Basi di dati

Leonardo Ganzaroli

Indice

Introduzione

Questi appunti sono derivanti principalmente dalle dispense dei corsi *Basi di dati 1 e 2* che ho seguito durante la laurea Triennale di informatica all'università "La Sapienza".

Il modulo 2 del corso era basato sulla progettazione di un DB nella sua interezza partendo da un documento di specifica, per modellare il tutto si usava una combinazione di UML e FOL che veniva alla fine "tradotto" in SQL. Essendo il tutto estremamente pratico riporto solo la parte teorica della FOL.

Parte I

1 Definizioni

Definizione Una base di dati è un insieme di file mutualmente connessi.

Definizione Il DBMS è un insieme di strumenti software atti a gestire grandi quantità di dati.

Definizione Un sistema informativo è un complesso di dati organizzati in memoria secondaria.

I dati possono essere:

- **Strutturati**
  Un oggetto viene rappresentato da stringhe di simboli/numeri.
- **Non strutturati**
  Testi in linguaggio naturale.

Esistono 2 modelli per organizzare i dati:

1. **Logici**
   Indipendenti dalle strutture fisiche.

   Alcuni sono:

   - **Reticolare**
     Ha una struttura a grafo in cui:
     - Nodi = record
     - Archi = link
   - **Gerarchico**
     Come il precedente ma il reticolo è una foresta.
   - **Relazionale**
     Presenta un'organizzazione a tabelle.
   - **A oggetti**
     Basato sul concetto di classe.
2. **Concettuali**
   Indipendenti dalle modalità realizzative, rappresentano le entità del mondo reale e le loro relazioni.

I database presentano un'organizzazione a livelli indipendenti che permette di fornire all'utente una vista astratta dei dati:

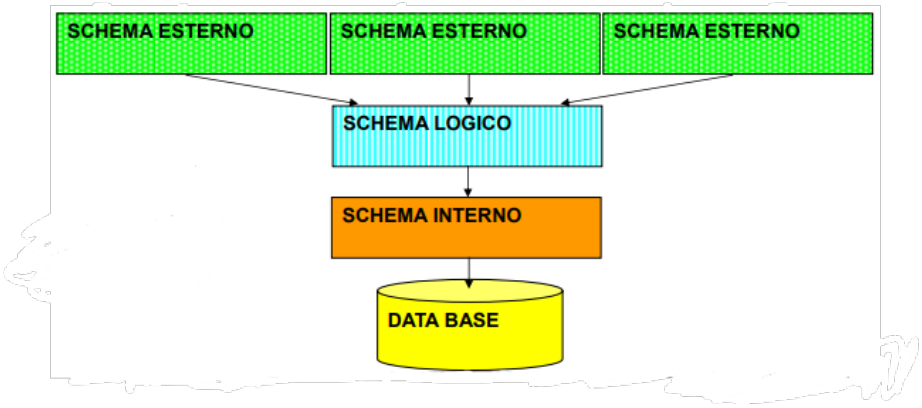

Figura 1: Livelli

- **Esterno**

  Fornisce delle "viste" sul contenuto del DB, queste possono mostrare i dati in modo diverso dai livelli sottostanti.

- **Logico**

  L'effettivo modello usato.

- **Interno**

  I file.

Definizione Lo schema descrive la struttura del DB.

Definizione L'istanza è data dai valori attuali presenti nel DB.

Per poter interagire con il DB si usano i linguaggi:

- DDL (per gli schemi)
- DML (per le istanze)

Nel caso di DB relazionali il linguaggio standard è l'SQL (racchiude entrambe le funzionalità).

2 Modello relazionale

In questo modello i dati vengono rappresentati sottoforma di tabelle, esse rappresentano delle relazioni matematiche tra insiemi.

Definizione Un dominio è un insieme potenzialmente infinito di valori.

Ogni colonna ha associato un dominio, una riga sarà quindi una n-upla del prodotto cartesiano dei domini. Per fornire un'interpretazione ai dati si associa ad ogni colonna un nome detto attributo che funge anche da identificativo della stessa.

Definizione Il valore NULL viene usato per indicare mancanza di informazione.

Nome	ID	Data di nascita
Franco	984	1/15/96
Mario	987	NULL

Gli elementi in rosso sono gli attributi, considerandoli nel loro insieme ottengo lo schema della relazione. Quelli in blu invece sono l'istanza della relazione.

