ODMG
The Object Database Standard

Carlo Combi

Dipartimento di Matematica e Informatica

Universita' degli Studi di Udine

ODMG
The Object Database Standard

- Introduzione
- Il modello a oggetti ODMG Object Model (OM)
- Object Specification Language: ODL (Object Definition Language) e OIF (Object Interchange Format)
- OQL: Object Query Language
- C++ Binding
- Smalltalk Binding
- Java Binding
- Confronto con il modello a oggetti OMG
- ODBMS nell'ambiente OMG ORB

Lo standard ODMG: Scopi

- Permettere di progettare applicazioni che possano fondarsi su differenti DBMS orientati agli oggetti.
  - Protezione degli investimenti in software da parte degli utenti;
  - riduzione della dipendenza da un singolo produttore di DBMS;
  - incoraggiamento ad uno sviluppo competitivo.

ODMG: Object Database Management Group

- Mercato degli OODBMS in crescita dall'inizio degli anni 90 (da meno di 20 milioni di dollari nel 91, a circa 115 milioni di dollari nel 1996, con una previsione di 1.6 bilioni di dollari nel 2000);

- Associazione, fondata nel 1991, dei produttori di OODBMS: SunSoft, Sybase, AT&T Bell Labs, Poet Software, GemStone Systems, Versant Object Technology, Objectivity, Hewlett Packard, Unidata, MITRE, ONTOS, ...

Lo standard ODMG
Definizioni di base

- I sistemi di basi di dati ad oggetti estendono le funzionalita' dei linguaggi di programmazione orientati agli oggetti (C++, Smalltalk, Java) al fine di fornire una completa capacita' di programmazione delle basi di dati.

- L'applicazione e la base di dati condividono lo stesso modello dei dati: meno codice, strutture dati piu' naturali, miglior manutenibilita' e riusabilita' del codice.

Lo standard ODMG
Definizioni di base

- Oltre alle tradizionali caratteristiche dei sistemi di basi di dati (controllo della concorrenza, interrogazioni, salvataggio e ripristino) le basi di dati ad oggetti forniscono ulteriori capacita', non riscontrabili nelle basi di dati relazionali: evoluzione dinamica dello schema ed aggiornamento delle istanze, interrogazioni navigazionali, versioning degli oggetti, supporto alla base di dati distribuito. La maggior parte delle basi di dati orientate agli oggetti possono essere interrogate con un SQL standard e sono in grado di creare oggetti sulla base di dati provenienti da basi di dati relazionali.

