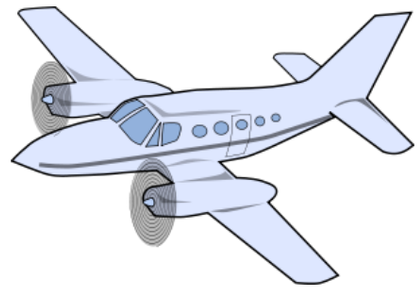
CLASS pilote : employe {
Nbre-H-Vol# INT,
salaire FLOAT,
SET REF <vol> assure INVERSE est-assuré_par
//méthodes...}

CLASS vol {
V# STRING,…
REF <pilote> est_assuré_par INVERSE assure,
REF <avion> utilise INVERSE est_utilisé_dans,.
VD...}



Exemple ODL

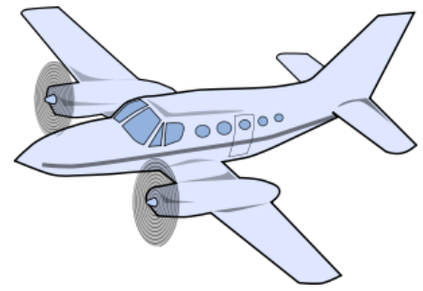
CLASS avion : {
av# INT,
avnom STRING,
…
SET REF <vol> est_utilisé_dans INVERSE utilise
…}



Notes :

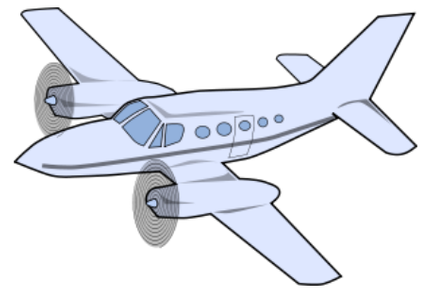
- *FAIRE TABLEAU DES LIENS ENTRE CLASSES pour traiter les questions*

- *Les liens entre classes (« jointures ») se feront* ***EXCLUSIVEMENT*** *par suivi de pointeurs (bidirectionnels) définis* ***EXPLICITEMENT*** *dans le schéma (par défaut dans clauses FROM imbriquées*)



Exemple OQL

Quels sont les numéros des vols assurés par un pilote de nom 'Serge' ?



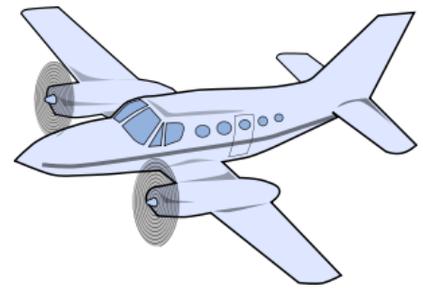
Exemple OQL

Quels sont les numéros des vols assurés par un pilote de nom 'Serge' ?

1) SELECT v.v#
   FROM
      v IN vol <IN collection>

      p IN v.est_assure_par

   WHERE p.Nom = 'Serge' ;

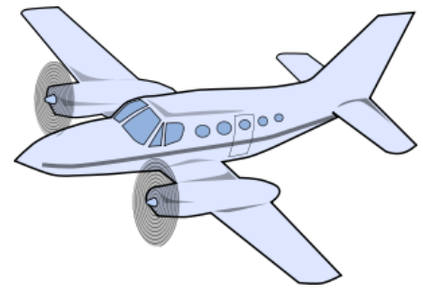


Exemple OQL

Quels sont les numéros des vols assurés par un pilote de nom 'Serge' ?

```
Select v.v#
from v IN vol
where v.est_assure_par.Nom=‘Serge’;
```

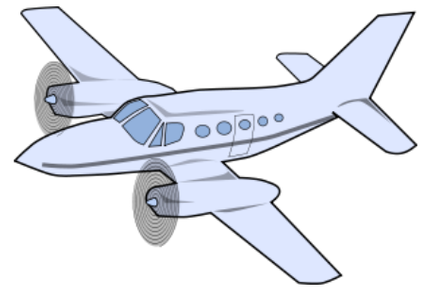
<Expression monovaluée X1.X2.X3>



EXERCICE
Exemple OQL

Quels sont les noms des avions conduits par un pilote Niçois ?