Definizione Una chiave è uno o più attributi che identificano univocamente una n-upla, nello specifico un insieme di attributi X di una relazione R è chiave se:

1. Per ogni istanza di R non esistono 2 n-uple distinte con stessi valori per tutti gli attributi in X
2. Nessun sottoinsieme proprio di X soddisfa il punto precedente

In presenza di più chiavi se ne sceglie una che verrà usata come chiave primaria.

N.B. Una chiave non può assumere valori nulli.

Definizione Un vincolo di integrità è una proprietà che deve essere soddisfatta da ogni istanza di una o più relazioni (si lega quindi agli schemi). Si distinguono:

- Intrarelazionali (Singola relazione)
  - **Chiave primaria**
  - **Dominio**
  - **Unicità**
  - **Esistenza**
  - **Di tupla**
- Interrelazionali (Più relazioni)
  - **Foreign Key**

2.1 Algebra relazionale

Definizione Alcune operazioni richiedono che gli operandi siano union compatibili, ossia devono avere:

- Stesso numero di attributi
- Attributi corrispondenti con stesso dominio

Saranno indicate con #.

Definizione La proiezione ($\pi_{A1,A2,\ldots}(r)$) permette di selezionare solo alcune colonne dell'istanza r, $A1, A2, \ldots$ rappresentano gli attributi di interesse.

N.B. Eventuali elementi duplicati non vengono mostrati.

Definizione La selezione ($\sigma_C(r)$) permette di scegliere le righe di r che rispettano la condizione C, la condizione è una formula booleana composta i cui termini possono essere:

- $A\theta B$
- $A\theta a'$

Dove:

- θ è un operatore di confronto $\{<, >, \leq, \geq, =\}$
- $dom(A) = dom(B)$
- $a' \in dom(A)$

Definizione La ridenominazione ($\rho_{B \leftarrow A}(r)$) fornisce una copia di r in cui l'attributo A è rinominato B.

Ovviamente si possono effettuare le principali operazioni insiemistiche tra le relazioni:

- # Unione
- # Intersezione
- # Differenza
- Prodotto cartesiano

Definizione Il join naturale ($r_1 \triangleright\triangleleft r_2$) fornisce l'insieme delle combinazioni delle n-uple di r_1, r_2 che sono uguali nei loro attributi in comune.

Definizione Il θ-join $(r_1 \triangleright\triangleleft_C r_2)$ fornisce le tuple del prodotto cartesiano $r_1 \times r_2$ che soddisfano C.

Nel caso di query che chiedano se una certa condizione è sempre vera basta trovare gli oggetti che soddisfano la condizione opposta e toglierli dalla soluzione, questo perchè l'opposto di ($\forall$ x è vero) è ($\exists x$ per cui è falso).

Esempi:

Nome	**C#**	**Città**
Rossi	C1	Roma
Rossi	C2	Milano
Bianchi	C3	Roma
Verdi	C4	Roma

Cliente

C#	**A#**	**N-pezzi**
C1	A1	100
C2	A2	200
C3	A2	150
C4	A3	200
C1	A2	200
C1	A3	100

Ordine

- Clienti non di Milano $\sigma_{\neg(\text{Città='Milano'})}(\text{Cliente})$

Nome	**C#**	**Città**
Rossi	C1	Roma
Bianchi	C3	Roma
Verdi	C4	Roma

- Nomi dei clienti che hanno ordinato almeno una volta 150 pezzi

$$\pi_{\text{Nome}}(\sigma_{\text{N-pezzi}=150}(\text{Cliente} \triangleright\triangleleft \text{Ordine}))$$

Con il prodotto cartesiano invece

$$\pi_{\text{Nome}}(\sigma_{\text{C\#}=\text{CC\#} \wedge \text{N-pezzi}=150}(\text{Cliente} \times \rho_{\text{CC\#} \leftarrow \text{C\#}}(Ordine)))$$

Nome
Bianchi

- Nomi e città dei clienti che hanno sempre ordinato al massimo 100 pezzi

$$\pi_{\text{Nome,Città}}(\text{Cliente} \triangleright\triangleleft \text{Ordine}) - \pi_{\text{Nome,Città}}(\sigma_{\text{N-pezzi}>100}(\text{Cliente} \triangleright\triangleleft \text{Ordine}))$$

Vuoto

---

3 Dipendenze funzionali

La ridondanza dei dati in una tabella porta a 3 possibili anomalie:

1. **Aggiornamento**

   Se un dato è presente diverse volte e si verifica un cambiamento dello stesso va aggiornata ogni sua occorrenza.