Lo standard ODMG
Relazioni con altri progetti di standard

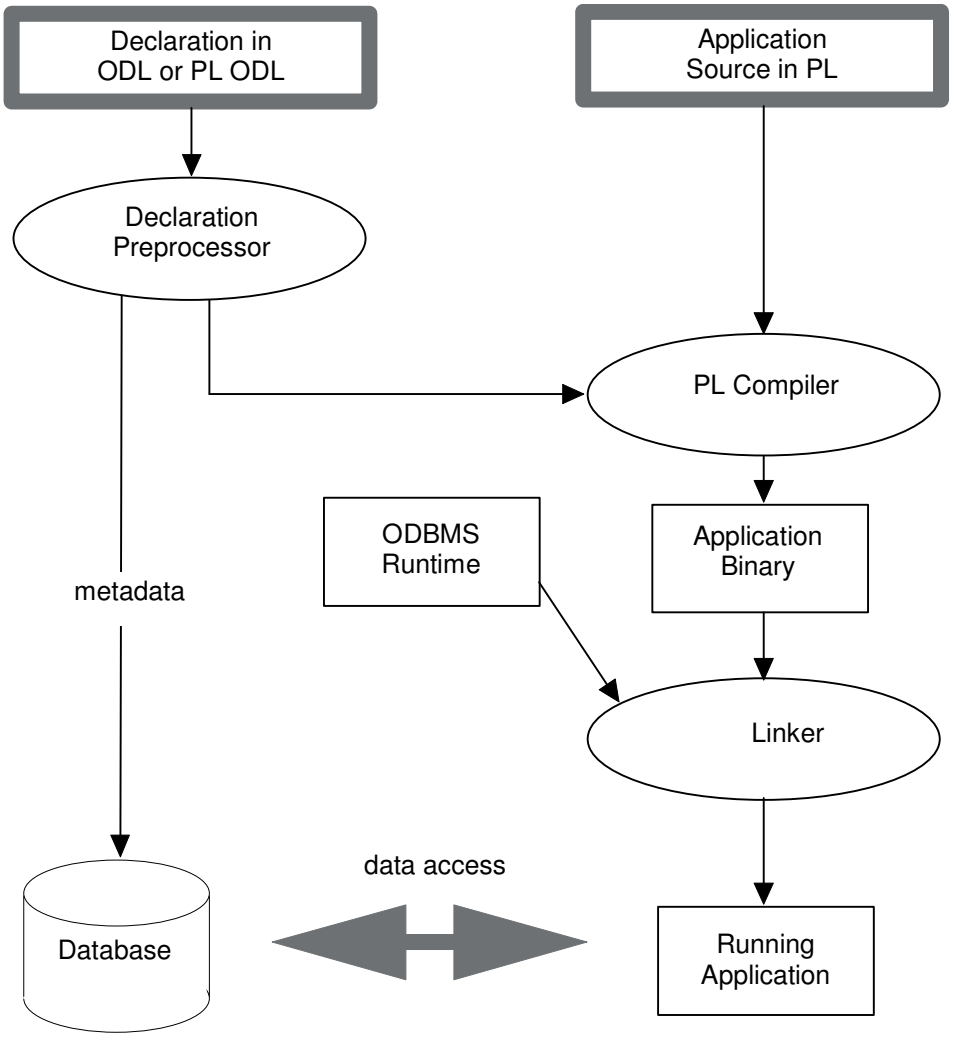
- ODMG si fonda sugli sforzi di standard in tre differenti aree, al fine di fornire una concisa specifica per progettare applicazioni di basi di dati con linguaggi di programmazione orientati agli oggetti:
  - basi di dati (SQL)
  - sistemi orientati agli oggetti (OMG)
  - linguaggi di programmazione orientati agli oggetti (C++, Smalltalk e Java).

- ODMG non e' affiliato a organizzazioni nazionali o internazionali di standardizzazione (ANSI, OSI). Relazioni con i comitati ANSI X3H2 (SQL), X3J16 (C++), X3J20 (Smalltalk) e X3H7 (Object Model) come con l'Object Management Group (OMG).

Lo standard ODMG
I componenti

- Object Model, basato sull'OMG object model;
- Object Definition Language, basato sull'OMG IDL (Interface Definition Language);
- Object Query Language, fondato su SQL, per l'aggiornamento e l'interrogazione di basi di dati;
- C++ Binding: definizione di un C++ OML (Object Manipulation Language), di una sintassi C++ per ODL e di meccanismi per invocare OQL e per operazioni su basi di dati e transazioni;
- Smalltalk Binding: legami e mapping fra ODL e Smalltalk (basato sull'OMG Smalltalk Binding);
- Java Binding: legami e mapping fra ODL e Java.

ODMG
Architettura di un ODBMS



Il modello a oggetti
ODMG Object Model (OM)

L' ODMG Object Model specifica i costrutti supportati da un ODBMS:

- le primitive di base per la modellazione delle informazioni sono l'*oggetto* (object) e il *letterale* (literal); ogni oggetto ha un identificatore unico. Un letterale non ha identificatore.

- lo stato di un oggetto e' definito dai valori assegnati per un insieme di *proprieta'* (property).

    - Le proprieta' possono consistere in *attributi* (attribute) dell'oggetto stesso o in *relazioni* (relationships) fra l'oggetto considerato ed altri oggetti. I valori delle proprieta' di un oggetto usualmente cambiano nel tempo.

Il modello a oggetti ODMG Object Model (OM)

- il comportamento di un oggetto e' definito attraverso un insieme di operazioni che possono essere eseguite sull'oggetto o da parte di esso.

- oggetti e letterali sono categorizzati attraverso il loro *tipo* (type). Tutti gli elementi di un dato tipo hanno la stessa gamma di *stati* (stesse proprieta') e un *comportamento* (behavior) comune (lo stesso insieme di operazioni). Un oggetto e' spesso detto instanza (instance) del suo tipo.

