
# Facoltà di Ingegneria Informatica
# SAPIENZA – Università di Roma

Tesina per il corso di:

## *Seminari di ingegneria del software*

## *A.A. 2007/2008*

Docente: **Giuseppe De Giacomo**

Autore: **Carlo Tassi**


# Obiettivi

- ✔ Traduzione in **DL-Lite** delle **ontologie** corrispondenti ai diagrammi UML delle classi presi dai testi d'esame del corso di Progettazione del software I dell'anno 2008

- ✔ Realizzazione (*per quanto possibile*) **query congiuntive** al posto dei programmi richiesti per gli Use Case


# Un passo indietro...

## ✔ Ontologia

*"An **ontology** in computer science and information science is a formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain." (da http://en.wikipedia.org).*


# Un passo indietro...

## ✔ Ontologia

- **Accesso ai dati trasparente**: nasconde all'utente dove e come sono memorizzati i dati
- **Vista concettualizzata dei dati**: simile al Data Integration
- **Necessità di strumenti di ragionamento automatici**: in quanto le informazioni qui possono essere incomplete o mancanti


# Un passo indietro...

## ✔ DL-Lite

*"**Description logics** (DL) are a family of knowledge representation languages which can be used to represent the concept definitions of an application domain (known as terminological knowledge) in a structured and formally well-understood way. The name description logic refers, on the one hand, to concept descriptions used to describe a domain and, on the other hand, to the logic-based semantics which can be given by a translation into first-order predicate logic" (da http://en.wikipedia.org).*


# Un passo indietro...

## ✔ DL-Lite vs OWL DL

**OWL DL:**

- Linguaggio per la definizione di ontologie approvato dal **W3C**
- **OWA (Open World Assumption)**: se un'affermazione non può essere provata come vera allo stato della conoscenza acquisita allora non si può concludere che sia falsa.
- Non può esprimere **attributi di ruolo**

**DL-Lite:**

- Essenzialmente un sottoinsieme di OWL DL
- Può esprimere **attributi di ruolo**


# Un passo indietro...

## ✔ DL-Lite

**Struttura del dominio di interesse**

- **Concetti:** insiemi di oggetti → CLASSI
- **Ruoli:** relazioni binarie sugli oggetti → ASSOCIAZIONI

**Struttura della conoscenza**

- **TBox:** gerarchie di concetto e ruoli. *Livello intensionale* della conoscenza.
- **ABox:** *asserzioni* sulla conoscenza anche parziale del modello descritto nella TBox. Descrive dunque il *livello estensionale* della conoscenza.


# Un passo indietro...

## ✔ DL-Lite

### Sintassi

- **Tedesca:** costrutti logici/matematici difficile da parsare in documenti xml per i programmi di interpretazione
- **Funzionale:** più intuitiva e completamente testuale


# Un passo indietro...

## ✔ SparSQL

**Linguaggio di query per ontologie in DL-Lite:**

- **SPARQL:** linguaggio di query standard W3C per OWL
- **SQL:** linguaggio di query per basi di dati (assume *CWA*)

**Implementazione di EQL-LITE(UCQ):**

- **Linguaggio epistemico**
- **Principio:** *"su ciò che conosci hai informazione completa"*
- **Chiusura dinamica della conoscenza** → queries in FOL decidibili


# Traduzione in DL-LiteA dei diagrammi UML delle classi

## ✔ Classe

**Concetto**

- **Sintassi tedesca:** (dedotto dalla presenza del concetto)
- **Sintassi funzionale:** `Class(NomeClasse)`

## ✔ Attributo di classe

**Attributo di concetto**

- **Sintassi tedesca:**

$$range(nomeAttributo) \subseteq xsd:string$$
$$domain(nomeAttributo) \subseteq NomeClasse$$

- **Sintassi funzionale:**

```
DataPropertyRange(nomeAttributo rdf:string)
DataPropertyDomain(nomeAttributo NomeClasse)
```