2. **Inserimento**

   Non si può inserire un dato finché non si verifica una certa condizione.

3. **Cancellazione**

   Cancellare un dato porta alla cancellazione in cascata di altri dati.

Definizione Dato R schema. La coppia ordinata $X, Y \subseteq R$ è detta dipendenza funzionale ($X \to Y$), una certa istanza r di R soddisfa la dipendenza se:

$$\forall\ t_1, t_2 \in r \quad (t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y])$$

Definizione Dato R schema e F insieme di dip. funzionali. Un'istanza r di R è legale se soddisfa tutte le dipendenze in F.

Esempio:

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a2	b2	c1	d3

Se $F = \{A \to B, B \to C\}$ quest'istanza è legale.

Banalmente risulta che esistono delle dipendenze non in F ma che sono comunque rispettate da ogni istanza, nell'esempio precedente una di queste è $A \to C$.

Definizione Dato R schema e F insieme di dip. funzionali. La chiusura di F (F^+) è l'insieme delle dipendenze funzionali soddisfatte da ogni istanza di R, risulta $F \subseteq F^+$.

Inoltre:

$$X \to Y \in F^+ \iff \forall\ A \in Y \quad X \to A \in F^+$$

Definizione Data $X \to Y \in F^+$. Se risulta $Y \subseteq X$ essa viene detta banale.

Definizione Si può ridefinire una chiave come il sottoinsieme $K \subseteq R$ tale che:

- $K \to R \in F^+$
- $\nexists K' \to R \in F^+$ con $K' \subset K$

3.1 F^A

Definizione Sia F^A l'insieme delle dipendenze funzionali definito tramite gli assiomi di Armstrong:

- $f \in F \Rightarrow f \in F^A$
- $Y \subseteq X \subseteq R \Rightarrow X \to Y \in F^A$ (riflessività)
- $X \to Y \in F^A \Rightarrow \forall\ Z \subseteq R\ \ XZ \to YZ \in F^A$ (aumento)
- $X \to Y, Y \to Z \in F^A \Rightarrow X \to Z \in F^A$ (transitività)

Combinandoli si ottengono:

- $X \to Y, X \to Z \in F^A \Rightarrow X \to YZ \in F^A$ (unione)
- $X \to Y \in F^A \wedge Z \subseteq Y \Rightarrow X \to Z \in F^A$ (decomposizione)
- $X \to Y, WY \to Z \in F^A \Rightarrow WX \to Z \in F^A$ (pseudotransitività)

Definizione Dati $X \subseteq R$ ed F. La chiusura di X rispetto ad F (X_F^+) è l'insieme di attributi determinati funzionalmente da X usando F.

Algorithm 1 Calcolo di X^+

```
Z = X
S = {A | Y → V ∈ F ∧ A ∈ V ∧ Y ⊆ Z}
while S ⊈ Z do
    Z = Z ∪ S
    S = {A | Y → V ∈ F ∧ A ∈ V ∧ Y ⊆ Z}
end while
Return Z
```

(Con l'algoritmo si può anche verificare se X è una chiave, basta che alla fine Z contenga tutti gli attributi e che non ci sia un sottoinsieme proprio di X per cui valga lo stesso)

Lemma 1

$X \to Y \in F^A \iff \forall\, A \in Y \;\; X \to A \in F^A \iff \forall\, A \in Y \;\; A \in X^+ \iff Y \subseteq X^+$

Teorema 2 ($F^+ = F^A$) *Si dimostra tramite la doppia inclusione:*

- $F^+ \supseteq F^A$

  Sia $X \to Y \in F^A$, usando i volte gli assiomi si dimostra per induzione che è anche in F^+.

  Per $i = 0$ è banale, per $i > 0$:

  - ***Riflessività***
    Risulta $Y \subseteq X$ quindi $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$.
  - ***Aumento***
    Assumiamo che $V \to W \in F^+$ usando $i-1$ volte gli assiomi, con l'aumento si ottiene $X = VZ, Y = WZ$ per qualche $Z \subseteq R$.

