Università degli Studi di Modena e Reggio Emilia

Facoltà di Ingegneria – Sede di Modena

Corso di Laurea in Ingegneria Informatica – *Nuovo Ordinamento*

Realizzazione di una interfaccia Web per la progettazione di uno schema ER e la sua traduzione in RDF

Relatore:
Prof. Sonia Bergamaschi

Candidato:
Mattia Bonacorsi

Anno Accademico 2003-2004

Parole Chiave

PROGETTO_ER
RDF
WEB_RDF
INTERFACCIA_GRAFICA
.NET

INDICE

INDICE DELLE FIGURE

1. INTRODUZIONE

La presente tesi affronta il problema della crearezione di pagine Web dinamiche grafiche per la progettazione concettuale ER di database e la traduzione degli schemi disegnati in RDF.

Già dalla prima definizione del problema da affrontare, l'applicazione si può dividere in due parti ben distinte. La prima parte è relativa allo sviluppo di un'interfaccia grafica per il disegno di schemi ER. Tale interfaccia deve permettere all'utente di disegnare tutti gli elementi che compongono uno schema e di combinarli tra loro a piacimento. La seconda parte è relativa alla traduzione in RDF dello schema ER ottenuto e deve fornire in output codice RDF/XML che, corredato da un vocabolario generale e a nuove definizioni di tipi di dato, rappresentano tutto ciò che ha espresso in modalità grafica l'utente.

Per sviluppare l'applicazione si è deciso di utilizzare il nuovo framework .NET di Microsoft. Questa scelta deriva dal fatto che la tecnologia ASP.NET sostituisce il modello lineare di elaborare le pagine di ASP con un'emulazione del modello ad eventi. Il framework delle pagine ASP.NET è fornito di un implicito produttore di associazioni tra un evento e il suo ascoltatore e permette di creare semplicemente interfacce utente che reagiscono alle azioni dell'utente. Questa scelta ha reso meno complicato creare classi che rappresentano gli elementi di uno schema ER e creare e posizionare oggetti di queste classi.

Come linguaggio di programmazione per la parte di codice per le pagine ASP.NET si è scelto il C#.

Nel capitolo conclusivo di questa tesi è stata introdotta una bibliografia ragionata delle fonti di RDF.

Per lo sviluppo dell'applicazione si è scelto di utilizzare l'ambiente integrato Visual Studio .NET all'interno del sistema operativo Windows XP Professional dotato di Internet Information Service 5.0 come Web server per testare le pagine.

La scrittura del vocabolario RDF e delle nuove definizioni di tipi di dato consiste nella scrittura di codice XML e non richiede particolari ambienti di sviluppo (basta un editor di testo).

L'articolazione della tesi è la seguente.

Nei capitoli due e tre verranno analizzate le tecnologie utilizzate per creare l'applicazione.

In particolare nel capitolo due verrà approfondita la tecnologia ASP.NET con particolare attenzione a come vengono elaborate le pagine Web e quali sono i metodi per mantenere lo stato degli oggetti tra un'iterazione con il server e la successiva.

Nel capitolo tre verrà approfondita la tecnologia RDF con particolare attenzione alla definizione di nuovi tipi di dato e la definizione di vocabolari specifici per l'applicazione.

Nel quarto capitolo verrà presentata com'è stata realizzata l'applicazione, in particolare nei primi due paragrafi si presenteranno il vocabolario e i tipi di dato definiti appositamente per l'applicazione, nel terzo paragrafo le classi utilizzate per rappresentare i tipi di elementi di uno schema ER e negli ultimi due paragrafi le due pagine dinamiche realizzate per il disegno di schemi ER e per la loro traduzione in RDF.

Nel quinto capitolo verrà fatto il punto su possibili sviluppi futuri, nel sesto capitolo verrà presentato un breve glossario con i termini utilizzati in questa tesi e nel settimo capitolo verrà presentata una bibliografia ragionata.

2. LE TECNOLOGIE UTILIZZATE: ASP.NET

Nei prossimi due capitoli vengono presentate le tecnologie utilizzate per la realizzazione della tesi. In particolare nel primo capitolo viene introdotto ASP.NET che è stato utilizzato per scrivere le pagine Web dinamiche del progetto. Per la scrittura del codice delle pagine è stato scelto il linguaggio C#.

.NET è un ambiente di sviluppo software ed esecuzione d'applicazioni che consente di creare, compilare, provare, distribuire ed eseguire software che può essere codificato in diversi linguaggi che si attengono tutti ad un singolo insieme di file Common Language Runtime. Questo significa consentire allo sviluppatore di adottare un qualunque linguaggio di programmazione conforme alle regole ferree del framework .NET e poter far interagire applicazioni scritte in linguaggi differenti.

2.1 IL FRAMEWORK .NET

Da un punto di vista di alto livello, il framework .NET può essere descritto come un sistema operativo virtuale eseguito sopra il sistema operativo vero e proprio. Un esame più preciso rivela che il framework è formato da due componenti principali: il CLR (Common Language Runtime) la Class Library .NET.

ORIENTAMENTO ALLE CLASSI DEL FRAMEWORK .NET

La Class Library .NET è un'ampia gerarchia organizzata di classi. Questi oggetti indicano i servizi da usare per sviluppare servizi personalizzati. Includono supporto per windows form e servizi Web, nonché oggetti per lavorare con XML e dati. Per includere questi servizi nelle applicazioni, si esplora la gerarchia usando i principi tradizionali della programmazione ad oggetti (ES: System.Data.SQLClient). Questi riferimenti espliciti a gruppi di classi nelle librerie del framework sono definiti spazi dei nomi in .NET. Uno spazio dei nomi è un riferimento gerarchico univoco ad una classe specificata o ad un gruppo di classi simili.

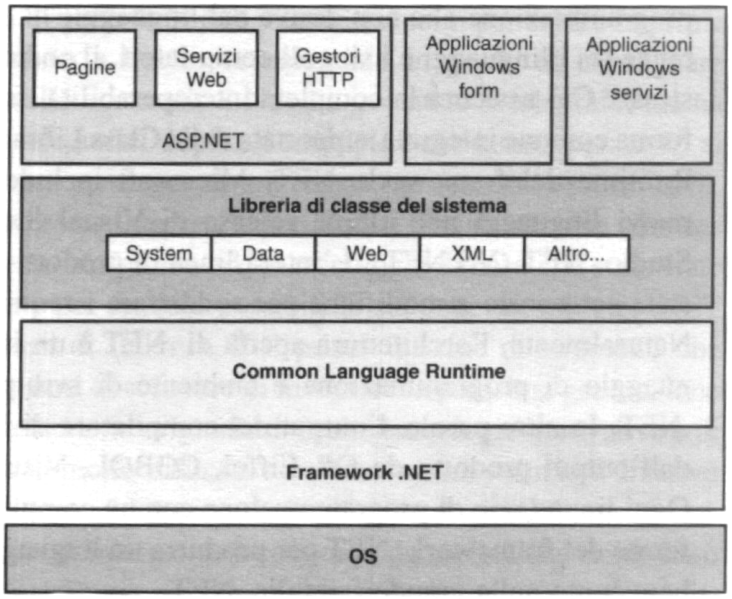

Figura 1 - ASP.NET - Gerarchia

COMMON LANGUAGE RUNTIME

La responsabilità per attività quali la creazione di oggetti, l'esecuzione di chiamate ai metodi e così via, è denominata Common Language Runtime che consente a runtime di fornire servizi aggiuntivi al codice in esecuzione.

Il CLR riveste due ruoli diversi, uno in fase di sviluppo e uno in fase di progettazione.

- In fase di sviluppo, il CLR include un sistema comune di tipi (che consente l'integrazione tra linguaggi diversi), un miglior controllo dei conflitti di versione e servizi di sicurezza oltre la gestione delle eccezioni tra linguaggi, la gestione degli eventi basati sui delegate, il binding dinamico e la reflection che riducono considerevolmente la quantità di codice che occorre scrivere per trasformare la logica di business in componenti riutilizzabili.
- In fase di esecuzione, il CLR fornisce servizi quali la gestione della memoria (incluso il garbage collection), la gestione dei processi e dei thread e il potenziamento della sicurezza.

INTERMEDIATE LANGUAGE (IL) E METADATI

IL (Intermediate Language) è la versione .NET del codice compilato. Qualsiasi linguaggio di programmazione si scelga per la scrittura di un programma, il risultato di una compilazione sarà sempre nello stesso linguaggio intermedio autodescrittivo. IL è una semplice sintassi basata su testo che fa da complemento a un componente autodescrittivo: i metadati. La combinazione di queste due tecnologie offre al runtime gestito di .NET l'abilità di eseguire più operazioni in meno tempo e con meno overhead.

I metadati sono rappresentati in formato XML e descrivono i contenuti dell'eseguibile. Contengono la posizione e l'obbiettivo di ogni oggetto e le sue proprietà, i suoi argomenti, i suoi delegati e i suoi metodi.

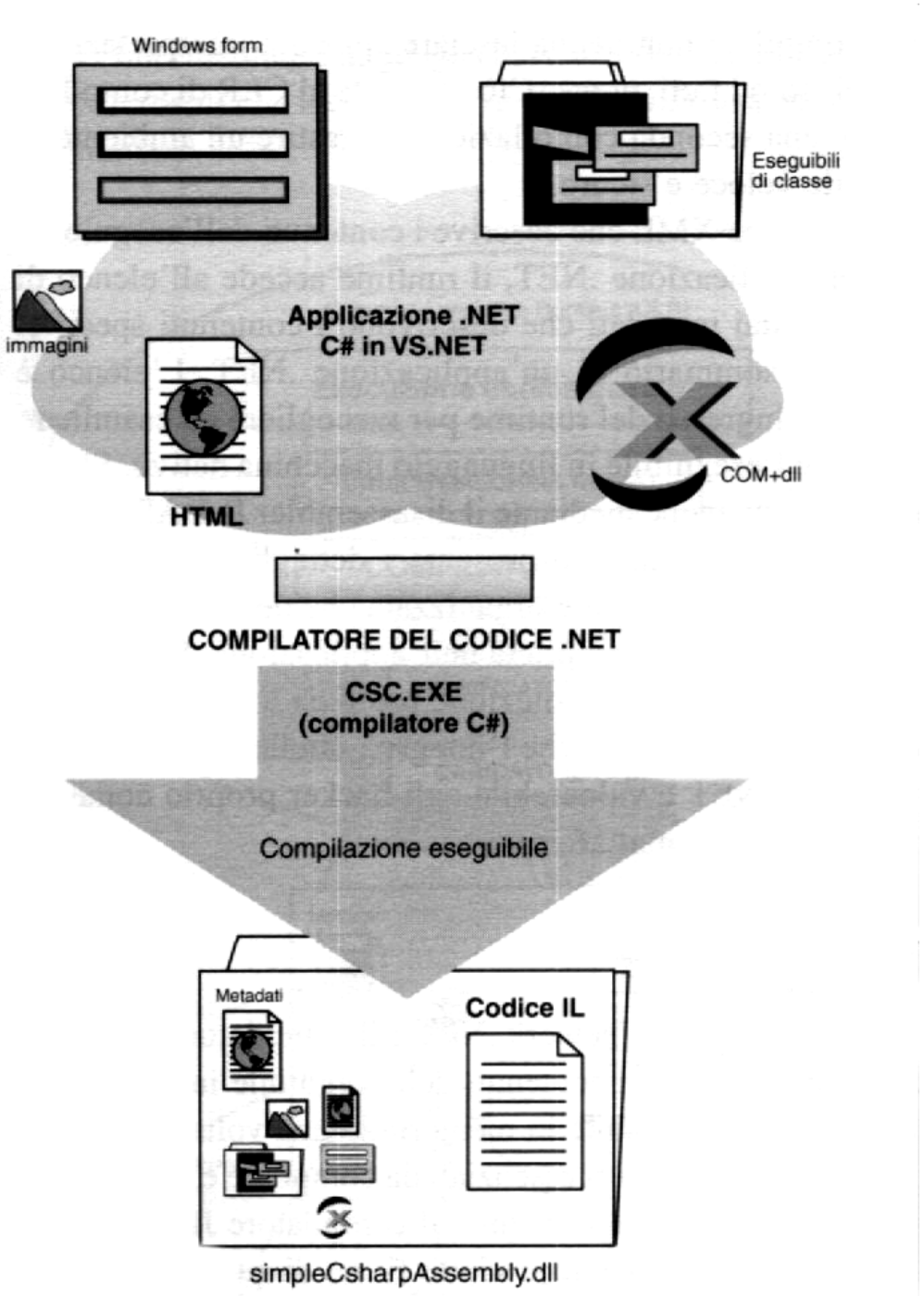

Figura 2 - ASP.NET - Compilare (1P)