# Traduzione in DL-LiteA dei diagrammi UML delle classi

## ✔ Associazione

**Ruolo**

- **Sintassi tedesca:**

$$\exists nomeAssoc \subseteq ClasseSorgente$$

$$\exists nomeAsso\bar{c} \subseteq ClasseDestinazione$$

- **Sintassi funzionale:**

```
ObjectPropertyRange(nomeAssoc ClasseSorgente)
ObjectPropertyDomain(nomeAssoc ClasseDestinazione)
```

## ✔ Attributo di associazione

**Attributo di ruolo**

- **Sintassi tedesca:**

$$range(nomeAttributo) \subseteq xsd:tipo$$

$$domain(nomeAttributo) \subseteq nomeAssociazione$$

- **Sintassi funzionale:**

```
ObjectPropertyDataRange(nomeAttributo rdf:string)
ObjectPropertyDataDomain(nomeAttributo nomeAssociazione)
```


# Traduzione in DL-LiteA dei diagrammi UML delle classi

## ✔ Generalizzazione

- **Sintassi tedesca:**

  $ClasseFiglia \sqsubseteq ClassePadre$

- **Sintassi funzionale:**

  `SubClassOf(ClasseFiglia ClassePadre)`

## ✔ Specializzazione

- **Sintassi tedesca:**

  $associazioneSubset \sqsubseteq associazione$

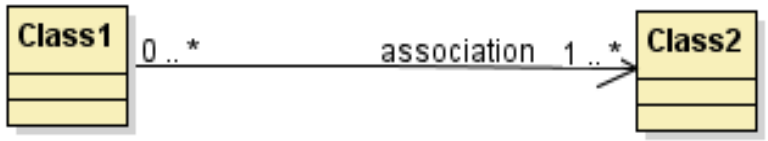
- **Sintassi funzionale:**

  `SubObjectPropertyOf(associazioneSubset associazione)`

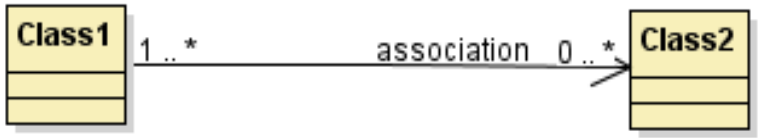
# Traduzione in DL-LiteA dei diagrammi UML delle classi

## ✔ Cardinalità di associazione

- **1..* relativa alla classe destinazione**
  - **Sintassi tedesca:**
    $Class1 \subseteq \exists association$
  - **Sintassi funzionale:**
    `SubClassOf(Class1 ObjectMinCardinality(1 association))`



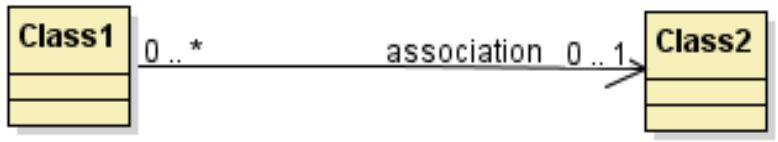

- **1..* relativa alla classe sorgente**
  - **Sintassi tedesca:**
    $\exists association^{-} \subseteq Class1$
  - **Sintassi funzionale:**
    `SubClassOf(ObjectMinCardinality(1 inverseObjectPropertyOf(association)) Class1)`



# Traduzione in DL-LiteA dei diagrammi UML delle classi

## ✔ Cardinalità di associazione

- **0..1 relativa alla classe destinazione**



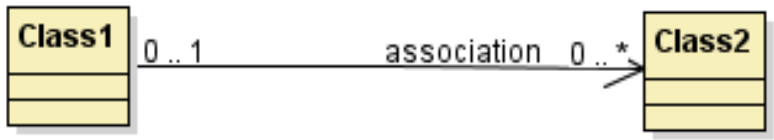

- **Sintassi tedesca:**

  $(funct\ association)$

- **Sintassi funzionale:**

```
FunctionalObjectProperty(association)
```

- **0..1 relativa alla classe sorgente**




- **Sintassi tedesca:**

  $(funct\ association\bar{n})$

- **Sintassi funzionale:**

```
FunctionalObjectProperty(
        inverseObjectPropertyOf(association))
```