- una *base di dati* immagazzina oggetti, abilitandoli ad essere condivisi da piu' utenti ed applicazioni. Una base di dati si fonda su uno schema, definito in ODL, e contiene istanze del tipi definiti attraverso lo schema. Lo schema (logico) della base di dati e' detto anche application's object model.

Il modello a oggetti
ODMG Object Model (OM)
Tipi e classi

- Un *tipo* e' composto da una *specifica esterna* e da una o piu' *implementazioni*.

- *Specifica esterna*

  - *interface*: definisce le operazioni (*operation*) che possono essere invocate sull'oggetto di quel tipo.

  - *class*: definisce le operazioni e lo stato di un tipo.

  - *literal definition*: definisce solo lo stato di un tipo di letterale.

Il modello a oggetti
ODMG Object Model (OM)
Tipi e classi

- Un'*implementazione* di un tipo consiste di una *rappresentazione* e di un insieme di *metodi*. La rappresentazione e' una struttura dati; un metodo e' un corpo di procedura.

  - E' presente una variabile di un tipo opportuno per ogni proprieta' che descrive lo stato di un tipo.

  - E' presente un metodo per ogni operazione definita nell'interfaccia del tipo; un metodo implementa il comportamento esternamente visibile dell'operazione ad esso associata.

Il modello a oggetti
ODMG Object Model (OM)
Tipi e classi

- Subtyping ed ereditarieta' legata al comportamento (supertype-subtype relationship; ISA relationship; generalization-specialization relationship).

```
interface Employee {.....};
interface Professor: Employee {.....};
interface Associate_professor : Professor {.....};
```

Il sottotipo piu' specifico di una gerarchia descrive tutti i comportamenti e le proprieta' delle istanze di quel tipo.

Il modello a oggetti
ODMG Object Model (OM)
Tipi e classi

♦ Un sottotipo

- eredita comportamento e proprieta' dei suoi supertipi;
- puo' specificare nuove caratteristiche rispetto a quelle ereditate;
- puo' specializzare comportamento e stato ereditati (polimorfismo: il comportamento e' invocato run time, rispetto al tipo dell'oggetto di volta in volta considerato).

Il modello a oggetti ODMG Object Model (OM) *Tipi e classi*

- Ereditarieta' multipla: un sottotipo puo' ereditare direttamente da piu' supertipi

```
interface Teaching_assistant: Employee, Student {.....};
```

  - Non sono permessi conflitti di nomi di operazioni.
  - interfacce e classi possono ereditare da interfacce, ma non da classi.

- Le classi sono *tipi concreti* (direttamente istanziabili); le interfacce sono *tipi astratti* (non direttamente istanziabili).

Il modello a oggetti
ODMG Object Model (OM)
Tipi e classi

♦ Ereditarieta' legata allo stato: EXTENDS

- la relazione EXTENDS si applica solo a tipi di oggetti (non a letterali): relazione di ereditarieta' singola fra classi dove la classe subordinata eredita tutte le proprieta' e tutte le operazioni della classe che viene estesa.

```
class Person {
attribute string name;
attribute Date birthDate;
};

class EmployeePerson extends Person: Employee {
attribute Date hireDate;
attribute Currency payRate;
relationship  Manager boss inverse Manager::subordinates;
};

class ManagerPerson extends EmployeePerson: Manager {
relationship  set< Employee> subordinates inverse
EmployeePerson::boss;
};
```

Il modello a oggetti ODMG
Object Model (OM)
Tipi e classi

- Extent: l'extent di un tipo e' l'insieme di tutte le istanze di quel tipo entro la base di dati considerata.
  - Se un oggetto e' istanza del tipo A, allora esso sara' un membro dell'extent di A. Se il tipo A e' un sottotipo del tipo B, allora l'extent di A sara' un sottoinsieme dell'extent di B.
- Chiavi: in alcuni casi le istanze di un tipo possono essere individuate in base al valore di alcune proprieta'.
  - simple key;
  - compound key.

Il modello a oggetti ODMG
Object Model (OM)
Oggetti

- Creazione dell'oggetto;
- Identificatori degli oggetti;
- Nomi di oggetti;
- Tempi di vita;
- Struttura (atomica o no).