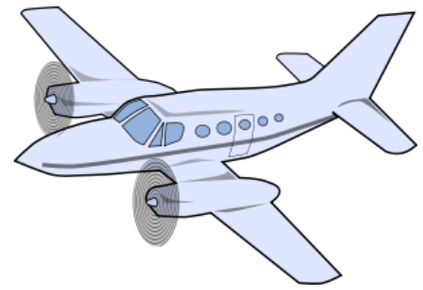
<Parcours d’ associations multivaluées en utilisant des collections dépendantes>



Exemple OQL

Quels sont les noms des avions conduits par un pilote Niçois ?

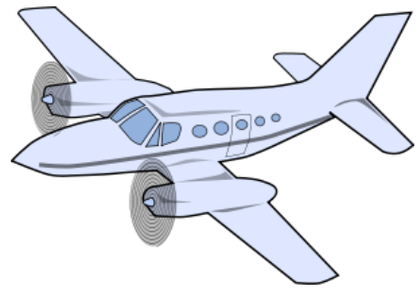
```
SELECT      a.avnom
FROM     a in avion <in COLLECTION>
             v in a.est_utilisé_dans <in COLLECTION>
             p in v.est_assuré_par
WHERE p.adr = ‘ Nice ’ ;
```



Exemple OQL

Quels sont les noms des avions conduits par un pilote Niçois ?

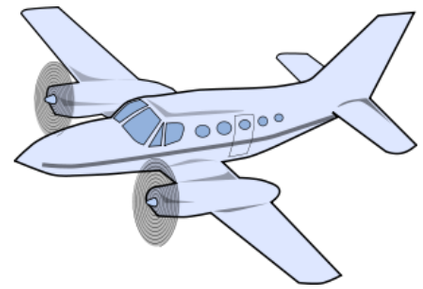
SELECT a.avnom
FROM a in avion *<in COLLECTION>*
v in a.est_utilisé_dans *<in COLLECTION>*
WHERE v.est_assuré_par.adr= ‘Nice’
<Expression>



Exercices

Q1 : Noms des pilotes Niçois qui assurent un vol au départ de Nice avec un avion localisé à Nice ?

Q2 :Noms des pilotes qui habitent dans la ville de localisation d’un Airbus ?



Réponses

Q1

```
SELECT p.plnom
From     p in PILOTE
         v in p.assure
         a in v.utilise
Where    p.adr = ‘Nice’ and a.loc = ‘Nice’ and v.VD= ‘Nice’;
```

(ou Where p.adr= ‘Nice’ and v.utilise.loc= ‘Nice ‘ and v.VD = ‘Nice’ ;

Réponses

Q2

1) Créer liens multivalués HABITE inverse EST-LOCALISE-DANS entre Pilote et Avion

2) SELECT p.plnom
From p in PILOTE
a in p.HABITE <collection>
Where a.avnom = ‘Airbus’ ;

OQL

Expression de chemin monovaluée à la place d'un attribut SQL de la forme X1.X2.X3 : chaque Xi contient une référence à un objet unique

SELECT imbriqués possibles,

méthodes dans where ou select

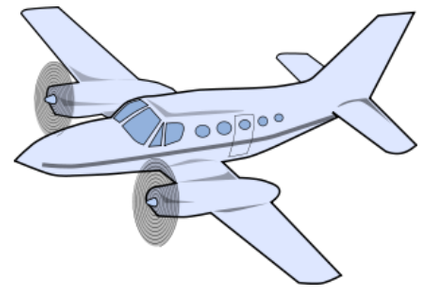
Quantificateur dans FROM

- universel (for all x in collection: prédicat)
  - Exemple : « for all a in Avions: a.cap<350 »
- existentiel (exists x in collection..)

GROUP BY possible

Exemple quantificateur universel

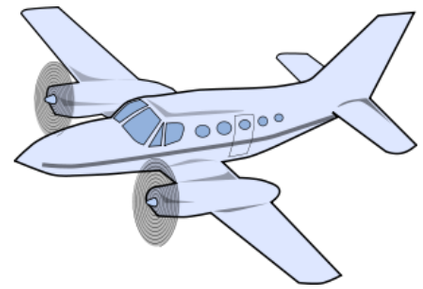
Quels sont les noms des pilotes qui conduisent TOUS les Airbus localisés à Nice ?



Exemple quantificateur universel

Quels sont les noms des pilotes qui conduisent TOUS les Airbus localisés à Nice ?