- **La cardinalità 1..1 è data dalla composizione delle precedenti**


# Traduzione in DL-LiteA dei diagrammi UML delle classi

## ✔ Cardinalità di attrib. di classe

- **1..* (attributo multivalore)**
  - **Sintassi tedesca:**
    $Classe \sqsubseteq domain(attributo)$
  - **Sintassi funzionale:**
    `SubClassOf(Classe DataMinCardinality(1 attributo))`

- **0..1 (opzionale)**
  - **Sintassi tedesca:**
    $(funct\ attributo)$
  - **Sintassi funzionale:**
    `FunctionalDataProperty(attributo)`

- **La cardinalità 1..1 è data dalla composizione delle precedenti**


# Traduzione in DL-LiteA dei diagrammi UML delle classi

## ✔ Cardinalità di attrib. di assoc.

- **1..* (attributo multivalore)**
  - **Sintassi tedesca:**
    $$associazione \sqsubseteq domain(attributo)$$
  - **Sintassi funzionale:**
    ```
    SubObjectPropertyOf(associazione
          ObjectPropertyDataMinCardinality(1 attributo))
    ```
- **0..1 (opzionale)**
  - **Sintassi tedesca:**
    $$(funct\ attributo)$$
  - **Sintassi funzionale:**
    ```
    FunctionalObjectPropertyData(attributo)
    ```
- **La cardinalità 1..1 è data dalla composizione delle precedenti**


# Vincoli non esprimibili in DL-Lite

Per tali vincoli sarà necessario verificarne la consistenza interrogando l'ontologia attraverso query booleane.
In particolare dovremo verificare:

## ✔ Completeness

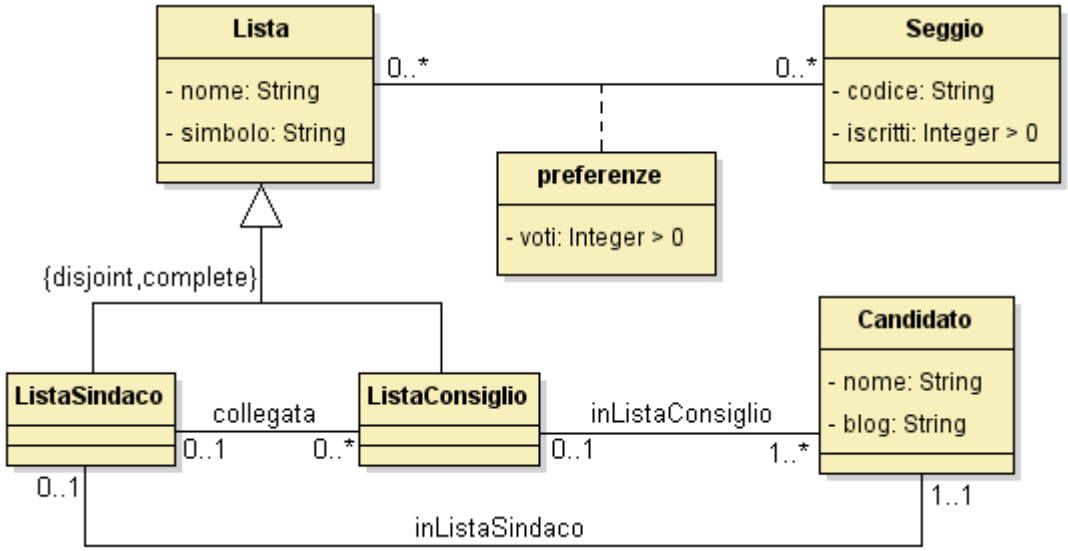
Una possibile strategia consiste nel verificare che l'insieme dato dalla differenza insiemistica deglia elementi della classe padre meno l'unione degli elementi delle classi figlie sia vuoto.