Il modello a oggetti ODMG Object Model (OM)

Oggetti

♦ Creazione di oggetti

- Operazione invocata su *factory interfaces*, fornite dall'ODBMS utilizzato.

```
interface ObjectFactory {
Object new();
};
```

- Tutti gli oggetti hanno la seguente interfaccia, ereditata implicitamente:

```
interface Object {
enum        Lock_Type{read, write, upgrade};
exception   LockNotGranted{};
void        lock(in Lock_Time mode)
raises(LockNotGranted);
boolean     try_lock(in Lock_Type mode);
boolean     same_as(in Object anObject);
Object      copy();
void        delete();
};
```

Il modello a oggetti ODMG Object Model (OM)

Oggetti

- L'*identificatore di un oggetto* e' unico e non cambia mai. La rappresentazione dell'identita' di un oggetto e' ottenuta attraverso l'dentificatore dell'oggetto (*object identifier*, o Object_Id). **Possono cambiare valore gli attributi di un oggetto e le sue relazioni con altri oggetti; l'Object_Id di un oggetto non muta mai il suo valore.**

- Gli *object identifier* vengono generati dall'ODBMS e non dall'applicazione.

Il modello a oggetti ODMG Object Model (OM)

Oggetti

- Un oggetto puo' avere uno o piu' *nomi*, significativi al programmatore o all'utente finale. Il nome rappresenta un modo per accedere in modo semplificato agli oggetti piu' importanti ("radici") della base di dati. l'ODBMS gestisce la corrispondenza fra nomi e *object identifier*. Il nome non e' definito in alcuna interfaccia e non corrisponde a valori di proprieta'.

- Il *tempo di vita* di un oggetto determina come deve essere gestita la memorizzazione dell'oggetto stesso ed e' definita alla creazione dell'oggetto. Il tempo di vita di un oggetto e' indipendente dal suo tipo.

  - transient
  - persistent

Il modello a oggetti ODMG
Object Model (OM)
Oggetti

- Atomic object: tipi definiti dall'utente;

- Collection object: sono supportati attraverso i generatori di tipi:
  - Set<t>
  - Bag<t>
  - List<t>
  - Array<t>
  - Dictionary<t,v>

- Structured object:
  - Date
  - Interval
  - Time
  - Timestamp

Il modello a oggetti ODMG
Object Model (OM)
Oggetti

♦ Tutti i tipi che modellano collezioni ereditano dal tipo Collection.

```
interface Collection: Object {
exception           InvalidCollectionType{};
exception           ElementNotFound{any element; };
unsigned long       cardinality();
boolean             is_empty();
boolean             is_ordered();
boolean             allows_duplicates();
boolean             contains_element(in any element);
void                insert_element(in any element);
void                remove_element(in any element)
                        raises(ElementNotFound);
Iterator            create_iterator(in boolean stable);
BidirectionalIterator   create_bidirectional_iterator(in boolean
                        stable) raises(InvalidCollectionType);
};
```

Il modello a oggetti ODMG
Object Model (OM)
Oggetti

```
interface Iterator: Object {
exception   NoMoreElements{};
exception   InvalidCollectionType{};
boolean     is_stable();
boolean     at_end();
boolean     next(out any nex_obj);
void        reset();
void        next_position() raises(NoMoreElements);
any         get_element() raises(NoMoreElements);
void        replace_element(in any element)
                raises(InvalidCollectionType);
};

interface BidirectionalIterator: Iterator {
boolean     at_beginning();
void        previous_position() raises(NoMoreElements);
};
```

Il modello a oggetti ODMG
Object Model (OM)
Oggetti

```
interface Set: Collection {
Set             create_union(in Set other_set);
Set             create_intersection(in Set other_set);
Set             create_difference(in Set other_set);
boolean         is_subset_of(in Set other_set);
boolean         is_proper_subset_of(in Set other_set);
boolean         is_superset_of(in Set other_set);
boolean         is_ proper_superset_of(in Set other_set);
};

interface Bag: Collection {
unsigned long   occurrences_of(in any element);
Bag             create_union(in Bag other_bag);
Bag             create_interesection(in Bag other_bag);
Bag             create_difference(in Bag other_bag);
};
```