Le applicazioni .NET sono compilate Just-In-Time una seconda volta nel codice macchina nativo. Il runtime .NET offre allo sviluppatore la scelta di opzioni di compilazione o la possibilità di specificare che la compilazione venga eseguita al momento dell'istallazione piuttosto che nella fase di lancio dell'applicazione. Questa opzione diventa utile quando si desidera evitare l'overhead della compilazione al lancio dell'applicazione.

Il codice compilato di .NET risolve anche il cosiddetto problema del "DLL Hell". In passato, gli sviluppatori dovevano creare più versioni delle loro applicazioni Win32 per produrre eseguibili

compatibili per ogni categoria delle principali configurazioni di sistema. Tutti gli sforzi fatti non erano sufficienti ad evitare problemi qual'ora un'altra applicazione avesse aggiornato una DLL di sistema condivisa a una nuova versione. Gli sviluppatori .NET non dovranno più preoccuparsi di questi problemi.

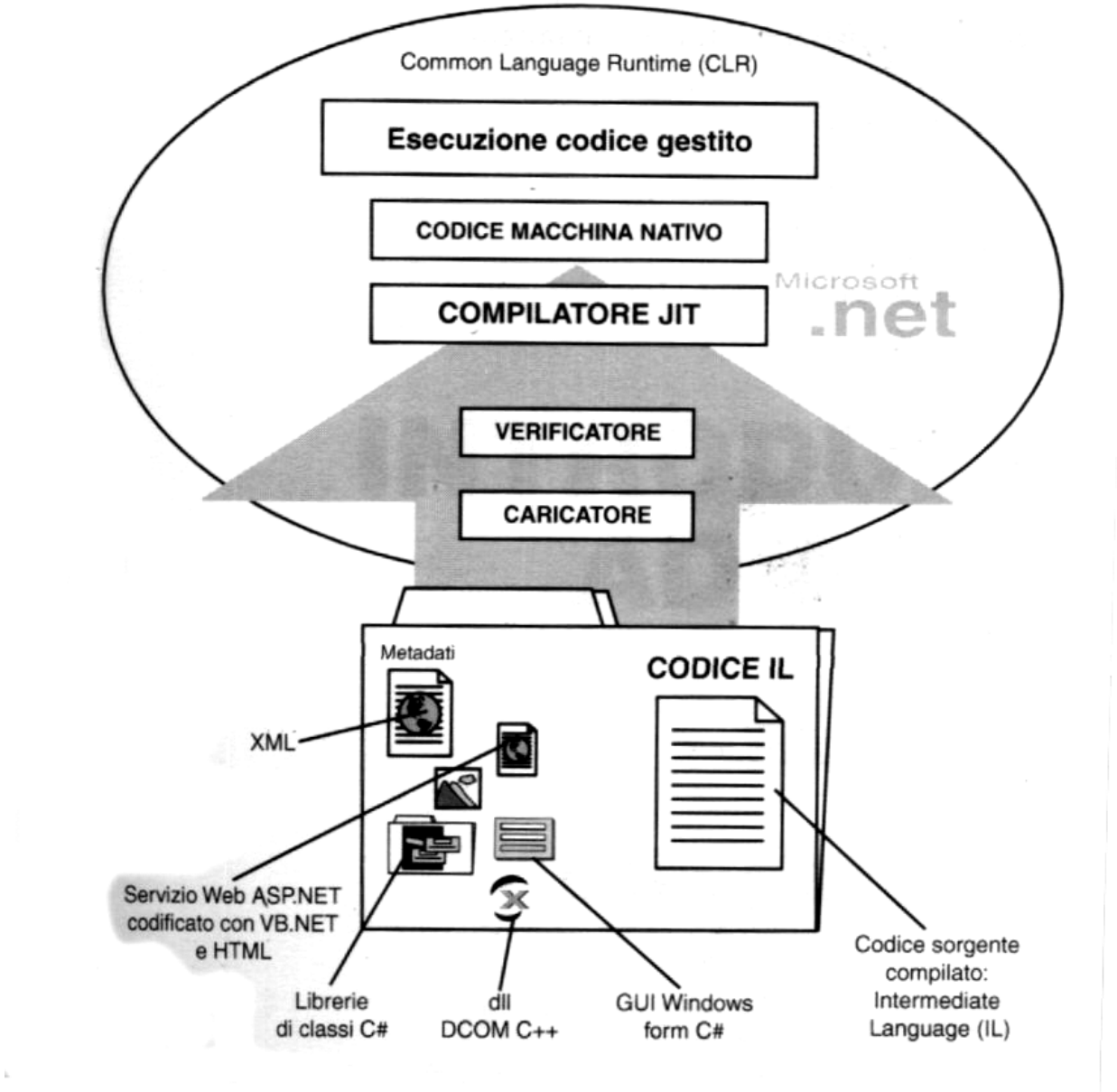

Figura 3 - ASP.NET - Compilare (2P) e eseguire

IL COMPILATORE JIT

Un componente fondamentale del framework .NET è il compilatore JIT (Just-In-Time). Ha il compito di compilare i contenuti dell'eseguibile in linguaggio macchina. Non compila l'intero eseguibile

in memoria in una volta sola. Deve esaminare il codice per individuare le sue parti essenziali da compilare e caricare per assicurare che l'applicazione si apra velocemente. Il compilatore JIT caricherà solo le parti necessarie dell'applicazione. Il prodotto finale di questo approccio è un ambiente di runtime con più spazio di memoria disponibile per elaborare più velocemente gli elementi usati.

2.2 BREVE STORIA DI ASP.NET

Quando ASP (Active Server Pages) è stato rilasciato per la prima volta nel novembre 1996, ha introdotto un metodo semplice per creare pagine Web dinamiche. ASP si è subito affermato per quattro motivi:

- Facilità di accesso ai dati: probabilmente se ASP non fosse stato rilasciato con ADO (ActiveX Data Object) come metodo di accesso a database preferito da Microsoft non si sarebbe affermato così facilmente.
- Semplice struttura di pagine
- Interoperatività con COM (Component Object Model): prima di ASP 1.0, l'abilità di acquistare componenti prefabbricati e installarli in un sito Web era disponibile solo per i programmatori esperti. Portando questa capacità alle masse, ASP ha dato il via a un nuovo mercato di produttori di componenti, che continuano a offrire strumenti prefabbricati potenti e facilmente integrabili nelle applicazioni ASP.
- Facilità di apprendimento per gli sviluppatori di Visual Basic: l'utilizzo di VBScript come linguaggio predefinito ha consentito ai programmatori già esperti di Visual Basic di iniziare subito a usare ASP con il minimo studio.

Ciononostante ASP 1.0 presentava limiti significativi. Il limite principale per chi lavorava con componenti COM era che il Web server doveva essere riavviato a ogni aggiornamento di una DLL (Dynamic Link Library) dove sono memorizzati gli oggetti COM.

Il principale miglioramento da ASP 1.0 a ASP 2.0 è stato MTS (Microsoft Transaction Server). Con MTS, la vita di chi sviluppava o usava componenti COM divenne più semplice. Gestiva l'installazione e la disinstallazione dei componenti, rimuoveva la necessità di riavviare il servizio Web e spesso anche il server. Infine, agiva da intermediario di oggetti, inserendo nella cache le istanze di oggetti e restituendole a ogni richiesta.

Nel febbraio 2000, Microsoft ha rilasciato insieme a IIS 5.0, ASP 3.0. In questa nuova versione MTS venne sostituito dai servizi COM+. COM+ combinava le funzionalità di MTS con i servizi di accoramento dei messaggi.

Mark Anders e Scott Guthtie hanno iniziato a sviluppare ciò che sarebbe diventato ASP.NET nel gennaio 1998. Gli sviluppatori hanno scelto di creare ASP.NET (all'epoca chiamato ASP+) su NGWS (Next Generation Web Servives) Runtime a quell'epoca in fase di sviluppo che sarebbe diventato .NET. NGWS offriva un ricco insieme di librerie di programmazione e presto avrebbe incluso il nuovo linguaggio C#, in cui è scritto ASP.NET.

ASP.NET è stato in fase di sviluppo per più di tre anni e Microsoft ha deciso di basare questo prodotto sulle priorità seguenti:

- Struttura fattorizzata. ASP.NET è scritto come insieme di componenti modulari che possono essere sostituiti o estesi come necessario.
- Scalabilità. Sono stati fatti molti sforzi per creare un modello altamente scalabile, nel rispetto del mantenimento dello stato.

- Disponibilità. AST.NET è stato strutturato per rilevare crash, memory leak, deadlock ed effettuare il ripristino dopo questi eventi.
- Prestazioni. ASP.NET trae vantaggio dai linguaggi compilati e dall'early binding per migliorare le prestazioni e offre supporto di caching estensivo.
- Integrazione di strumenti. L'obbiettivo di Microsoft è rendere la creazione di un sito Web semplice come la creazione di un form di Visual Basic.

2.3 BENEFICI DI ASP.NET RISPETTO AD ASP

ASP.NET e il framework .NET offrono diversi vantaggi rispetto ad ASP classico. ASP.NET è più robusto, sicuro e scalabile, offre strumenti migliori, che consentono ai programmatori di essere più produttivi e supporta diversi linguaggi, in modo da usare quello che si preferisce. ASP.NET è anche più facile da gestire e distribuire. Di seguito sono illustrati i vantaggi ASP.NET rispetto a ASP.