## ✔ Cardinalità diverse da 1 e *

Una possibile strategia per questo caso invece consiste nel contare per ogni associazione il numero di occorrenze degli elementi della classe interessata al vincolo e verificare che rispetti la cardinalità richiesta.

# Un caso di studio...

L'applicazione da progettare riguarda una parte del sistema di gestione di elezioni in un collegio elettorale del comune...




L'ufficio elezioni è interessato ad effettuare diversi controlli sui seggi e le liste, in particolare:
data una lista $\ell$, restituire l'insieme dei candidati a sindaco contenuti in $\ell$, cioè se $\ell$ è una lista per il sindaco restituire l'insieme costituito dal solo candidato presente in essa, se $\ell$ è una lista per il consiglio restituire l'insieme formato dai candidati presenti in essa che sono anche candidati in una lista per il sindaco;


# TBox

```
Class(Lista)
Class(Seggio)
Class(ListaSindaco)
Class(ListaConsiglio)
Class(Candidato)


DataPropertyRange(nomeLista rdf:string)
DataPropertyDomain(nomeLista Lista)
SubClassOf(Lista dataMinCardinality(1 nomeLista))
FunctionalDataProperty(nomeLista)

DataPropertyRange(simbolo rdf:string)
DataPropertyDomain(simbolo Lista)
SubClassOf(Lista dataMinCardinality(1 simbolo))
FunctionalDataProperty(simbolo)

DataPropertyRange(codice rdf:string)
DataPropertyDomain(codice Seggio)
SubClassOf(Seggio dataMinCardinality(1 codice))
FunctionalDataProperty(codice)

DataPropertyRange(iscritti rdf:integer)
DataPropertyDomain(iscritti Seggio)
SubClassOf(Seggio dataMinCardinality(1 iscritti))
FunctionalDataProperty(iscritti)

DataPropertyRange(nome rdf:string)
DataPropertyDomain(nome Candidato)
SubClassOf(Candidato dataMinCardinality(1 nome))
FunctionalDataProperty(nome)

DataPropertyRange(blog rdf:string)
DataPropertyDomain(blog Candidato)
SubClassOf(Candidato dataMinCardinality(1 blog))
FunctionalDataProperty(blog)
```

```
SubClassOf(ListaSindaco Lista)
SubClassOf(ListaConsiglio Lista)
DisjointClasses(ListaSindaco ListaConsiglio)


ObjectPropertyDomain(preferenze Lista)
ObjectPropertyRange(preferenze Seggio)

ObjectPropertyDomain(inListaConsiglio Candidato)
ObjectPropertyRange(inListaConsiglio ListaConsiglio)
SubClassOf(ObjectMinCardinality(1inverseObjectPropertyOf(inListaCo
nsiglio)) Candidato)
FunctionalObjectProperty(inListaConsiglio)

ObjectPropertyDomain(inListaSindaco Candidato)
ObjectPropertyRange(inListaSindaco ListaSindaco)
SubClassOf(ObjectMinCardinality(1
InverseObjectPropertyOf(inListaSindaco)) Candidato)
FunctionalObjectProperty(InverseObjectPropertyOf(inListaSindaco))
FunctionalObjectProperty(inListaSindaco)

ObjectPropertyDomain(collegata ListaSindaco)
ObjectPropertyRange(collegata ListaConsiglio)
FunctionalObjectProperty(InverseObjectPropertyOf(collegata))


ObjectPropertyDataRange(voti rdf:integer)
ObjectPropertyDataDomain(voti preferenze)
SubObjectPropertyOf(preferenze ObjectPropertyDataMinCardinality(1
voti))
FunctionalObjectPropertyData(voti)
```