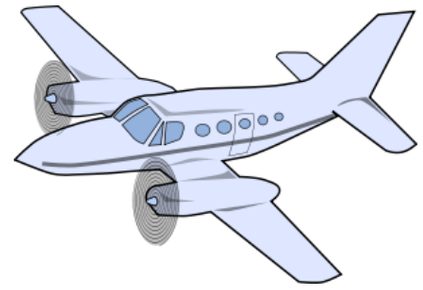
Select p.plnom
From p in pilote
V in p.assure
For all a in V.utilise : a.avnom = ‘airbus ’ and a.loc = ‘Nice’;



Exemple GROUP BY

Partitionner les instances de la classe VOL en 2 groupes : les vols avant 9H (nom partition : TOT) et les vols après 18H (nom : TARD).

```
Select v
from v in VOL
group by (tot : v.HD<9, tard : v.HD>18)
```



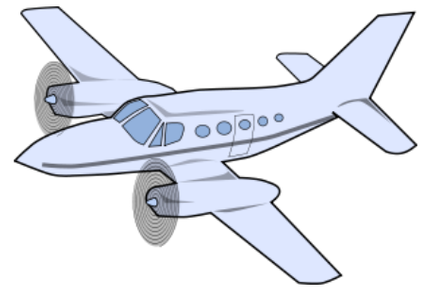
OQL

Join exprimé **EXCLUSIVEMENT** par une expression de chemin sur lien/PTR prédéfini de manière symétrique (notation '.') : "le programmeur redevient un navigateur (un Surfeur ?)

Langage de requête très COMPLET/COMPLEXE ‘double paradigme) difficile à implanter

note : amélioration cosmétique SQLà partir de la version ODMG 2.0

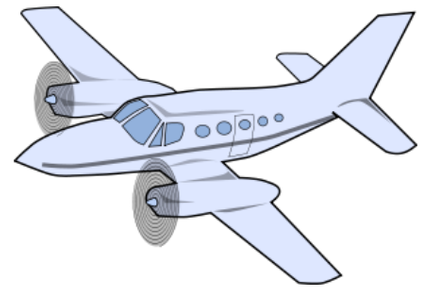
"from c IN class1" remplacé par
"from class1 c"



Exemple Thésaurus Complet

ODL

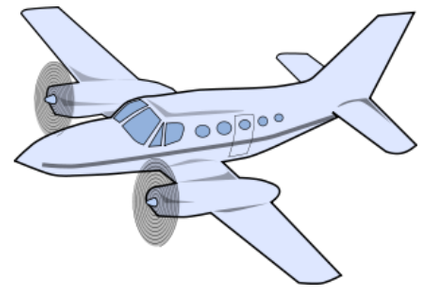
INTERFACE document
(extent documents
key doc#) : persistent
{attribute integer doc#,
  attribute string titre ;
  attribute string editeur ;
  attribute date date-publi ;
attribute integer nb_pages ;
}



Exemple Thésaurus

relationship LIST<auteur>
est_ecrit_par INVERSE a_ecrit;

relationship SET <mot_cle> contient
INVERSE est_dans ;

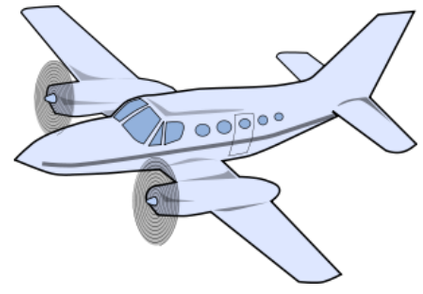


Exemple Thésaurus

INTERFACE THESAURUS
Relationship PEREDE <thesaurus> inverse FILSDE
Relationship VOISIN …
Relationship SYNONYMIE….

INTERFACE auteur: personne (extent auteurs)
relationship LIST<document> a_ecrit
INVERSE est_ecrit_par

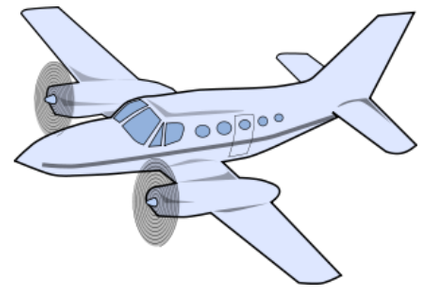
INTERFACE personne ….