- ASP.NET è compilato, non interpretato. I programmi compilati sono eseguiti più velocemente di quelli interpretati. Ogni pagina è compilata alla prima richiesta; il codice compilato viene conservato fino a quando non si modifica la pagina o si riavvia l'applicazione.
- Separazione del codice che implementa la logica di business dal codice per la presentazione delle informazioni. ASP.NET consente la vera separazione del codice dalla presentazione, che consente ai designer e ai programmatori di collaborare con meno frustrazioni e tempo passato a unire aspetto e funzionalità delle pagine.
- Fine del "DLL Hell". Con ASP.NET, i componenti non devono essere condivisi sul server, ma possono essere posizionati con le singole applicazioni. Inoltre, i componenti sono memorizzati con l'applicazione, che può essere spostata ricorrendo alla copia di file. Non è necessario apportare modifiche al registro di sistema o preoccuparsi di MTS/COM+.
- Installazione lato per lato. I nuovi servizi e funzioni possono essere installati ed eseguiti in parallelo con applicazioni ASP classiche esistenti. Condividono infatti la stessa struttura di cartelle; per migrare ogni file, dopo averlo impostato per usare le nuove funzioni di ASP.NET, sarà sufficiente cambiare l'estensione da .asp a .aspx e aggiornare tutti i collegamenti di conseguenza. Sarà quindi possibile migrare le applicazioni una pagina alla volta.
- Debug reale. In ASP.NET, il debug è più semplice che in ASP classico. E' stato aggiunto un comando trace compilato solo nel codice in esecuzione quando si imposta un flag di compilazione. Con Visual Studio .NET si può esaminare passo per passo il codice ASP.NET, i file include, i controlli Web e i componenti .NET anche se ciascuno di essi usa un linguaggio di programmazione diverso.
- Linguaggi di programmazione reali. Anche se ASP supporta diversi linguaggi di scripting, ASP.NET e il framework .NET supportano qualsiasi linguaggio che può essere compilato in formato IL.
- Gestione degli errori reale. ASP.NET offre una migliore gestione degli errori, tra cui l'abilità di inviare gli errori di programmazione a una sola pagina di gestione degli errori, trasferendo anche tutti gli attributi della pagina. Disporre di una locazione centrale per la gestione degli errori evita di controllare gli errori su ogni riga di codice e scrivere un gestore di errori personalizzato per ogni caso. Inoltre Visual Basic .NET supporta ora la struttura try … catch.

- Distribuzione basata sulle directory. Migrare un'applicazione ASP da un server all'altro è un'operazione complicata. Le estensioni di FrontPage, i componenti COM e le impostazioni del Web server sono separati dai file nella directory da spostare. Con ASP.NET, si può distribuire l'applicazione completa di componenti e impostazioni del server, usando XCOPY o FTP. Questo semplifica il beckup dei siti e elimina molti dei problemi relativi all'hosting Web remoto.
- Configurazione di applicazioni basata su file. L'amministrazione dell'applicazione può avvenire interamente mediante file di configurazione XML senza necessità di alcun accesso diretto al server.
- Modello di programmazione basato sugli eventi. La pagine ASP sono semplici script che iniziano l'esecuzione all'inizio del file e continuano riga per riga fino a raggiungere la fine dello script. Al contrario, le pagine ASP.NET seguono un modello di programmazione basato su eventi.
- Modello ad oggetti migliorato ed estensibile. I Web form possono essere creati con di una interfaccia drag-and-drop semplificando la creazione di pagine guidate agli eventi. Inoltre, il framework .NET include un elenco completo di classi disponibili per gli sviluppatori non incluse negli oggetti di ASP 3.0.
- Più funzioni integrate. Non è più necessario accedere esplicitamente alle variabili modulo usando l'oggetto request. Tutto questo è gestito automaticamente da ASP.NET. E' sufficiente aggiungere runat="server" al modulo e a ogni elemento del modulo. Le funzioni di convalida dei moduli sono incorporate in ASP.NET, che viene fornito con le funzioni di convalida più comuni e consente agli sviluppatori di scrivere controlli di convalida personalizzati.
- Servizi Web. ASP.NET include supporto per i servizi Web, che consentono alle applicazioni di funzionare assieme su Internet visualizzando e/o utilizzando metodi e dati di altri siti. I servizi Web usano lo standard SOAP (Simple Object Access Protocol)
- Miglioramenti nelle prestazioni. Le prestazioni di ASP.NET sono migliori di quelle di ASP classico. Il vantaggio principale è che ASP.NET è compilato, non interpretato. ASP.NET supporta il caching a livello di pagina, che può essere configurato di pagina in pagina.
- Strumenti migliori. Non esistono più strumenti separati per ogni linguaggio supportato da Microsoft. Fa tutto Visual Studio .NET
  Per maggiori informazioni su ASP.NET vedi bibliografia [6].

2.4 COME VENGONO ELABORATE LE PAGINE ASP.NET

In generale, il ciclo di vita di una pagina Web con un Forms è simile a quello che ogni processo Web svolge nel server. Certe caratteristiche dei processi Web (informazioni comunicate con il protocollo HTTP, la natura senza stato delle pagine Web, ecc.) sono applicabili alle pagine Web con Form.

Comunque, il framework delle pagine ASP.NET fornisce molti servizi per applicazioni Web. Per esempio, il framework delle pagine ASP.NET cattura le informazioni inviate con i Form delle pagine Web, estrae i valori rilevanti, e produce informazioni accessibili tramite proprietà di oggetti.

È importante capire la sequenza degli eventi scatenati quando un From di una pagina Web viene elaborato.

ROUND TRIPS

Una delle cose più interessanti è la divisione del lavoro nei form nelle pagine Web. Il browser presenta il form all'utente, e l'utente interagisce con il form, causando dei post back al server. Tutti i processi che interagiscono con i componenti server devono essere elaborati nel server. Questo significa che per ogni azione che richiede di essere elaborata, deve essere inviata al server, elaborata e restituita al browser. Questa sequenza di eventi è chiamata round trip.

Da notare che nei form delle pagine Web si possono creare script client, che sono usati per convalidare gli input dell'utente e che possono essere elaborati senza essere inviati al server perché non richiedono l'interazione con i componenti server.

Consideriamo un esempio nel mondo del business. Un utente inserisce un ordine e conferma un numero sufficiente di prodotti per quell'ordine, quindi la sua applicazione invia la pagina a un server appropriato per elaborare l'ordine. Il processo server esamina l'ordine, scorre l'inventario, svolge alcune azioni definite dalla logica di business e restituisce la pagina al browser.

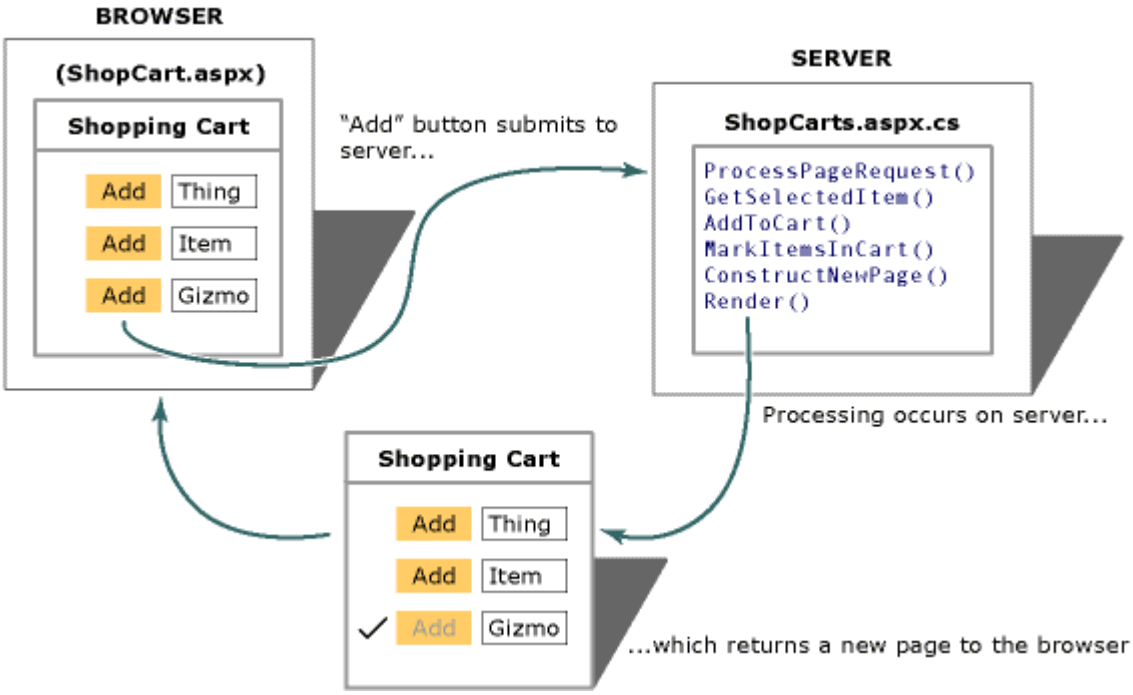

Figura 4 - ASP.NET Round Trip

Nei forms Web, molte azioni degli utenti, come cliccare un bottone, scatenano un round trip. Per questa ragione, gli eventi disponibili nei controlli server di ASP.NET sono solitamente limitati ai click. Molti controlli server espongono l'evento click, che richiede però una gestione da parte dell'utente.

Per la stessa ragione, i controlli server non espongono eventi usati di frequente come onmouseover, perché tutte le volte che vengono generati, viene eseguito un round trip, che ha considerevoli effetti sulla risposta del form.

RICREAZIONE DELLE PAGINE (VIEW STATE E STATE MANAGEMENT)

Nello scenario del Web, le pagine sono ricaricate ad ogni round trip. Non appena il server finisce di elaborare e spedire una pagina al browser, elimina le informazioni sulla pagina, per liberare le sue risorse dato che deve supportare centinaia o migliaia di utenti contemporanei. La volta successiva che viene inviata la pagina, il server crea un nuovo processo per elaborarla, e per questa ragione la

pagina Web viene chiamata senza stato (stateless) e i valori delle variabili e dei controlli della pagina non vengono preservati nel server.

Da notare che il server può essere configurato per memorizzare le informazioni delle pagine in una cache per ottimizzarne l'esecuzione, ma non allo scopo di supporto alla programmazione delle pagine.

Nelle applicazioni Web tradizionali, le sole informazioni che il server ha sul form sono le informazioni che l'utente ha inserito nei controlli del form, perché esse sono inviate al server quando il form viene spedito. Le altre informazioni, come i valori delle variabili e le proprietà settate, sono perse.

ASP.NET lavora all'interno di queste limitazioni con le seguenti modalità:

- Salva le proprietà della pagina e dei controlli ad ogni round trip. Questa tecnica è conosciuta come salvare il view state del controllo.
- Fornisce facilitazioni nella gestione dello stato per cui si possono salvare le variabili e le informazioni relative alla applicazione o alla sessione ad ogni round trip.
- Può rilevare quando un form è richiesto per la prima volta o quando è stato già inviato.

BENEFICI DEL MODELLO A EVENTI RISPETTO AL MODELLO LINEARE

Un programmatore che ha esperienza nello scrivere pagine usando ASP, saprà che ASP utilizza un modello di elaborazione delle pagine lineare. Una pagina ASP viene elaborata in sequenza dall'alto verso il basso. Ogni linea di codice ASP e di HTML statico viene elaborata in sequenza come appare nel file. Un'azione dell'utente causa l'invio della pagina al server in un round trip. Quando questa azione causa un round trip, il server ricrea la pagina. Dopo che la pagina è stata ricreata, viene elaborata nella stessa sequenza dall'alto verso il basso come in precedenza. Per creare un metodo di elaborazione delle pagine a eventi, è necessario avere a disposizione un metodo per mantenere lo stato della pagina e dei controlli. Questo limite del modello limita la ricchezza dell'interfaccia utente che viene assemblata, e incrementa la complessità del codice necessario per gestirla.