Il modello a oggetti ODMG
Object Model (OM)
Oggetti

```
interface List: Collection {
exception   InvalidIndex{unsigned long index; };
void   replace_element_at(in unsigned long index, in any element)
           raises(InvalidIndex);
any    remove_element_at(in unsigned long index)
           raises(InvalidIndex);
any    retrieve_element_at(in unsigned long index)
           raises(InvalidIndex);
void   insert_element_after(in any obj, in unsigned long index)
           raises(InvalidIndex);
void   insert_element_before(in any obj, in unsigned long index)
           raises(InvalidIndex);
void   insert_element_first(in any obj);
void   insert_element_last(in any obj);
any    remove_first_element () raises(ElementNotFound);
any    remove_last_element () raises(ElementNotFound);
any    retrieve_first_element () raises(ElementNotFound);
any    retrieve_last_element () raises(ElementNotFound);
List   concat(in List other_list);
void   append(in List other_list);
};
```

Il modello a oggetti ODMG
Object Model (OM)
Oggetti

```
interface Array: Collection {
exception   InvalidIndex{unsigned long index; };
void        replace_element_at(in unsigned long index, in any
              element) raises(InvalidIndex);
void        remove_element_at(in unsigned long index)
              raises(InvalidIndex);
any         retrieve_element_at(in unsigned long index)
              raises(InvalidIndex);
void        resize(in unsigned long new_size);
};
```

Il modello a oggetti ODMG Object Model (OM)

Oggetti

```
struct Association {anyK key; anyV value; };

interface Dictionary: Collection {
exception   KeyNotFound{anyK key; };
void   bind(in anyK key, in anyV value);
void   unbind(in anyK key) raises(KeyNotFound);
anyV lookup(in anyK key) raises(KeyNotFound);
boolean      contains_key(in anyK key);
};
```

Il modello a oggetti ODMG Object Model (OM)

Letterali

- Atomic literal
  - long
  - short
  - unsigned long
  - unsigned short
  - float
  - double
  - boolean
  - octet
  - char (abbreviato per character)
  - string
  - enum (abbreviato per enumeration)

Il modello a oggetti ODMG
Object Model (OM)
Letterali

♦ Collection literal

- set<t>
- bag<t>
- list<t>
- array<t>

♦ Structured literal

- date
- interval
- time
- timestamp
- strutture definite dall'utente: structure<>

♦ Tipo Table

Table(a1:t1, a2: t2, ..., an:tn)

equivale a

bag<Struct<t1 a1, t2 a2, tn an>>

Il modello a oggetti ODMG Object Model (OM)

Modellazione dello stato: proprieta'

- Attributi

```
Class Person {
attribute     short age;
attribute     string name;
attribute     enum gender{male, female};
attribute     Set<Phone_no> phones;
.............
};
```

- Relazioni

```
Class Professor {
.............
relationship Set<Course> teaches inverse
Course::is_taught_by;
.............
};

Class Course {
.............
relationship Professor is_taught_by inverse
Professor::teaches;
.............
};
```

Il modello a oggetti ODMG
Object Model (OM)
Modellazione del comportamento: operazioni

- Operation signature (conforme a OMG CORBA)

- Ogni operazione e' definita su un singolo tipo

  - overload (stesso nome di operazione per diversi tipi)

- Eccezioni gestite attraverso il tipo predefinito exception

```
interface Iterator: Object {
exception       NoMoreElements{ };
.................
void   next_position() raises(NoMoreElements);
any    get_element() raises(NoMoreElements);
.................
};
```

Il modello a oggetti ODMG
Object Model (OM)
Metadati

Informazioni che descrivono gli oggetti della base di dati, ovvero lo schema della base di dati, memorizzati nel *ODL schema repository*. (MetaObject, Interface, Class, Attribute, Type, ...)

- Built-in types
- Regole di compatibilita' fra i tipi
  - Se TS e' sottotipo di T, un oggetto di tipo TS puo' essere assegnato ad una variabile di tipo T; l'inverso non e' possibile.
- Null value

Il modello a oggetti ODMG Object Model (OM)

Transazioni

- Proprieta' ACID
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- Sequenza lineare di esecuzione di transazioni entro un singolo processo, su un singolo database logico

Il modello a oggetti ODMG Object Model (OM)