Exemple Thésaurus

Liaison C++

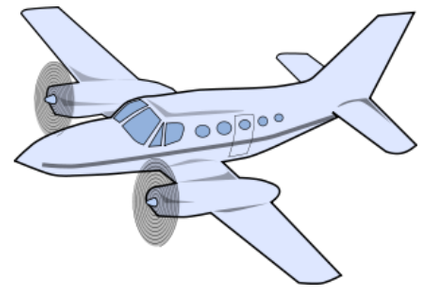
```
CLASS document
{ integer  doc#,
  string titre;
  string editeur ;
  date    date_de_publi;
  integer nb_pages;
}
```



Exemple Thésaurus

list Ref <auteur> a_ete_ecrit_par
inverse a_ecrit ;
set Ref <mot_cle>contient
inverse est_dans;

CLASS auteur : personne
List Ref<document>a_ecrit
inverse a-ete_ecrit_par ;



Exemple Thésaurus

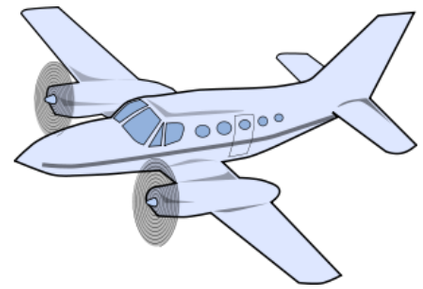
<thesaurus ; liaison C++ >

CLASS personne,…

CLASS mot_cle
{ string ident };
Set Ref <document> est_dans
inverse contient;

Exemple Thésaurus

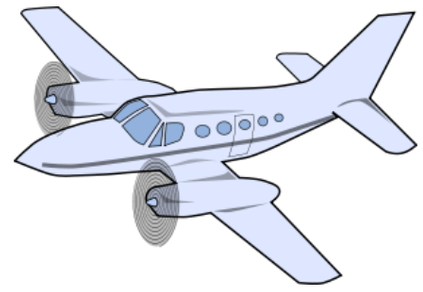
List REF<mot_cle>est_synonyme_de
inverse est_synonyme_de;
Set REF< mot_cle>est_pere_de
inverse est_fils_de;
Set REF<mot_cle>est_fils_de
inverse est_pere_de;
Set REF<mot_cle>est_voisin_de
inverse est_voisin_de;



Exemple Thésaurus (OQL)

Documents indexés par "software"?

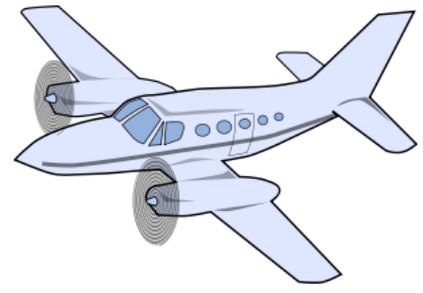
```
Select D
 from
      D in documents
      M in D. contient
Where M.ident = « software »;
```



Exemple Thésaurus (OQL)

Documents concernant le "software" dont le titre commence par "concepts" ?

```
Select d
From
      m in mots_cle
     ?...
```



Exemple Thésaurus (OQL)

Documents concernant le "software" dont le titre commence par "concepts" ?

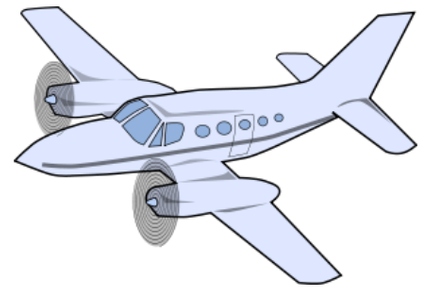
```
    Select d
From
    m in mots_cle
    c in m.est_voisin_de
    s in m.est_synonyme_de
    f in m.est_fils_de
    d in set (m.est_dans, c.est_dans, s.est_dans, f.est_dans)
Where m.ident = 'software" and d.titre = Concepts%;
```

Illustration du Thésaurus
(avec le SGBD 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);
    }
}
```

Exemple Thésaurus (SGBD Poet)

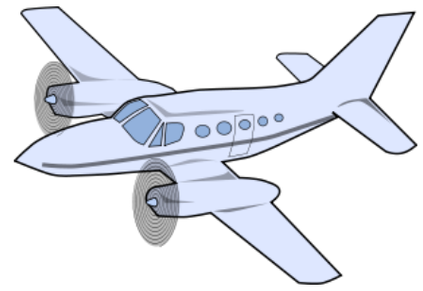
```
import COM.POET.odmg.*;
import COM.POET.odmg.collection.*;

class MotCle{
    String ident;
    SetOfObject est_dans;          // liste des documents qui
  contiennt 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);
    }
```



Exemple Thésaurus avec Poet (OQL)