A confronto, il modello a eventi, come quello delle tradizionali applicazioni Visual Basic, contiene degli elementi programmabili che vengono inizializzati e visualizzati nel form. L'utente interagisce con essi, causando degli eventi che vengono ascoltati da un ascoltatore degli eventi. Questo modello supporta un vero modello ad eventi, che estende la ricchezza dell'interfaccia utente che viene assemblata, e riduce la complessità del codice necessario per gestirla.

ASP.NET sostituisce il modello lineare di processare le pagine di ASP con una emulazione del modello ad eventi. Il framework delle pagine ASP.NET è fornito di un implicito produttore di associazioni tra un evento e il suo ascoltatore. Usando il framework si possono creare semplicemente interfacce utente che reagiscono alle azioni dell'utente.

In più, il framework semplifica l'implementazione della gestione dello stato della pagina e dei controlli.

Per esempio, ASP.NET permette di settare un ascoltatore degli eventi nel codice del server per un evento passato dal browser. Assumiamo che l'utente stia interagendo per un Form Web che contiene un controllo server bottone. L'utente clicca nel controllo bottone e viene generato un evento che viene trasmesso con un post HTTP al server dove il framework della pagina ASP.NET interpreta le informazioni inviate e associa l'evento scatenato con l'appropriato ascoltatore di eventi. Questo ascoltatore di eventi può essere l'ascoltatore di eventi di default fornito da ASP.NET o può essere una implementazione personalizzata. Il framework automaticamente chiama l'appropriato ascoltatore degli eventi per il bottone premuto.

FASI DI ELABORAZIONE DI UNA PAGINA ASP.NET

Il framework delle pagine ASP.NET elabora i from Web in diversi passaggi. Durante ogni fase dell'elaborazione dei From Web, sono scatenati degli eventi, e per ogni evento scatenato viene chiamato il rispettivo ascoltatore. Questo metodo fornisce la possibilità di aggiornare il contenuto di un Form di una pagina Web.

La tabella che segue elenca i più comuni fasi di come vengono elaborate le pagine, gli eventi generati, quando vengono generati, e il loro uso tipico. Questi passaggi vengono ripetuti tutte le volte che un form viene richiesto o inviato. La proprietà Page.IsPostBack permette al programmatore di testare quando la pagina viene processata per la prima volta.

Passaggio	Significato	Uso tipico
ASP.NET Page Framework Initialization	L'evento della pagina Page_Init viene scatenato e il view state della pagina e dei controlli sono ricaricati.	Durante questi evento, il framework delle pagine ASP.NET, ricarica le proprietà dei controlli e dei dati dei postback.
User Code Initialization	L'evento della pagina Page_Load viene scatenato	Leggere e ricaricare i valori salvati precedentemente.
Validation	Il metodo Validate di ogni controllo server validator viene invocato per eseguire i controlli specificati.	
Event Handling	Se la pagina viene chiamata in risposta a un evento, il corrispondente ascoltatore degli eventi viene chiamato durante questa fase.	Esegue le istruzioni per l'applicazione specifica.
Cleanup	L'evento Page_Unload viene chiamato perché la pagina è stata interpretata e le sue informazioni sono pronte per essere eliminate.	Esegue i lavori di pulizia finale come chiudere file, chiudere connessioni a database e eliminare oggetti.

Per maggiori informazioni su come vengono elaborate le pagine ASP.NET vedi bibliografia [7].

2.5 GESTIONE DELLO STATO DELLE PAGINE ASP.NET

Le pagine Web sono ricreate ogni volta che sono spedite al server. Nella programmazione Web tradizionale, questo significa che tutte le informazioni associate alla pagina e ai controlli contenuti nella pagina vengono perse ad ogni round trip. Per esempio, se un utente inserisce informazioni in una textbox, questa informazione viene persa nel round trip tra il browser al server.

Per superare questa limitazione intrinseca della programmazione tradizionale Web, il framework delle pagine ASP.NET include varie opzioni per aiutare il programmatore a preservare lo stato. Il framework delle pagine include una facilitazione, chiamata view state che automaticamente preserva il valore di una proprietà della pagina o di un controllo tra i round trip. Comunque, è probabile avere a che fare con specifiche applicazioni che devono preservare particolari valori. Per fare questo, si può usare una delle modalità per la gestione dello stato.

Alcune di queste opzioni comportano la memorizzazione delle informazioni nel client, ad esempio direttamente nella pagina o in un cookie, altre comportano la memorizzazione nel server delle informazioni tra un round trip e il successivo.

VIEW STATE

La proprietà Control.ViewState fornisce un dizionario di oggetti per conservare valori tra più richieste della stessa pagina. Questo è il metodo che le pagine utilizzano per preservare i valori delle proprietà della pagina e dei controlli tra i round trip.

Quando la pagina viene elaborata, lo stato corrente della pagina e dei controlli vengono memorizzate in una stringa e salvati come un campo nascosto. Quando la pagina viene rispedita dal server, la pagina scansiona la stringa di view state durante la sua inizializzazione e ripristina le proprietà salvate.

CAMPI NASCOSTI

ASP.NET permette di usare i campi nascosti di HTML nei form. Un campo nascosto non viene reso visibile nel browser, ma è possibile settarne il valore come per un controllo standard. Quando la pagina viene spedita al server, il contenuto del campo nascosto viene spedito nel form con i valori degli atri controlli.

Per quanto riguarda la sicurezza, c'è da dire che per i malintenzionati è facile vedere e modificare il contenuto di un campo nascosto.

Il campo nascosto memorizza una singola variabile nella sua proprietà value e deve essere aggiunto esplicitamente al contenuto della pagina. Dopo aver inserito il campo nascosto, ASP.NET fornisce il controllo HtmlInputHidden che offre tutte le funzionalità del campo nascosto.

COOKIES

Un cookie è una piccola quantità di dati memorizzata all'interno di un file di testo nel file system del client. Esso contiene delle specifiche informazioni della pagina che il server spedisce al client insieme alla pagina. I cookie possono essere temporanei o persistenti.

Si possono usare cookie per memorizzare informazioni riguardanti un particolare client, o una particolare sessione o una particolare applicazione. Il cookie viene salvato sul client e quando il browser richiede la pagina, il client invia le informazioni del cookie insieme con la richiesta. Il server può leggere il cookie e estrarre il suo valore. Un suo uso tipico è quello per memorizzare se un utente si è già autenticato nella applicazione.

QUERY STRING

La query string sono le informazioni che vengono appese alla fine dell'URL della pagina. Un tipico esempio potrebbe essere il seguente:

http://www.contoso.com/listwidgets.aspx?category=basic&price=100

Nel precedente URL, la query string parte con il simbolo di punto interrogativo (?) e include due coppie attributo valore, una chiamata category e l'altra chiamata price.

La query string fornisce un semplice, ma limitato, modo per mantenere alcune informazioni di stato. Per esempio, è un semplice modo per passare delle informazioni tra una pagina e un'altra. Comunque, molti browser impongono un limite di 255 caratteri alla lunghezza dell'URL.

Per quanto riguarda la sicurezza, le informazioni passate tramite query string possono essere modificate da utenti malintenzionati, quindi non possono essere usate per passare dati importanti.

APPLICATION STATE

ASP.NET permette di salvare valori usando lo stato di applicazione. Lo stato di applicazione è un meccanismo di memorizzazione accessibile da tutte le applicazioni Web ed è molto usato per salvare informazioni che necessitano di essere mantenute tra round trip e tra pagine.

Lo stato di applicazione è un vocabolario creato durante ogni richiesta a uno specifico URL. Si possono aggiungere le informazioni della specifica applicazione alla struttura per memorizzarle per ogni richiesta di pagina.

SESSION STATE

ASP.NET permette di salvare valori usando lo stato di sessione. Lo stato di sessione è simile allo stato di applicazione, eccezion fatta che la sua vita è limitata a una sessione corrente del browser. Se utenti diversi usano la stessa applicazione, ognuno di loro avrà uno stato di sessione diverso. In più, se lo stesso utente lascia l'applicazione per poi tornarvi in seguito avrà comunque uno stato di sessione diverso..

Lo stato di sessione è un vocabolario per memorizzare informazioni della sessione specifica che è necessario mantenere tra un round trip e il successivo e tra richieste di pagine differenti.

Lo stato di sessione permette di:

- Un identificatore unico della richiesta del browser e di mappare essa in una sessione individuale nel server.
- Memorizzare dati relativi alla specifica sessione nel server.

DATABASE SUPPORT

Mantenere lo stato usando un database è pratica comune quando si memorizzano informazioni specifiche per ogni utente e queste informazioni sono molte. La memorizzazione in database è particolarmente usata per mantenere informazioni a lungo termine e che devono essere mantenute anche se il server viene riavviato.

Per maggiori informazioni sulla gestione dello stato delle pagine ASP.NET vedi bibliografia [8].

3. LE TECNOLOGIE UTILIZZATE: RDF

RDF (Resource Description Framework) è un linguaggio per la descrizione delle informazioni delle risorse del Web. E' stato creato con il particolare intento di rappresentare i metadati delle risorse Web come il titolo, l'autore, la data dell'ultima modifica di una pagina Web, le informazioni sulle licenze e sui copyright di un documento Web o la disponibilità di risorse condivise. Comunque, dalla generalizzazione del concetto di risorsa Web, RDF può anche essere usato per rappresentare informazioni riguardanti tutte le cose che possono essere identificate nel Web come, ad esempio, nel caso di un sito per lo shopping on-line, si possono rappresentare informazioni sui prodotti (es: informazioni riguardanti le specifiche, il prezzo e la disponibilità) o la descrizione delle informazioni sul metodo di consegna preferito da un utente.

RDF è stato pensato per quelle situazioni in cui le informazioni devono essere elaborate da un'applicazione per essere presentate a una persona. RDF fornisce una struttura per esprimere queste informazioni per essere scambiate tra applicazioni senza perdite di significato.

RDF è basato sull'idea di identificare qualcosa usando identificatori Web, chiamati URI (Uniform Resource Identifiers), e descrivere le risorse in termini di coppie proprietà – valore. Questo permette a RDF di rappresentare semplici dichiarazioni di risorse come un grafo di nodi e di archi che rappresentano le proprietà e i valori della risorsa. Ad esempio la frase "C'è una persona identificata da http://www.w3.org/People/EM/contact#me, il cui nome è Eric Miller, il cui indirizzo email è em@w3.org e il cui titolo è Dr." può essere rappresentata da un grafo RDF come in figura.

Figura 5 - RDF - Primo esempio

La figura mette in evidenza che RDF usa URI per identificare:

- Individui (es: Eric Miller è identificato da http://www.w3.org/People/EM/concat#me).
- Tipi di cose (es: le persone sono identificate da http://www.w3.org/2000/10/swap/pim/contact#Person).
- Proprietà di queste cose (es: mailbox è identificato da http://www.w3.org/2000/10/swap/pim/contact#mailbox)
- Valori di queste proprietà (es: malto:em@w3.org è il valore della proprietà mailbox.