    Risulta $t_1[X] = t_2[X] \Rightarrow t_1[V] = t_2[V] \wedge t_1[Z] = t_2[Z]$
    Per l'ipotesi $t_1[V] = t_2[V] \Rightarrow t_1[W] = t_2[W]$
    Segue $t_1[W] = t_2[W] \wedge t_1[Z] = t_2[Z] \Rightarrow t_1[Y] = t_2[Y]$
  - ***Transitività***
    Assumiamo che $X \to Z, Z \to Y \in F^+$ usando $i-1$ volte gli assiomi, banalmente risulta $t_1[X] = t_2[X] \Rightarrow t_1[Z] = t_2[Z] \Rightarrow t_1[Y] = t_2[Y]$.

- $F^+ \subseteq F^A$

  Per assurdo $X \to Y \in F^+, \notin F^A$.

  Si definisce la seguente istanza:

X^+				$R - X^+$			
1	1	$\cdots$	1	1	1	$\cdots$	1
1	1	$\cdots$	1	0	0	$\cdots$	0

  Prendendo qualunque dipendenza $V \to W \in F$:

  - $t_1[V] \neq t_2[V] \Rightarrow$ *ok*
  - *In caso contrario deve essere $V \subseteq X^+$, quindi $X \to V \to W$ e $W \subseteq X^+$*

  Quindi l'istanza è legale, a questo punto risulta $Y \subseteq X^+$ e per il lemma $X \to Y \in F^A$.

4 3NF

Definizione Uno schema è in terza forma normale se:

$$\forall\ X \to A \in F^+ \text{ con } A \notin X \quad A \text{ è primo o } X \text{ è superchiave}$$

- Primo = appartiene ad una chiave
- Superchiave = contiene una chiave

Esempi:

- $R = ABCD, F = \{A \to B, B \to CD\}, A$ chiave
  - $A \to B$ è ok, A è superchiave
  - Scompongo $B \to CD$, diventa $B \to C, B \to D$, entrambe violano la condizione.
- $R = ABCD, F = \{AB \to CD, BC \to A, D \to AC\}, AB, BC, DB$ chiavi
  - $AB \to CD$ è ok, AB è superchiave
  - $BC \to A$ è ok, BC è superchiave e A è primo
  - Scompongo $D \to AC$, diventa $D \to A, D \to C$, entrambi primi quindi lo schema è in 3NF.

Definizione Data $X \to A \in F^+ \mid A \notin X$. La dipendenza è detta:

- Parziale se A non è primo e $\exists K \mid X \subset K$
- Transitiva se A non è primo e $\forall\ K \quad K - X \neq \emptyset \wedge X \not\subset K$

Si può dire che uno schema è in 3NF sse in F non ci sono né dipendenze parziali né transitive.

4.1 Boyce-Codd

Definizione La forma di Boyce-Codd (BCNF) è una forma più restrittiva in cui:

- Lo schema è in 3NF
- Ogni dip. funzionale non banale $X \to Y$ ha X superchiave

Viene usata meno della 3NF perché scomporre uno schema BCNF in sottoschemi BCNF che preservano tutte le dipendenze non è sempre possibile.

5 Decomposizioni

Definizione Dato R. Una decomposizione di R è una famiglia $\rho = \{R_1, R_2, \ldots\}$ di sottoinsiemi di R che lo ricopre (l'intersezione tra le coppie di sottoinsiemi può essere non nulla).

Bisogna proiettare ogni tupla dell'istanza sugli attributi dei singoli sottoschemi, eventuali duplicatti dovuti a porzioni comuni vanno eliminati.

Solitamente uno schema viene decomposto se non è in 3NF, bisogna garantire che la decomposizione:

- Preservi le dipendenze in F^+
- Abbia un join senza perdita

5.1 Preservazione dipendenze

Definizione 2 insiemi di dipendenze funzionali F, G su R sono equivalenti ($F \equiv G$) se $F^+ = G^+$.

Lemma 3 (Chiusure)

$$F_1 \subseteq F_2^+ \Rightarrow F_1^+ \subseteq F_2^+$$

Essendo $F, G \subseteq G^+$ si ha che ogni dipendenza in F si può ricavare da G tramite Armstrong dato che $G^+ = G^A$, quindi $G \to_A F \to_A F^+$, essendo quest'ultimo derivato da G risulta $F^+ \subseteq G^+$.