# Vincolo “complete”

```
VERIFY not exists (
      SELECT lista.nome
      FROM sparqltable (
            SELECT ?nome ?simbolo
            WHERE {
                  ?x rdf:type 'Lista'.
                  ?x :nome ?nome.
                  ?x :simbolo ?simbolo.
            }
      ) lista
      WHERE lista.nome not in (
            SELECT listaSindaco.nome
            FROM sparqltable (
                  SELECT ?nome ?simbolo
                        WHERE {
                        ?x rdf:type 'ListaSindaco'.
                        ?x :nome ?nome.
                        ?x :simbolo ?simbolo.
                        }
            ) listaSindaco
            WHERE lista.simbolo = listaSindaco.simbolo AND
                  lista.nome = listaSindaco.nome
            UNION
            SELECT listaConsiglio.nome
            FROM sparqltable (
                  SELECT ?nome ?simbolo
                  WHERE {
                        ?x rdf:type 'ListaConsiglio'.
                        ?x :nome ?nome.
                        ?x :simbolo ?simbolo.
                        }
            ) listaConsiglio
            WHERE lista.simbolo = listaConsiglio.simbolo AND
                  lista.nome = listaConsiglio.nome
      )
)
```


# UCQ SparSQL per Use Case

```
SELECT     cs.nome
FROM  sparqltable (
           SELECT ?nome ?lista
           WHERE {
                ?x rdf:type 'Candidato'.
                ?x :inListaSindaco ?y.
                ?y rdf:type 'ListaSindaco'.
                ?x :nome ?nome.
                ?y :nomeLista ?lista.
           }
     ) cs
WHERE cs.lista = 'listaConsiglio'
UNION
SELECT     cc.nome
FROM  sparqltable (
           SELECT     ?nome ?lista
           WHERE {
                ?x rdf:type 'Candidato'.
                ?x :inListaConsiglio ?y.
                ?y rdf:type 'ListaConsiglio'.
                ?x :nome ?nome.
                ?y :nomeLista ?lista.
           }
     ) cc,
     sparqltable (
           SELECT     ?nome
           WHERE {
                ?x rdf:type 'Candidato'.
                ?x :inListaSindaco ?y.
                ?y rdf:type 'ListaSindaco'.
                ?x :nome ?nome.
           }
     ) cs
WHERE      cc.nome = cs. nome AND
     cc.lista = 'listaConsiglio'
```


# ABox

```
classassertion(L1 Lista)
dataPropertyAssertion(nomeLista L1 listaSindaco)
dataPropertyAssertion(simbolo L1 symLista1)

classassertion(L2 Lista)
dataPropertyAssertion(nomeLista L2 listaConsiglio)
dataPropertyAssertion(simbolo L2 symLista2)

classassertion(LS ListaSindaco)
dataPropertyAssertion(nomeLista LS listaSindaco)
dataPropertyAssertion(simbolo LS symLista1)

classassertion(LC ListaConsiglio)
dataPropertyAssertion(nomeLista LC listaConsiglio)
dataPropertyAssertion(simbolo LC symLista2)

classassertion(C1 Candidato)
dataPropertyAssertion(nome C1 candidato1)
dataPropertyAssertion(blog C1 blogCandidato1)

classassertion(C2 Candidato)
dataPropertyAssertion(nome C2 candidato2)
dataPropertyAssertion(blog C2 blogCandidato2)

classassertion(C3 Candidato)
dataPropertyAssertion(nome C3 candidato3)
dataPropertyAssertion(blog C3 blogCandidato3)

classassertion(C4 Candidato)
dataPropertyAssertion(nome C4 candidato4)
dataPropertyAssertion(blog C4 blogCandidato4)
```

```
classassertion(S1 Seggio)
dataPropertyAssertion(codice S1 seggio1)
dataPropertyAssertion(iscritti S1 100)

objectPropertyAssertion(inListaSindaco C1 LS)

objectPropertyAssertion(inListaConsiglio C1 LC)
objectPropertyAssertion(inListaConsiglio C2 LC)
objectPropertyAssertion(inListaConsiglio C3 LC)
objectPropertyAssertion(inListaConsiglio C4 LC)

objectPropertyAssertion(preferenze LS S1)
objectPropertyDataAssertion(voti LS S1 10)

objectPropertyAssertion(preferenze LC S1)
objectPropertyDataAssertion(voti LC S1 50)
```

# Risultato interrogazione su QuOnto