RDF fornisce una sintassi basata su XML, chiamata RDF/XML, per registrare e scambiare le informazioni contenute in questo grafo. Ad esempio, considerando il grafo precedente, il corrispondente codice in RDF/XML risulta:

```
<?xml version=”1.0”?>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
                                    xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
        <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
                <contact:fullName>Eric Miller</contact:fullName>
                <contact:mailbox rdf:resource="mailto:em@w3.org"/>
                <contact:personalTitle>Dr.</contact:personalTitle>
        </contact:Person>
</rdf:RDF>
```

Come HTML, RDF/XML è elaborabile dalle macchine e, usando degli URI, si possono collegare informazioni attraverso il Web.

Comunque, al contrario di un ipertesto convenzionale, gli URI di RDF possono riferire a ogni cosa identificabile, incluse cose non direttamente riferibili dal Web. Il risultato è che, in aggiunta alle

descrizioni di elementi che sono pagine Web, RDF può anche descrivere auto, persone, novità, eventi, ecc. In più, le proprietà RDF hanno al loro interno un URI che precisa l'identificazione della relazione che esiste tra le due cose collegate.

3.1 CREARE DICHIARAZIONI DI RISORSE

Immaginiamo che qualcuno di nome John Smith crei una particolare pagina Web. Una semplice via per esprimerlo in un linguaggio naturale come l'Italiano è

http://www.example.org/index.html ha un creatore il cui valore è John Smith

La frase è stata enfatizzata per illustrare le varie parti che specifica:

- La cosa che la dichiarazione descrive (nell'esempio la pagina Web).
- Una specifica proprietà della cosa che la dichiarazione descrive (nell'esempio creatore).
- Il valore della proprietà della cosa che la dichiarazione descrive (nell'esempio John Smith).

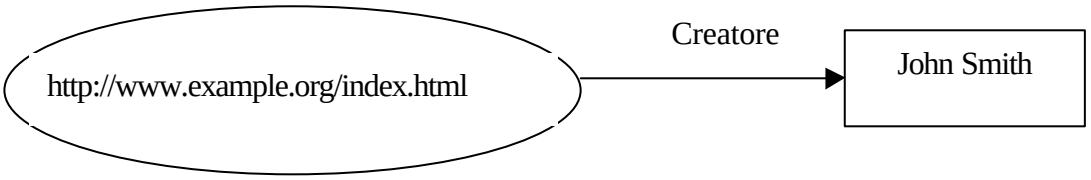

Figura 6 - RDF- Secondo esempio

I termini con cui RDF identifica le varie parti della dichiarazione sono:

- Il soggetto è l'indirizzo Web http://www.example.org/index.html.
- Il predicato è la parola "creatore".
- L'oggetto è il nome "John Smith".

Figura 7 - RDF - Soggetto, predicato e oggetto

IL MODELLO RDF

Considerando l'esempio precedente:

http://www.example.org/index.html ha un creatore il cui valore è John Smith

può essere rappresentato da una dichiarazione RDF con:

- Un soggetto http://www.example.org/index.html.
- Un predicato http://purl.org/dc/elements/1.1/creator.
- Un oggetto http://www.example.org/staffid/85740.

Da notare come i riferimenti URI sono usati per identificare non solo il soggetto della dichiarazione, ma anche il predicato e il soggetto invece di usare le parole “creatore” e “John Smith”.

Il modello RDF rappresenta la dichiarazione con un grafo composto di nodi e archi. Con questa notazione, la dichiarazione viene rappresentata da un nodo per il soggetto, da un nodo per l’oggetto e da un arco per il predicato, diretto dal soggetto verso l’oggetto.

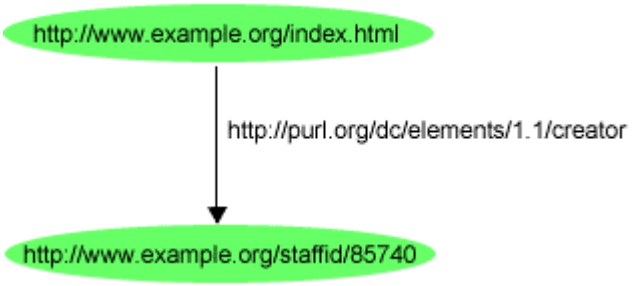

Figura 8 - RDF - Esempio con URI

Un gruppo di dichiarazioni possono essere rappresentate con un gruppo di nodi e di archi. Se aggiungiamo le seguenti due dichiarazioni all’esempio precedente:

http://www.example.org/index.html ha una data di creazione il cui valore è 16 agosto 1999.
http://www.example.org/index.html ha una lingua il cui valore è Inglese.

Otterremo il grafo RDF che segue:

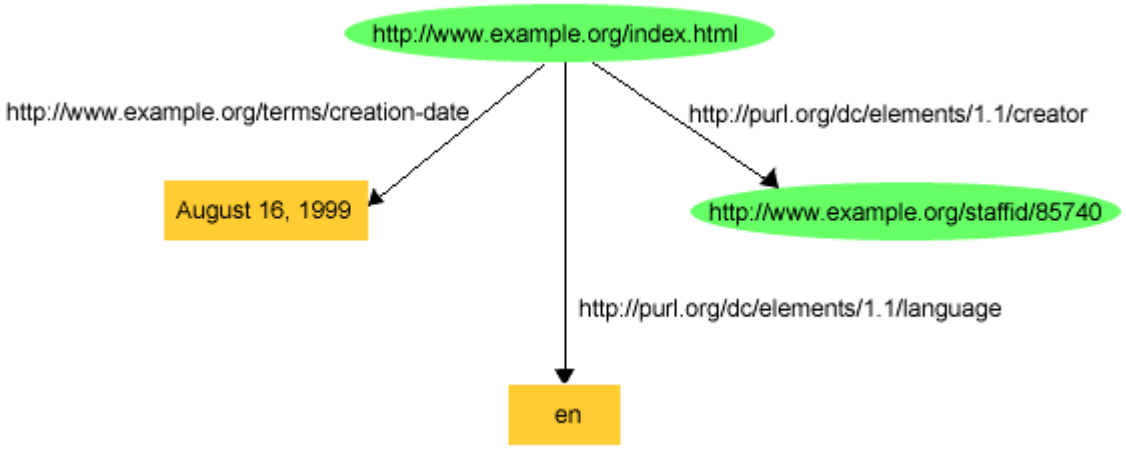

Figura 9 - RDF - Esempio con più dichiarazioni

Alcune volte non è conveniente disegnare il grafo. Può essere usata una via alternativa per scrivere le dichiarazioni, chiamata a terne. In questa notazione, ogni dichiarazione viene scritta come una terna con soggetto, predicato e oggetto, in questo ordine. Per esempio, le dichiarazioni precedenti possono essere scritte come:

```
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/creator>
                                                    <http://www.example.org/staffid/85740>

<http://www.example.org/index.html> <http://www.example.org/terms/creation-date>
                                                                "16 agosto 1999"

<http://www.example.org/index.html>      <http://purl.org/dc/elements/1.1/language>      "en"
```

Ogni terna corrisponde a ogni arco nel grafo, completato con il nodo di inizio e con il nodo di fine. Al contrario della rappresentazione del grafo, ma come le dichiarazioni originali, la notazione a terne richiede che i nodi vengano identificati separatamente per ogni dichiarazione. La notazione a terne completa richiede che i riferimenti siano scritti completamente e racchiusi tra parentesi angolari. Esiste una via più breve per enunciare i riferimenti. Questa modalità utilizza un XML qualified name (o QName) senza le parentesi angolari come abbreviazione del riferimento URI. Il QName contiene un prefisso, detto anche namespace. Quindi, il riferimento completo è formato dal prefisso seguito dal local name. Ad esempio, se il QName foo è assegnato al namespace http://example.org/ somewhere/, il QName foo:bar è una scorciatoia per il riferimento http://example.org/somewhere /bar.

Riprendendo l'esempio precedente e usando i seguenti QName conosciuti:

- prefisso rdf:, namespace: http://www.w3.org/1999/02/22-rdf-syntax-ns#
- prefisso rdfs:, namespace: http://www.w3.org/2000/01/rdf-schema#
- prefisso dc:, namespace: http://purl.org/dc/elements/1.1/
- prefisso owl:, namespace: http://www.w3.org/2002/07/owl#
- prefisso ex:, namespace: http://www.example.org/
- prefisso xsd:, namespace: http://www.w3.org/2001/XMLSchema#

Utilizzando anche i seguenti prefissi necessari per l'esempio e derivati dal prefisso ex:

- prefisso exterms:, namespace: http://www.example.org/terms/
- prefisso exstaff:, namespace: http://www.example.org/staffid/

La notazione a terne può essere riscritta come:

ex:index.html	dc:creator	exstaff:85740 .
ex:index.html	exterms:creation-date	"August 16, 1999"
ex:index.html	dc:language	"en" .

Quando RDF usa riferimenti URI invece di nomi di cosa nelle dichiarazioni, RDF riferisce a un insieme particolare di riferimenti utilizzati per quell'applicazione chiamato vocabolario. Spesso, i riferimenti contenuti nel vocabolario sono organizzati per essere rappresentati come un insieme di QName che usano un prefisso comune.

VALORI DI PROPRITA' STRUTTURATI E NODI VUOTI

Risorse che sono molto semplici possono essere rappresentate da proprietà semplici. Comunque, molti dati reali comportano strutture dati più complicate. Per esempio, in un sito Web la proprietà data di creazione può venire memorizzata come una proprietà unica come nell'esempio del paragrafo precedente, o essere memorizzata come un giorno, un mese e un anno memorizzati separatamente. Un altro esempio è la necessità di registrare un indirizzo. In questo caso, l'indirizzo di John Smith potrebbe essere così descritto:

exstaff:85740	exterms:address	"1501 Grant Avenue, Bedfod, Massachutts 01630"

Supponiamo la necessità di memorizzare l'indirizzo di John Smith come una struttura contente una strada, una città, uno stato e un numero di avviamento postale. Come si può rappresentare in RDF? Informazioni strutturate come questa possono essere rappresentate in RDF considerando la proprietà aggregata descritta come risorsa, e creare nuove dichiarazioni per definire le proprietà singole. Nel grafo RDF, l'indirizzo di John Smith viene scomposto come un nuovo nodo che rappresenta il concetto di indirizzo di John Smith con un nuovo URI per identificarlo (nell'esempio

http://www.example.org/addressid/85740 abbreviato con exaddressid:85740 e nuovi archi e nodi che a partire dal soggetto indirizzo descrivono le singole proprietà. Ad esempio:

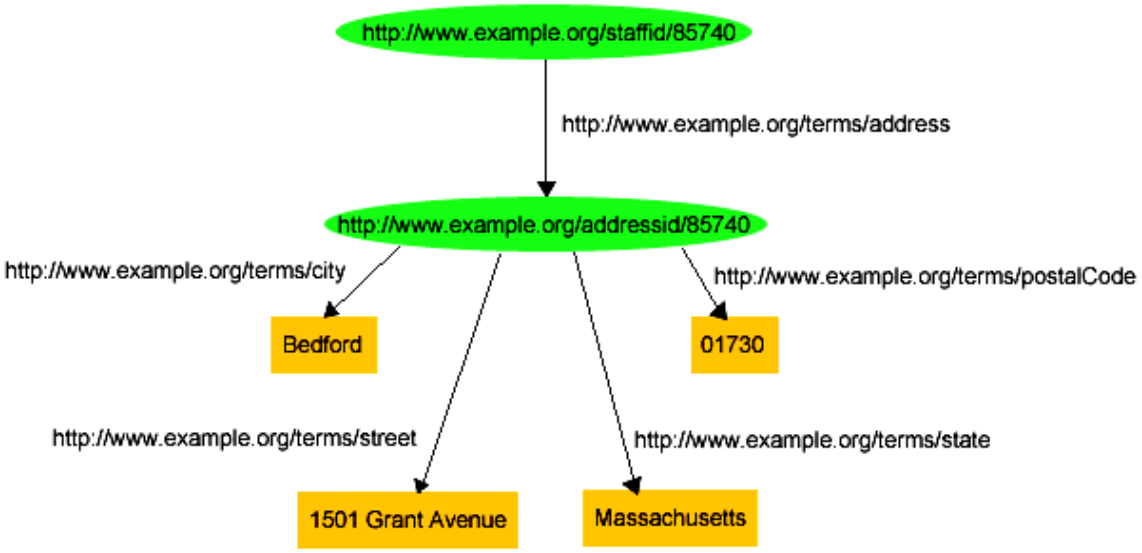

Figura 10 - RDF - Grafo con attributi strutturati

O con la notazione a terne:

exstaff:85740	exterms:address	exaddressid:85740
exaddressid:85740	exterms:street	"1501 Grant Avenue"
exaddressid:85740	exterms:city	"Bedford"
exaddressid:85740	exterms:state	"Massachusetts"
exaddressid:85740	exterms:postalCode	"01730"

Questa via per rappresentare informazioni strutturate in RDF può comportare la generazioni di numerosi riferimenti intermedi come, nell'esempio precedete exaddressid:85740, per rappresentare concetti aggregati. Questi concetti non devono essere referenziati direttamente dall'esterno del loro grafo e non sono richiesti identificatori universali. In oltre nel disegnare il grafo, questi concetti non sono indispensabili. In questo caso il grafo si può trasformare come segue:

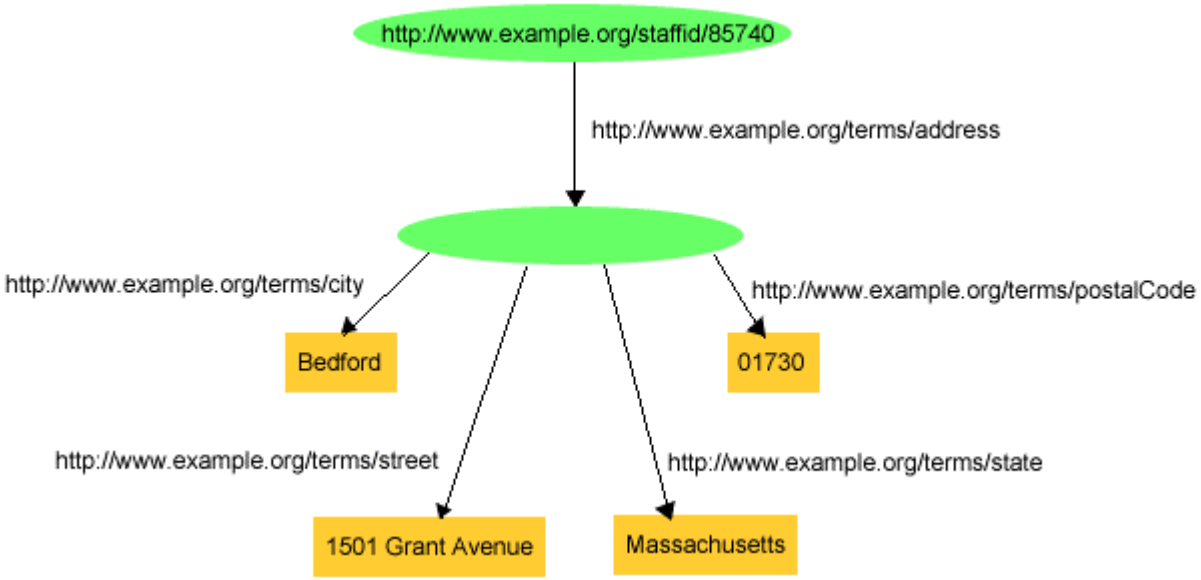

Figura 11 - RDF - Grafo con proprietà strutturate - 2 versione

La figura rappresenta un perfetto grafo RDF, che usa un nodo senza riferimento URI per rappresentare il concetto di indirizzo di John Smith. Questo nodo, chiamato nodo vuoto, è necessario per collegare nodi appartenenti a diverse parti del grafo. Se proviamo ora a tradurre il grafo nella notazione a terne ci accorgiamo di un problema:

exstaff:85740	exterms:address	??? .
???	exterms:street	"1501 Grant Avenue" .
???	exterms:city	"Bedford" .
???	exterms:state	"Massachusetts" .
???	exterms:postalCode	"01730" .

Dove ??? è qualcosa che indica la presenza del nodo vuoto. Siccome grafi complessi possono contenere più di un nodo vuoto, occorre un modo per differenziare un nodo vuoto da un altro nodo vuoto. Il risultato è utilizzare identificatori per i nodi vuoti con la sintassi _:name. Con questa sintassi, nel nostro esempio possiamo scrivere:

exstaff:85740	exterms:address	_:johnaddress .
_:johnaddress	exterms:street	"1501 Grant Avenue" .
_:johnaddress	exterms:city	"Bedford" .
_:johnaddress	exterms:state	"Massachusetts" .
_:johnaddress	exterms:postalCode	"01730" .

Nella rappresentazione a terne del grafo, ogni nodo vuoto distinto nel grafo deve avere un differente identificatore di nodo vuoto. Al contrario dei riferimenti URI, gli identificatori di nodo vuoto non sono considerati come parte del grafo RDF. Gli identificatori di nodo vuoto sono la giusta modalità per rappresentare nodi vuoti quando vengono scritti con il metodo a terne. Se si ci aspetta che un nodo in un grafo possa essere riferito dall'esterno, esso sarà identificato da un riferimento URI. In conclusione, siccome gli identificatori di nodi vuoti rappresentano nodi, possono essere usati come soggetto o come oggetto di una terna, ma non come predicato.

Nodi vuoti forniscono, anche, una via per dare definizioni di risorse che non hanno un URI, ma che sono descritte in termini di relazioni con altre risorse che hanno un URI. Per esempio, bisogna memorizzare le informazioni sull'indirizzo email di Jane Smith, ma quest'ultima non ha una propria home page (al contrario di John). Jane non ha, quindi, un proprio URI. Un primo approccio potrebbe essere quello di rappresentare Jane con il proprio indirizzo email. In questo caso il problema si potrebbe risolvere così, ma sarebbe una descrizione assai poco accurata e, inoltre, se volessimo rappresentare anche l'indirizzo reale di Jane non andrebbe più bene. In questi casi, si utilizza un nodo vuoto per rappresentare la risorsa senza URI e le singole proprietà vengono memorizzate in nodi distinti. Nel nostro esempio:

_:jane	exterms:mailbox	<mailto:jane@example.org> .
_:jane	rdf:type	exterms:Person .
_:jane	exterms:name	"Jane Smith" .
_:jane	exterms:empID	"23748" .
_:jane	exterms:age	"26" .

Da notare come per indicare l'URI mailto:jane@example.org siano state necessarie le parentesi angolari. Infatti se fossero state omesse mailto potrebbe essere confuso come un prefisso di un QName.

In pratica, in alcuni casi, l'utilizzo di nodi vuoti non viene sempre usato. Infatti, per esempio la dichiarazione "l'autore del libro è malto:jane@example.org" è usata al posto di "l'autore del libro è qualcuno il cui indirizzo email è malto:jane@example.org" dove qualcuno rappresenta il nodo vuoto e fornisce solo un metodo più accurato per descrivere una situazione reale.

Nell'esempio precedente, Jane aveva un'età di 26. In questo caso, l'età di 26 anni è stata espressa omettendo anni che era ovvio. In conclusione, nodi vuoti e proprietà strutturate conviene utilizzarle solo nei casi in cui il concetto senza queste costruzioni risulta confuso o può generare fraintendimenti. Ad esempio, l'unità di misura per l'altezza non può essere omessa in quanto il sistema europeo e quello americano utilizzano unità di misura diversi.

TIPIZZAZIONE

Nella sezione precedente, tutti i valori di proprietà non avevano un tipo. Vediamo ora come indicare il tipo di un valore. Consideriamo, ad esempio, il grafo mostrato in figura.

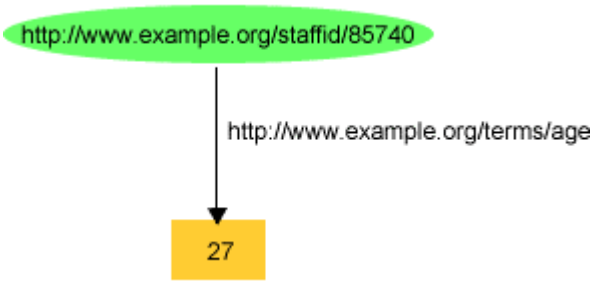

Figura 12 - RDF – Esempio grafo con proprietà non tipizzata

In questo caso, l'ipotetica organizzazione example.org probabilmente intende per "27" il numero e non la stringa formata dal carattere "2" e dal carattere "7". Nella figura successiva viene riportato il grafo con l'indicazione esplicita che "27" deve essere interpretato come un numero.

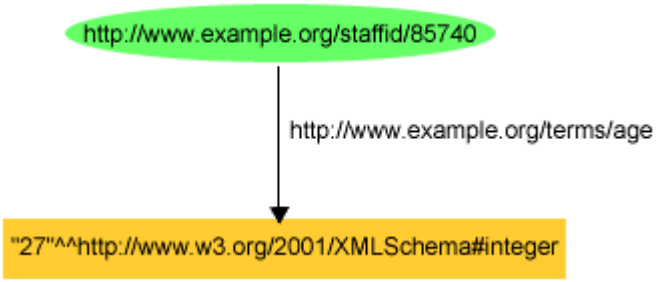

Figura 13 - RDF – Esempio Grafo con proprietà tipizzata

La pratica comune nei linguaggi di programmazione o dei database systems è fornire informazioni addizionali su come interpretare i dati. Ad esempio, una applicazione riconoscerà se il numero "10" è inteso come il numero 10 (decimale) o il numero 2 (binario) a seconda che il tipo di dati sia integer o binary.

Un tipo di dato RDF è formato da un riferimento URI che identifica il particolare tipo di dato. Ad esempio traducendo il grafo precedente con il metodo a terne otteniamo:

```
<http://www.example.org/staffid/85740>  <http://www.example.org/terms/age>
                                        27"^^<http://www.w3.org/2001/XMLSchema#integer>
```

O utilizzando la notazione abbreviata

```
exstaff:85740  exterms:age  "27"^^xsd:integer .
```

Per maggiori informazioni sulla sintassi RDF vedi bibliografia [5].