Definizione Dati R schema, F su R e $\rho = R_1, R_2, \ldots, R_k$. La decomposizione ρ preserva le dipendenze se:

$$F \equiv G = \bigcup_{i=0}^{k} \pi_{R_i}(F) \quad \text{con } \pi_{R_i}(F) = \{X \to Y \in F^+ \mid XY \subseteq R_i\}$$

Anche in questo caso si verifica tramite doppia inclusione:

- Per definizione stessa di G si ha $G \subseteq F^+$, per il lemma $G^+ \subseteq F^+$
- Basta verificare che $F \subseteq G^+$, per il lemma sarà $F^+ \subseteq G^+$

Algorithm 2 $F \subseteq G^+$

```
for X → Y ∈ F do
    if Y ⊈ X_G^+ then
        Return false
    end if
end for
Return true
```

Per poter calcolare X_G^+ bisogna calcolare G che a sua volta necessita di F^+, si possono saltare questi passaggi usando il seguente algoritmo:

Algorithm 3 X_G^+ tramite F

$Z = X$
$S = \emptyset$
for $i = 1, \ldots, k$ **do**
    $S = S \cup ((Z \cap R_i)_F^+ \cap R_i)$
end for
while $S \not\subseteq Z$ **do**
    $Z = Z \cup S$
    for $i = 1, \ldots, k$ **do**
        $S = S \cup ((Z \cap R_i)_F^+ \cap R_i)$
    end for
end while
Return Z

5.2 Join senza perdita

Definizione Dati R schema, F su R e $\rho = R_1, R_2, \ldots, R_k$. La decomposizione ρ ha un join senza perdita se ogni istanza legale r di R è ricostruibile tramite il join naturale di un'istanza legale dello schema decomposto.

Algorithm 4 Controllo join senza perdita

Costruisci $r_{|\rho| \times |R|}$ in cui $r_{i,j} = \begin{cases} a_j & \text{se } A_j \in R_i \\ b_{i,j} & \text{altrimenti} \end{cases}$

repeat
    for $X \to Y \in F$ **do**
        if $\exists t_1, t_2 \in r \mid t_1[X] = t_2[X] \wedge t_1[Y] \neq t_2[Y]$ **then**
            for $A_j \in Y$ **do**
                if $t_1[A_j] = a_j$ **then**
                $t_2[A_j] = t_1[A_j]$
                else
                $t_1[A_j] = t_2[A_j]$
                end if
            end for
        end if
    end for
until r non è cambiato o ha una riga di sole a
if r ha una riga di sole a **then**
    Return true
end if
Return false

5.3 Algoritmo di decomposizione

Definizione Dato F. Una copertura minimale di F è un insieme $G \equiv F$ tale che:

- $\forall\ X \to Y \in G \quad Y$ è un singolo attributo
- $\forall\ X \to A \in G \quad \nexists X' \subset X \mid G \equiv ((G - \{X \to A\}) \cup \{X' \to A\})$
- $\forall\ X \to A \in G \quad \neg(G \equiv G - \{X \to A\})$

F nel seguente algoritmo è una copertura minimale.

Algorithm 5 Algoritmo di decomposizione

$S = \emptyset$
for $A \in R$ non presente in alcuna dip. funz. in F **do**
    $S = S \cup \{A\}$
end for
if $S \neq \emptyset$ **then**
    $R = R - S$
    $\rho = \rho \cup \{S\}$
end if
if $\exists X \to R \in F$ **then**
    $\rho = \rho \cup \{R\}$
else
    for $X \to A \in F$ **do**
        $\rho = \rho \cup \{XA\}$
    end for
end if
Return ρ

6 Organizzazione fisica

Nel caso relazionale:

- Cella = Campo di un record
- Riga = Record
- Tabella = File
- Database = Insieme di file

La ricerca di un record si effettua tramite il valore di uno o più campi chiave.

Definizione Un albero di ricerca a m-vie è una generalizzazione di un ABR in cui:

- Ogni nodo può avere $[1, m-1]$ valori chiave
- I sottoalberi di un nodo sono massimo $i+1$ con i max dei valori chiave nel nodo
- Ogni nodo ha grado max m
- Le chiavi del sottoalbero il cui puntatore è tra le chiavi k, k' sono nell'intervallo (k, k')

Definizione Un B-Tree è un albero di ricerca a m-vie con le caratteristiche aggiuntive:

- Ogni nodo ha almeno $\frac{m}{2}$ figli (radice esclusa)
- Tutte le foglie sono allo stesso livello

I file possono essere organizzati in vari modi:

- **Heap**
  - Record inseriti alla fine del file
- **Hash**
  - Organizzazione a bucket
  - I record vengono smistati tramite Hashing
  - Bucket Directory in memoria
- **ISAM**
  - File principale + File indice
  - Il primo contiene i record e viene diviso in partizioni
  - Il secondo contiene una entry per partizione del tipo:

    $$< \text{Chiave primo record, Puntatore primo record} >$$

  - Entrambi ordinati rispetto le chiavi di ricerca
- **B-Tree**
  - Il file principale viene puntato solo dall'ultimo livello

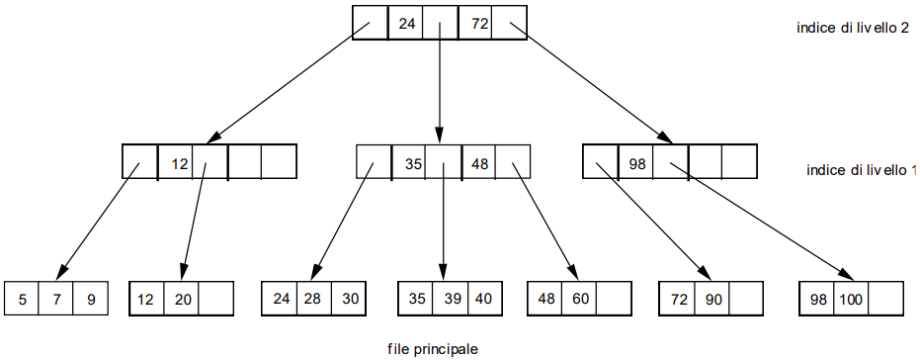

Figura 2: Esempio di B-Tree

7 Concorrenza

Definizione Una transazione è l'esecuzione di una parte di programma che interagisce con la base di dati, esse devono avere le seguenti priorità:

- **Atomicità**

  La transazione deve essere eseguita completamente o per nulla.

- **Consistenza**

  Il DB non deve violare vincoli di integrità prima e dopo la transazione.

- **Isolamento**

  Ogni transazione deve essere indipendente dalle altre.

- **Durabilità**

  Non devono perdersi i cambiamenti apportati.

Definizione La schedule di un insieme di transazioni è seriale se si ottiene permutando le transazioni, risulta sempre corretta.

Definizione Una schedule non seriale è corretta se è serializzabile, ossia "equivale" ad una schedule seriale.

Definizione Un item è un'unita con accesso controllato.

Per evitare eventuali problemi si applica un lock sugli item:

- **Binario**

- **A 3 valori**

  Si aggiunge un altro tipo di lock che permette ad altri di leggere l'item ma non di modificarlo.

7.1 Stallo

Definizione Il punto di commit di una transazione è il punto in cui:

- Ha ottenuto tutti i lock necessari
- Ha effettuato tutti i calcoli

Se si trova a questo punto non può essere abortita.

Definizione I dati sporchi sono quelli scritti da una transazione prima del suo punto di commit.

In caso di deadlock si effettua il roll-back di una delle transazioni coinvolte, ossia:

- La transazione viene abortita
- I suoi effetti sui dati vengono annullati
- Tutti i suoi lock vengono rilasciati

Con il procedimento precedente diventa problematica la situazione in cui altre transazioni abbiano letto i dati sporchi prodotti dalla transazione abortita, per risolvere totalmente il problema si usa il locking a due fasi stretto:

1. Una transazione non scrive fino al suo punto di commit
2. Una transazione non rilascia nessun lock fino alla fine della scrittura

Parte II

8 FOL

8.1 Sintassi

L'alfabeto della logica del primo ordine è formato da:

- Un insieme di variabili V
- Un insieme di simboli di funzione F, ognuno con la sua arità
- Un insieme di simboli di predicato P, ognuno con la sua arità ($= /2 \in P$)
- I connettivi logici ($\neg, \wedge, \vee, \rightarrow, \iff$)
- I quantificatori $\forall, \exists$
- I simboli ( ) ,

Esempi:

- Funzione
  - succ/1, succ(X) è il numero naturale successore di X
  - zero/0, il numero 0 (arità 0 = simbolo di costante)
  - padre/1, padre(X) è il padre della persona X
- Predicato
  - doppio/2, doppio(X,Y) indica che Y è il doppio di X
  - uomo/1, uomo(X) indica che X è un uomo
  - somma/3, somma(X,Y,Z) indica che Z=X+Y

Partendo dall'alfabeto si può definire il linguaggio della FOL, si procede in modo induttivo su 2 passi:

1. Si definisce il linguaggio dei termini

   L'insieme dei termini è definito induttivamente come:

   (a) Ogni variabile in V è un termine

   (b) Ogni simbolo di costante in F è un termine

   (c) $f/n \in F \wedge n > 0 \wedge (t_1, \ldots, t_n$ sono termini$) \Rightarrow f(t_1, \ldots, t_n)$ è un termine

2. Si definisce il linguaggio delle formule (usando il precedente)

   L'insieme delle formule è definito induttivamente come:

   (a) p/n è predicato $\wedge$ $t_1, \ldots, t_n$ sono termini $\Rightarrow$ $p(t_1, \ldots, t_n)$ è una formula (atomica)

   (b) Se ϕ, ϵ sono formule lo sono anche:
   - (ϕ)
   - $\neg\phi$
   - $\phi \wedge \epsilon$
   - $\phi \vee \epsilon$
   - $\phi \rightarrow \epsilon$
   - $\phi \iff \epsilon$

   (c) Se ϕ è una formula e X è una variabile sono formule:
   - $\forall X \;\; \phi$
   - $\exists X \;\; \phi$

Esempi:

- $F = \{zero/0, succ/1, padre/1\}$

  Alcuni termini sono:
  - $zero$
  - var (una variabile)
  - $succ(zero)$
  - $succ(succ(succ(padre(var))))$

- $P = \{doppio/2, somma/3, mortale/1\}$ ed F

  Alcune formule sono:
  - $\forall\ X \;\; mortale(X)$
  - $doppio(succ(zero), X)$
  - $(padre(X) = Z) \rightarrow (mortale(Z))$ (l'uguale non si scrive come gli altri predicati per comodità)
  - $\forall X\ \exists Y\ doppio(X, Y)$

  Alcune sequenze di simboli non formule sono:
  - $X \wedge Y$ (2 termini)
  - $succ(zero)$ (termine)
  - $mortale(mortale(X))$ (predicato dentro predicato)
  - $padre(doppio(X, Z))$ (predicato dentro termine)
  - $\exists zero \;\; mortale(zero)$ ($zero$ non è una variabile)

8.2 Semantica

La semantica definisce il significato di una formula logica (la sua verità), in FOL ci sono 2 livelli sintattici quindi bisogna dare 2 concetti di valutazione:

- Dei termini
  - **Atomici**
    - **Pre-interpretazione**

      Una pre-interpretazione è data da:
      - Un insieme $D \neq \emptyset$ detto dominio di interpretazione
      - La funzione totale $PreI : D^n \to D : f/n \to d \in D$
    - **Assegnamento di variabili**

      Un assegnamento per $PreI$ è la funzione $S : V \to D$
  - **Complessi**

    Dato T insieme dei termini. Si definisce la funzione:

    $$pre - eval^{PreI,S} : T \to D \quad \text{(indicata con } pe\text{)}$$

    Come:
    - X è variabile $\Rightarrow pe(X) = S(X)$
    - c simbolo di costante $\Rightarrow pe(c) = PreI(c)$
    - $pe(f(t_1, \ldots, t_n)) = PreI(f(pe(t_1), \ldots, pe(t_n)))$
- Delle formule
  - **Atomiche**
    - **Interpretazione**
      Una funzione che associa a p/n una relazione $I(p) \subseteq \underbrace{D \times \ldots \times D}_{\text{n volte}}$

      Per l'uguale risulta $\{(d, d) \mid d \in D\}$
  - **Complesse**

    Dato ϕ insieme delle formule. Si definisce la formula:

    $$eval^{I,S} : \phi \to \{vero(1), falso(0)\} \quad \text{(indicata con } e\text{)}$$

    Come:
    - $e(p(t_1, \ldots, t_n)) = I(p(pe(t_1), \ldots, pe(t_n)))$
    - Se ϕ, ϵ sono formule:
      - $e((\phi)) = e(\phi)$
      - $e(\neg\phi) = 1$ se $e(\phi) = 0,\ 1$ altr.
      - $e(\phi \wedge \epsilon) = 1$ se $e(\phi) = e(\epsilon) = 1,\ 0$ altr.
      - $e(\phi \vee \epsilon) = 1$ se $e(\phi) = 1$ o $e(\epsilon) = 1,\ 0$ altr.