Transazioni

- Locking e controllo della concorrenza

- Operazione di transazione

```
interface Transaction {
exception TransactionInProgress{ };
exception TransactionNotInProgress{ };
void   begin() raises(TransactionInProgress);
void   commit() raises(TransactionNotInProgress);
void   abort() raises(TransactionNotInProgress);
void   checkpoint() raises(TransactionNotInProgress);
void   join();
void   leave();
boolean     isOpen();
};
```

Il modello a oggetti ODMG Object Model (OM) *Operazioni sulla base di dati*

♦ Il tipo Database

```
interface Database {
void   open(in string database_name);
void   close();
void   bind(in any an_object, in string_name);
Object       unbind(in string_name);
Object       lookup(in string object_name);
Module       schema();
};
```

ODMG
Object Definition Language

- Linguaggio per la definizione astratta di interfacce, letterali, classi, transazioni, eccezioni, ...
  - a supporto di tutti i costrutti semantici di ODMG Object Model;
  - compatibile con OMG IDL;
  - estendibile;
  - non legato ad alcun linguaggio di programmazione.

ODMG
Object Definition Language

```
class Person
(       extent people)
{
        attribute string  name;
        attribute struct Address {unsigned short number, string
                street, string city_name} address;
        relationship Person spouse inverse Person::spouse;
        relationship set<Person> children inverse Person::parents;
        relationship list<Person> parents inverse Person::children;
        void birth (in string name);
        boolean marriage(in string person_name)
        raises(no_such_person);
        unsigned short ancestors(out set<Person> all_ancestors)
        raises(no_such_person);
        void move(in string new_address);
};

class City
(       extent cities)
{
        attribute unsigned short city_code;
        attribute string name;
        attribute set<Person> population;
};
```

ODMG
Object Interface Language

- Formato usato per scambiare oggetti fra differenti basi di dati, fornire una descrizione esterna degli oggetti, caricare oggetti da file.

SCHEMA

```
Struct PhoneNumber {
        unsigned short    CountryCode;
        unsigned short    AreaCode;
        unsigned short    PersonCode;
};
Struct Address {
        string            Street;
        string            City;
        PhoneNumber       Phone;
};
Class Person {
attribute string Name;
attribute Address PersonAddress;
};
```

ISTANZA

Sarah **Person**{**Name** “Sarah”, **PersonAddress**{**Street** “Willow Road”, **City** “New York”, **Phone**{**CountryCode** 1, **AreaCode** 560, **PersonCode** 5677}}}

ODMG
Object Query Language
OQL

- si fonda su ODMG Object Model;
- e' simile a SQL-92;
- fornisce primitive per considerare insiemi di oggetti;
- e' un linguaggio funzionale dove gli operatori possono essere liberamente composti: il risultato e' un oggetto appartiene all'insieme dei tipi di ODMG Object Model;
- non e' computazionalmente completo;

ODMG
Object Query Language
OQL

- puo' essere invocato dall'interno di un linguaggio di programmazione o invocare operazioni programmate in quel linguaggio;
- non sono forniti espliciti operatori di update (rispetto dell'encapsulation);
- accesso dichiarativo agli oggetti;
- dotabile di semantica formale.

ODMG
Object Query Language
OQL

♦ Definizione e risultato di query

•

select distinct x.age

from people x

where x.name = “Pat”

Tipo del risultato: set<integer> (literal)

•

select distinct struct(a: x.age, s: x.sex)

from people x

where x.name=”Pat”

Tipo del risultato: set<struct> (literal)

•

Chairman

•

Persons

ODMG
Object Query Language
OQL

- Identita' degli oggetti

  - Creazione di oggetti

  Person(name: "Pat", birthdate:"3/28/56", salary: 100,000)

  - Selezione di oggetti esistenti

  select x from Persons x where x.name="Pat"

ODMG
Object Query Language
OQL

♦ Path expression (la dot notation)

- p.spouse.address.city_name

- select c.name
  from p.children c
  Tipo del risultato: Bag<string>

- select distinct c.name
  from p.children c
  Tipo del risultato: Set<string>

- select c.address
  from Persons p, p.children c
  Tipo del risultato: Bag<string>

ODMG
Object Query Language
OQL

Estensione agli oggetti del comando SELECT

select [distinct] *projection_attributes*
from *variable_declarations*
[where *condition*]
[group by *partition_attributes*]
[having *condition*]
[order by *sort_criterion*]