3.2 TIPI DI DATO IN RDF

I concetti di tipi di dato RDF sono basati sulla struttura concettuale dei dati di XML Schema. Questa struttura concettuale definisce i tipi di dato come costituiti da:

- Un insieme di valori, chiamato spazio dei valori, che un tipo di dato intende rappresentare. Per esempio il tipo di dato xsd:date avrà uno spazio dei valori formato da un insieme di date.
- Un insieme di stringhe, chiamato spazio lessicale, che il tipo di dato utilizza per rappresentare i suoi valori. Per esempio, il tipo di dato xsd:date definisce "1999-08-16" come un metodo legale per scrivere 16 agosto 1999.
- Una mappa tra lo spazio lessicale e quello dei valori. Questa mappa determina quale valore è associato alla particolare stringa dello spazio lessicale per quel particolare tipo di dati. Per esempio, la mappa per il tipo di dato xsd:date determina che, per questo tipo di dato, la stringa "199-08-16" rappresenta la data 16 agosto 1999. La mappa tra lo spazio lessicale e quello dei valori è un fattore che fa si che la stessa stringa possa rappresentare valori differenti per differenti tipi di dati.

Un insieme di coppie costituite da un riferimento URI e il relativo tipo di dato associato dove nessun riferimento URI compare in due coppie diverse viene chiamato mappa di tipi di dato. Tutte le mappe di tipi di dato devono contenere la coppia <rdf:XMLLiteral,X> dove X è il tipo di dato che rappresenta un contento di tipo XML. La mappa dei tipi di dato deve contenere anche un insieme di coppie <http://www.w3.org/2001/XMLSchema#SSS, SSS>, dove SSS è un tipo di dato chiamato SSS e contenuto nei tipi di dati di XML Schema.

I tipi di dato possono essere ulteriormente divisi in tipi di dato primitivi o derivati. I tipi di dato primitivi sono quelli che non sono definiti in termini di altri tipi di dato. I tipi di dato derivati sono quelli che sono definiti in termini di altri tipi di dato. Ad esempio, nella specifica dei tipi di dato per XML schema, il tipo di dato float è un concetto matematico ben definito senza ricorrere ad altri tipi di dato, al contrario la definizione di integer è data come caso particolare del tipo di dato decimal.

Esiste un tipo di dato concettuale, che si chiama anySimpleType che può essere considerato come il tipo base per tutti i tipi di dato primitivi. Lo spazio dei valori di anySimpleType si può immaginare come l'unione di tutti gli spazi dei valori di tutti i tipi di dato primitivi.

Tutti i tipi di dati derivati possono essere definiti come derivati da altri tipi di dato con una delle seguenti tre strade:

- Derivato da restrizione. Un tipo di dato si dice derivato da restrizione rispetto a un altro tipo di dato quando il suo spazio dei valori e/o il suo spazio lessicale sono ottenuti come sottoinsieme degli spazi del tipo di dato di partenza.
- Derivati da liste. Un tipo di dato si dice derivato da lista rispetto a un altro tipo di dato quando il suo spazio dei valori consiste in una lista finita di alcuni dei valori dello spazio dei valori del dato di partenza.
- Derivati da unioni. Un tipo di dato può essere derivato da uno o più tipi di dato mediante un'unione o una restrizione dei loro spazi di valori e, conseguentemente, dei loro spazi lessicali. Di seguito viene riportata la gerarchia di tutti i tipi di dato definiti in XML schema.

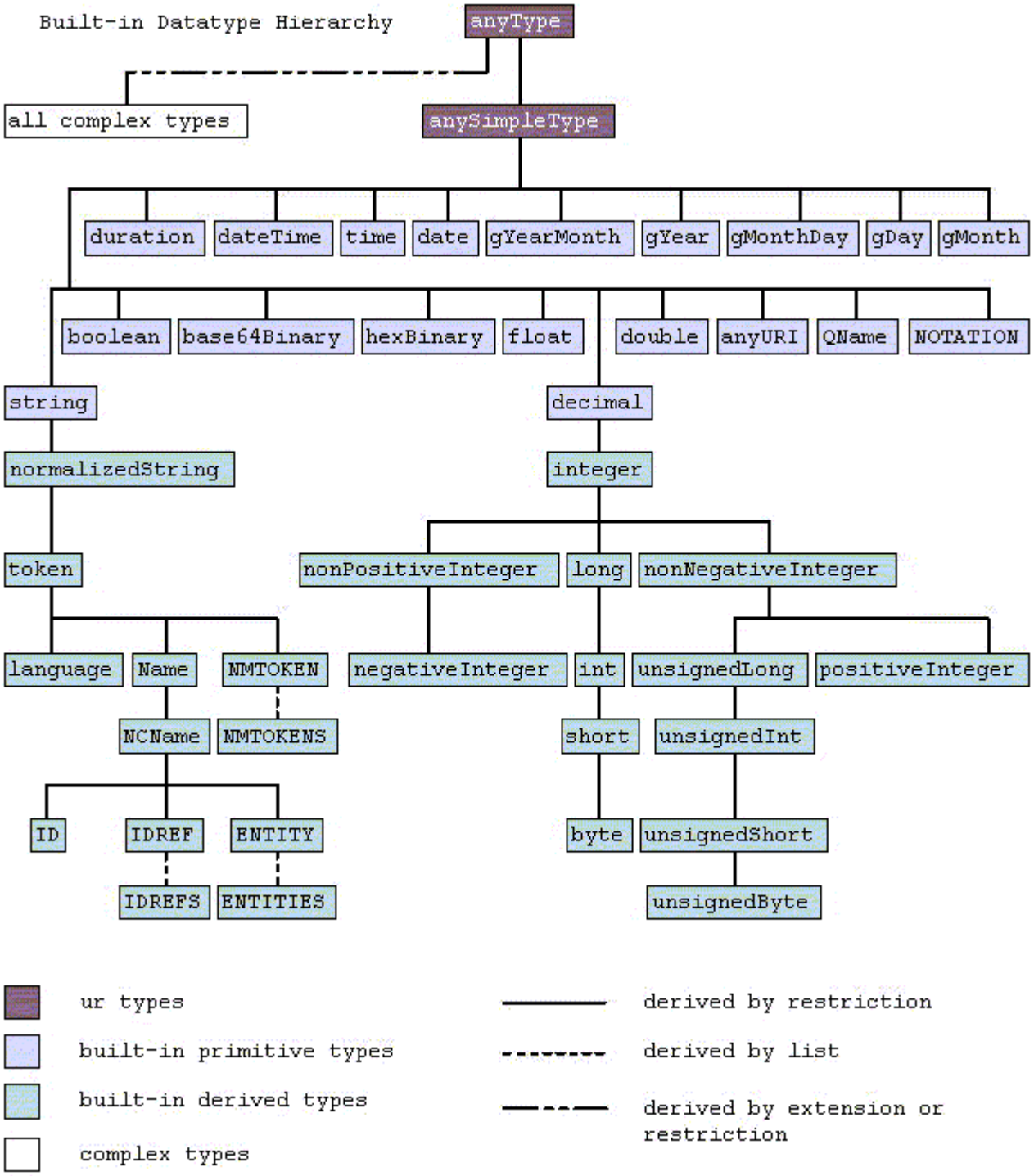

Figura 14 - RDF - Tipi di dato

Tutti i tipi di dato di questa specifica, sia primitivi che derivati, possono essere riferiti tramite un URI costruito nel seguente modo:

- La base dell'URI è l'URI del namespace di XML schema.
- Per identificare il tipo di dato va aggiunto il nome del tipo di dato.
  Ad esempio per identificare il tipo di dato int il suo URI è:

http://www.w3.org/2001/XMLSchema#int

Di seguito viene riportata una tabella con tutti i tipi di dato definiti nella specifica con una breve descrizione.

Nome	Tipo	Descrizione
xsd:string	primitivo	È una sequenza finita di caratteri.
xsd:boolean	primitivo	È il tipo di dato che supporta il concetto matematico di vero o falso. Definizioni legali di un valore boolean sono {true, false, 1, 0}
xsd:decimal	primitivo	È il tipo di dato che rappresenta numeri con precisione arbitraria.
xsd:float	primitivo	È il tipo di dato corrispondente alla definizione della IEEE di numero a virgola mobile a 32 bit.
xsd:double	primitivo	È il tipo di dato corrispondente alla definizione della IEEE di numero a virgola mobile a 64 bit.
xsd:dateTime	primitivo	È il tipo di dato che rappresenta uno specifico istante del tempo. La rappresentazione lessicale è quella della ISO 8601, cioè CCYY-MM-DDThh:mm:ss±hh:mm.
xsd:time	primitivo	È il tipo di dato che rappresenta uno specifico istante di tempo nell'arco di un giorno. La rappresentazione lessicale è del tipo hh:mm:ss.
xsd:date	primitivo	È il tipo di dato che rappresenta una data del calendario. La sua rappresentazione lessicale è yyyy-mm-dd.
xsd:gYearMonth	primitivo	È il tipo di dato che rappresenta uno specifico mese in uno specifico anno. Ad esempio per indicare luglio 2004 si scriverà 2004-07.
xsd:gYear	primitivo	È il tipo di dato che rappresenta un anno.
xsd:gMonthDay	primitivo	È il tipo di dato che rappresenta uno specifico giorno in un specifico mese. Ad esempio, per memorizzare che il mio compleanno è il 25 marzo scriverò 03-25.
xsd:gDay	primitivo	È il tipo di dato che rappresenta un giorno.
xsd:gMonth	primitivo	È il tipo di dato che rappresenta un mese.
xsd:hexBinary	primitivo	È il tipo di dato che contiene un arbitrario numero un formato esadecimale.
xsd:base64Binary	primitivo	È il tipo di dato che rappresenta un numero arbitrario in formato binario codificato in base 64.
xsd:anyURI	primitivo	È il tipo di dato che rappresenta un riferimento URI.
xsd:normalizedString	derivato	È il tipo di dato che rappresenta una stringa normalizzata, cioè che non contiene nessuno dei due caratteri di ritorno a capo e nessun tabulatore.
xsd:token	derivato	È il tipo di dato che rappresenta stringhe che non contengono line feed, tabulatori, spazi all'inizio o alla fine o una sequenza di due o più spazi all'interno della stringa.
xsd:language	derivato	È il tipo di dato che rappresenta una lingua del linguaggio naturale memorizzando la sua sigla.
xsd:NMTOKEN	derivato	È il tipo di dato che rappresenta una sequenza di stringhe di tipo xsd:token.
xsd:integre	derivato	È il tipo di dato che rappresenta un numero intero qualunque.
xsd:nonPositiveInteger	derivato	È il tipo di dato che rappresenta un numero intero

		qualunque negativo o nullo.
xsd:negativeInteger	derivato	È il tipo di dato che rappresenta un numero intero qualunque negativo.
xsd:long	derivato	È il tipo di dato che rappresenta un numero intero compreso tra -9.223.372.036.854.775.808 e +9.223.372.036.854.775.807 (8 byte).
xsd:int	derivato	È il tipo di dato che rappresenta un numero intero compreso tra -2.147.483.648 e +2.147.483.647 (4 byte).
xsd:short	derivato	È il tipo di dato che rappresenta un numero intero compreso tra -32.768 e +32.767 (2 byte).
xsd:byte	derivato	È il tipo di dato che rappresenta un numero intero compreso tra -128 e +127 (1 byte).
xsd:nonNegativeInteger	derivato	È il tipo di dato che rappresenta un numero intero qualunque positivo o nullo.
xsd:unsignedLong	derivato	È il tipo di dato che rappresenta interi positivi o nulli tra 0 e 18.446.744.073.709.551.615 (8 byte).
xsd:unsignedInt	derivato	È il tipo di dato che rappresenta interi positivi o nulli tra 0 e 4.294.967.295 (4 byte).
xsd:unsignedShort	derivato	È il tipo di dato che rappresenta interi positivi o nulli tra 0 e 65.535 (2 byte).
xsd:unsignedByte	derivato	È il tipo di dato che rappresenta interi positivi o nulli tra 0 e 255 (1 byte).
xsdPositiveInteger	derivato	È il tipo di dato che rappresenta un numero intero qualunque positivo.