```
// tous les documents concernant le software et
// dont l'auteur est Serge Miranda

SELECT d
FROM
    m IN MotCleExtent,
    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

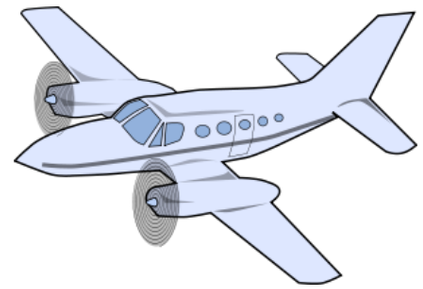


OIF

Langage de spécifications pour

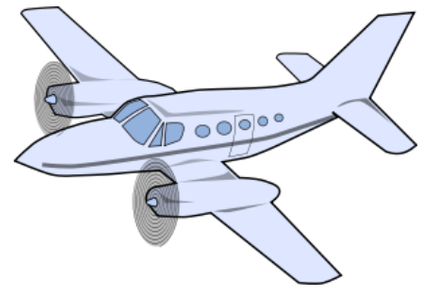
- Importer
- Exporter

- Echanger des objets entre 2 bases
- Fournir une documentation
- Piloter les « tests suites »



OIF

- OIF doit supporter tous les états des objets d ’une BD ODMG
- OIF est un langage de spécification
- OIF respecte au mieux les standards ANSI et STEP
- Mot clé OIF : *type*, *attribut*, et *identifiant* d’une relation



OIF Exemple

Prenons par exemple la définition ODL :