- $e(\phi \to \epsilon) = e(\neg\phi \vee \epsilon)$
- $e(\phi \iff \epsilon) = e((\phi \to \epsilon) \wedge (\epsilon \to \phi))$

* $e(\exists V\ \phi) = 1$ se esiste $d \in D \mid e^{I,S[V/d]}(\phi) = 1,\ 0$ altr.
* $e(\forall V\ \phi) = 1$ se per ogni $d \in D\ e^{I,S[V/d]}(\phi) = 1,\ 0$ altr.

($S[X/d]$ indica l'assegnamento S eccettuato il fatto che ad X si assegna d)

Esempio:

$$S(X) = 3 \wedge S(Y) = 4 \Rightarrow S[Y/9](X) = 3 \wedge S[Y/9] = 9$$

Definizione Una formula è detta chiusa se tutte le sue variabili sono quantificate, il suo valore di verità dipende dall'interpretazione usata e conseguentemente dal dominio.

Definizione Il modello di una formula chiusa ϕ è un'interpretazione M per cui $e^{M,S}(\phi) = vero$ per qualsiasi assegnamento S ($M| = \phi$).

Definizione Una formula ϕ è definita:

- **Soddisfacibile**

$$\exists I, S \quad e^{I,S}(\phi) = vero$$

- **Insoddisfacibile**

$$\forall\ I, S \quad e^{I,S}(\phi) = falso$$

- **Valida**

$$\forall\ I, S \quad e^{I,S}(\phi) = vero$$

Convenzionalmente si usa la seguente precedenza per connettivi e quantificatori:

1. $\neg$
2. $\wedge, \vee$
3. $\forall, \exists$
4. $\to$

8.3 Esempio completo

- $F = \{zero/0, succ/1\}$
- $P = \{doppio/2, somma/3\}$
- $V = \{X, Y, Z\}$
- Pre-interpretazione $PreNAT$
  - $D = \mathbf{N}$
  - $PreNAT(zero) = 0$
  - $PreNAT(succ) : D \to D : PreNAT(0) = 1, PreNAT(1) = 2, \ldots$
- Assegnamento W
  - $W(X) = 3$
  - $W(Y) = 6$
  - $W(Z) = 4$
- Valutazione termini pe
  - $pe(zero) = PreNAT(zero) = 0$
  - Esempio:

$$\begin{aligned} pe(succ(succ(X))) &= PreNAT(succ(pe(succ(X)))) \\ &= PreNAT(succ(PreNAT(succ(pe(X))))) \\ &= PreNAT(succ(PreNAT(succ(W(X))))) \\ &= PreNAT(succ(PreNAT(succ(3)))) = 5 \end{aligned}$$

- Interpretazione NAT
  - $NAT(doppio) = \{(x, y) \mid y = 2x\}$
  - $NAT(somma) = \{(x, y, z) \mid z = x + y\}$
- Valutazione e

Esempi di valutazione:

$$\begin{aligned} &e(doppio(succ(succ(zero)), X)) = \\ &= NAT(doppio(pe(succ(succ(zero))), pe(X))) \\ &= NAT(doppio(PreNAT(succ(pe(succ(zero)))), W(X))) \\ &= NAT(doppio(PreNAT(succ(PreNAT(succ(pe(zero))))), 3)) \\ &= NAT(doppio(PreNAT(succ(PreNAT(succ(PreNAT(zero))))), 3)) \\ &= NAT(doppio(PreNAT(succ(PreNAT(succ(0)))), 3)) \\ &= NAT(doppio(PreNAT(succ(1)), 3)) \\ &= NAT(doppio(2, 3)) = falso \end{aligned}$$

$e(somma(succ(zero), zero, succ(zero))) =$
$= NAT(somma(pe(succ(zero)), pe(zero), pe(succ(zero))))$
$= NAT(somma(PreNAT(succ(pe(zero))), PreNAT(zero), PreNAT(succ(pe(zero)))))$
$= NAT(somma(PreNAT(succ(0)), 0, PreNAT(succ(zero))))$
$= NAT(somma(1, 0, 1)) = vero$

$e(\forall X \quad doppio(succ(succ(zero))), X) = e^{NAT, W[X/d]}(doppio(succ(succ(zero)), X))$
$= e^{NAT, W[X/d]}(doppio(2, X)) = falso$ (dato $d = 1$)

$e(\forall X \ \exists Y \quad doppio(X, Y))$ essendo chiusa risulta vera data l'interpretazione di *doppio* e $D = \mathbf{N}$
Invertendo X, Y nei quantificatori diventa falsa, infatti non esiste un numero in $\mathbf{N}$ il cui doppio è 3