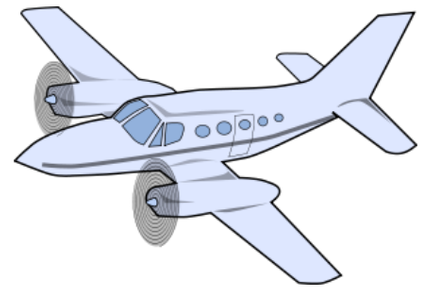
```
Interface Person {
 attribute string Name;
 relationship Employer
     inverse Company : : Employees ;
 relationship Property
     inverse Company : :Owner ;
} ;
Interface Company {
 relationship set<Person> Employees
     inverse Person : : Employer ;
 relationship Person Owner
     inverse Person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

Dans le fichier OIF les objets seront créés ainsi :

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

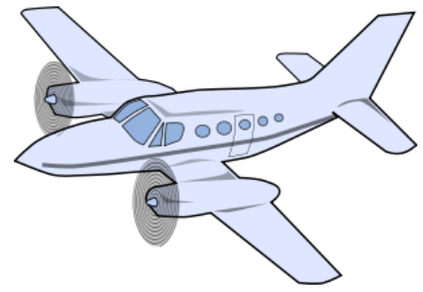
OIF (commandes)

- Pour exporter les objets d ’une base : odbdump <database name>
- Pour IMporter des objets depuis un ou plusieurs fichiers OIF : odbload <database name> <file 1>…<file n>



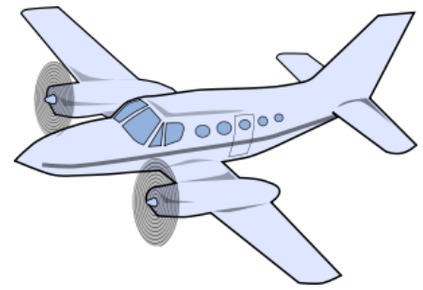
INTEGRATION à C++, Java, Smalltalk

- Implémentation du modèle abstrait
  - mapping des concepts
  - mapping des types
  - mapping des collections
- Nécessité d'adapter le modèle
  - certains concepts n’existent pas dans le langage
    - **interface** en C++ ==> **classe**
    - **association** en C++ et Java ==> attributs roles de type **Ref** <T>
    - clés ==> pas de clés !
- Nécessité d’intégrer OQL



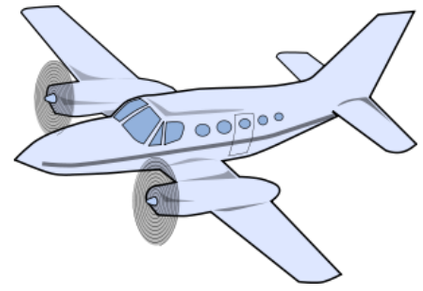
Java Binding
(OML Java)

- Système de Types de Données Uniques
- Syntaxe de Java à respecter
- Gestion automatique de Stockage (*Persistence By Reachability*)



JAVA ODL

- Types Simples d’ODL -> types simples Java
- Types Complexes (Collections) ->
  - Interfaces des Collections (Dset, Dbag, etc. de Java 1.2)
- Pas de Gestion Automatique des LIENS BIDIRECTIONNELS



JAVA OML/OQL

JAVA OML

- Déclaration Opérations en Java
- Des Classes Pour Database, Transaction, Collections

Java OQL

- Méthode query() de DCollection
- Classe OQLQuery

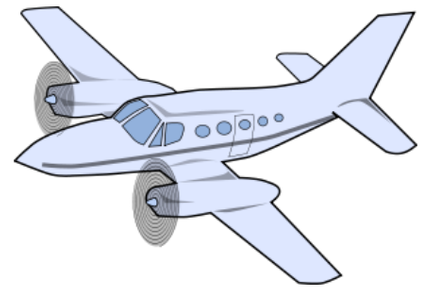
JAVA OML/OQL

- Persistance par atteignabilité (*reachability*)
  - classes connues du SGBDOO
  - objets capables de persister
  - nommage par objets «database»
    - -opérations bind, unbind, lookup
    - -tout objet nommé est racine de persistance
  - tout objet référencé par un objet persistant est persistant
- Mapping des types
- Package collections ODMG :set, bag, list, varray : collection

Gestion de transactions

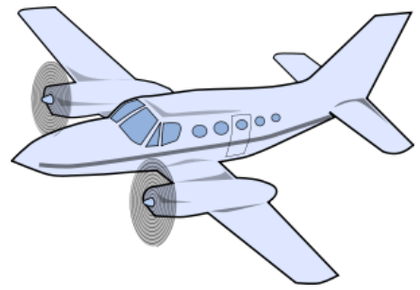
Objet Transaction créé par Factory

- begin() pour ouvrir une transaction ;
- commit() pour valider les mises à jour de la transaction ;
- abort() pour défaire les mises à jour de la transaction ;
- checkpoint() = commit() + begin(), sans relâcher verrous
- join() pour récupérer l'objet transaction dans la thread ;
- leave() pour dissocier un objet transaction de la thread ;
- possibilités d'imbriquer des transactions
- contrôle de concurrence niveau objet (explicite ou défaut) ;



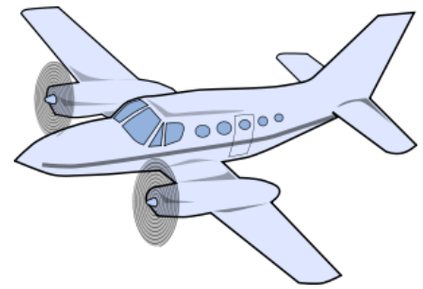
Verrouillage

- Verrouillage **Implicite** (pendant traversée graphe objets) ou **Explicite** (LOCK ou TRY_LOCK dans Interface Objet) avec **isolation niveau 3 de SQL2** (qui évite lectures sales, fantômes et lectures non reproductibles) et **transactions ACID**
  - **Interface** TransactionFactory
    - Transaction…
    - Transaction.....
- LOCK
  - Read
  - Write
  - Upgrade (avant Read ou Write pour éviter livelock)



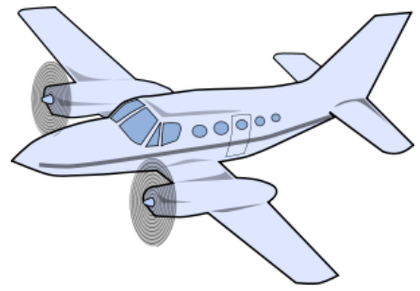
Exemple

- Exemple 2D avec SGBD POET



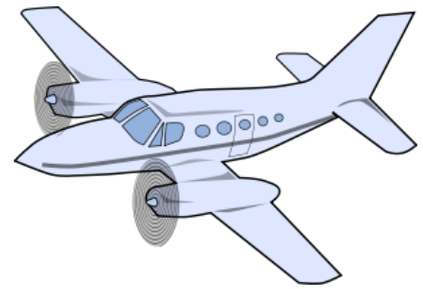
Exemple 2D

```
//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;
    }
}
```



Exemple 2D

```
// 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");
    }
}
```



Exemple 2D (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 point qui sont dans des cercles d'un rayon superieure à 10
SELECT p
FROM (SELECT c FROM Cercle2DExtent c WHERE c.rayon > 10) AS x,
     Point2DExtent p
WHERE
    x.contient(p)
```