Esistono altri tipi di dato in XML schema che, per varie ragioni, non vengono usati. Ad esempio, xsd:duration non ha uno spazio dei valori ben definito o xsd:NOTATION non è stato creato per l'uso diretto.

DEFINIZIONI DI NUOVI TIPI DI DATO

Per definire un tipo di dato bisogna:

- Definire lo spazio dei valori e lo spazio lessicale del tipo di dato.
- Definire un nome univo allo spazio dei valori e allo spazio lessicale.

La definizione di un tipo di dato segue lo schema seguente:

{name}
Opzionale. Nome del tipo di dato.
{target namespace}
Nome del namespace in cui è definito
{Variety}
Uno tra {atomic, list, union} e a seconda di quello scelto segue:

- Atomic: {primitive type definition} Un tipo di dato di XML schema.
- List: {item type definition} Un tipo di dato atomico o di tipo unione
- Union {member type definitons} Una sequenza non vuota di definizioni di tipi di dato.

{base type definition}
Se il tipo di dati derivato ha delle restrizioni rispetto al tipo di dato originale
{final}
Un sottoinsieme di {restriction, list, union} Specifica i metodi di derivazioni con cui un nuovo tipo di dato non potrà essere derivato da questo.

{annotation}
Opzionale. Annotazione

La sintassi XML per la dichiarazione di un nuovo tipo di dato è:

```
<simpleType
final = (#all | (list | union | restriction))
        name = NCName
        {any attributes with non-schema namespace . . .}>
        Content: (annotation?, (restriction | list | union))
</simpleType>
```

Consideriamo ora un esempio di nuovo tipo di dato derivato con restrizione dal tipo di dato string:

```
<simpleType name='Sku'>
        <restriction base='string'>
                <pattern value='\d{3}-[A-Z]{2}'/>
        </restriction>
</simpleType>
```

dove sku è un tipo di dato che rappresenta un codice a barre.

Consideriamo ora una semplice dichiarazione di un nuovo tipo di dato che rappresenti una lista di numeri decimali di tipo float:

```
<simpleType name='listOfFloat'>
        <list itemType='float'/>
</simpleType>
```

Per finire, consideriamo un esempio di dichiarazione di un tipo di dato derivato per unione. Il tipo di dato size che può essere: un numero intero positivo compreso tra 8 e 72 o una stringa tra “small”, “medium” e “large”. La dichiarazione potrebbe essere la seguente:

```
<xsd:simpleType name="size">
        <xsd:union>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:positiveInteger">
                                <xsd:minInclusive value="8"/>
                                <xsd:maxInclusive value="72"/>
                        </xsd:restriction>
                </xsd:simpleType>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                                <xsd:enumeration value="small"/>
                                <xsd:enumeration value="medium"/>
                                <xsd:enumeration value="large"/>
                        </xsd:restriction>
                </xsd:simpleType>
        </xsd:union>
</xsd:simpleType>
```

I criteri di restrizione che si possono utilizzare all’interno del tag <xsd:restriction> </xsd:restriction> sono elencati con una breve descrizione nella seguente tabella:

Nome	Descrizione
Length	Il tipo di dato deve contenere esattamente n unità di lunghezza, dove n è il valore assegnato. Per unità di lunghezza si intende per le stringhe e gli URI il numero di caratteri, per gli hexBinary e base64Binary il numero di ottetti e per i dati derivati da lista il numero di elementi.
minLength	Come il precedente ma assegna solo una quota minima di unità di lunghezza.
maxLength	Come il precedente ma assegna solo una quota massima di unità di lunghezza.
Pattern	Specifica una espressione regolare di filtraggio delle stringhe
Enumeration	Specifica un valore contenuto nello spazio dei valori di quel tipo di dato.
whiteSpace	Specifica come trattare una stringa. Se il valore di whiteSpace è preserve la stringa verrà memorizzata così comè, se il valore di whiteSpace è replace, tutti i caratteri uguali ai due del ritorno a capo o al tabulatore vengono sostituiti da uno spazio, se il valore di whiteSpace è collapse oltre a agire come il replace sostituisce tutti gli spazi bianchi ripetuti con uno spazio bianco solo.
maxInclusive	È il valore massimo che può assumere il tipo di dato secondo l'ordinamento definito su quel tipo di dato. Il valore massimo è accettato.
maxExclusive	È il valore massimo che può assumere il tipo di dato secondo l'ordinamento definito su quel tipo di dato. Il valore massimo non è accettato.
minExclusive	È il valore minimo che può assumere il tipo di dato secondo l'ordinamento definito su quel tipo di dato. Il valore minimo non è accettato.
minInclusive	È il valore minimo che può assumere il tipo di dato secondo l'ordinamento definito su quel tipo di dato. Il valore minimo è accettato.
totalDigits	È il numero massimo di cifre totali (cifre parte intera + cifre parte decimale) che può avere un tipo di dato derivato dal tipo di dato decimal.
fractionDigits	È il numero massimo di cifre decimali che può avere un tipo di dato derivato dal tipo di dato decimal.

3.3 SINTASSI XML PER RDF (RDF/XML)

Questa sezione introduce la sintassi RDF/XML per descrivere i grafi RDF. Consideriamo il seguente grafo RDF.

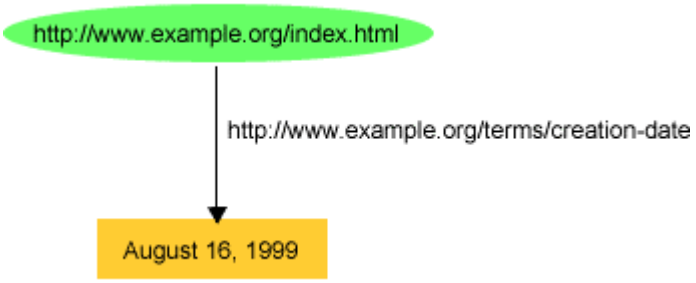

Figura 15 - RDF/XML - 1 Esempio

Rappresentato dalla seguente terna:

ex:index.html	exterms:creation-date	"August 16, 1999"

Il corrispondente codice in RDF/XML è:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
                                        xmlns:exterms="http://www.example.org/terms/">
    <rdf:Description rdf:about="http://www.example.org/index.html">
        <exterms:creation-date>August 16, 1999</exterms:creation-date>
    </rdf:Description>
</rdf:RDF>
```

Nella prima linea il tag <?xml version=”1.0”?> dichiara che quello che segue è scritto in XML e indica la versione utilizzata.

La linea 2 inizia con l’elemento rdf:RDF. Questo elemento indica che il codice XML che segue (inizia qui e finisce con il tag </rdf:RDF>) rappresenta delle dichiarazioni RDF. Di seguito all’elemento rdf:RDF, sulla stessa linea, vengono elencate le dichiarazioni dei namespace XML che verranno utilizzati nel codice successivo e vengono introdotte dall’attributo xmlns. Ad esempio, il frammento di codice xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns” specifica che tutti i tag nel codice che contengono un prefisso di rdf: sono parte del namespace identificato dall’URI http://www.w3.org/1999/02/22-rdf-syntax-ns#. Normalmente questo namespace viene utilizzato come riferimento per i termini del vocabolario RDF.

La linea 3 specifica un’altra dichiarazione di un namespace XML. Questa volta al prefisso exterms: viene associato l’URI http://www.example.org/terms/. In questo esempio, questo prefisso viene utilizzato per i termini definiti nel vocabolario dell’organizzazione example.org. Il simbolo “>” alla fine della linea 3 indica la fine dell’apertura del tag rdf:RDF e la fine di dichiarazioni di namespace.

Le linee da 4 a 6 forniscono la descrizione del grafo in RDF/XML. IL tag rdf:Description nella linea 4 indica l’inizio della descrizione di una risorsa e l’attributo rdf:about identifica il soggetto della dichiarazione rappresentato dal suo riferimento URI. La linea 5 fornisce la proprietà dell’elemento, introdotta da un tag chiamato con il QName exterms:creation-date che rappresenta il riferimento al predicato della dichiarazione. Il contenuto della proprietà, che è l’oggetto della dichiarazione, viene inserito tra i tag di apertura e chiusura del predicato. (Nell’esempio August 19, 1999). I tag che rappresentano la proprietà sono racchiusi all’interno del tag rdf:description che è stato aperto nella linea 4 e che viene chiuso nella linea 6.

In conclusione, la linea 7 indica la chiusura dell’elemento rdf:RDF aperto nelle riga 2.

Consideriamo un esempio più complesso con più di una dichiarazione per ogni soggetto e con un nodo vuoto. Il grafo è riportato in figura.

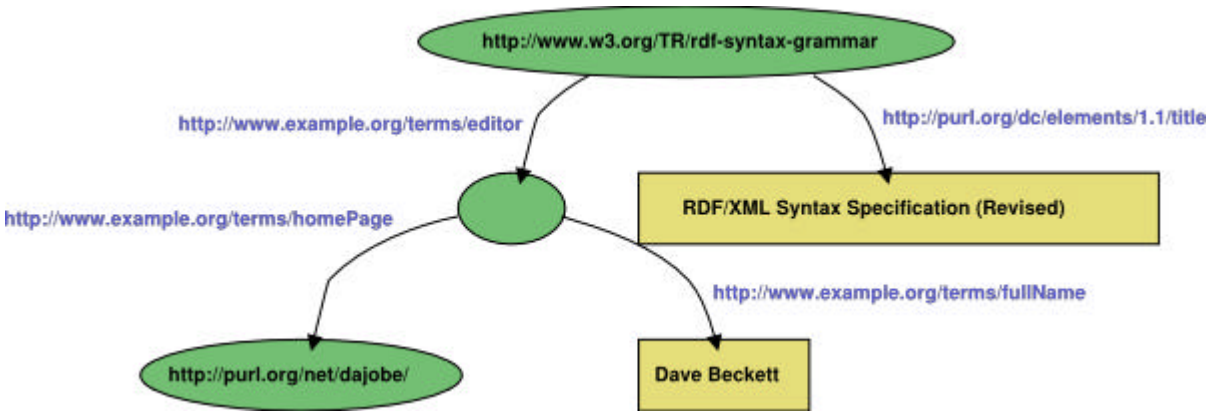

Figura 16 - RDF/XML - 2 Esempio

Il precedente grafo RDF può essere rappresentato dal seguente codice RDF/XML:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ex="http://example.org/stuff/1.0/">
```

```
        <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
                <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
                <ex:editor rdf:nodeID="abc"/>
        </rdf:Description>
        <rdf:Description rdf:nodeID="abc">
                        <ex:fullName>"Dave Beckett"</ex:fullName>
                        <ex:homePage rdf:resource="http://purl.org/net/dajobe/" />
        </rdf:Description>
</rdf:RDF>
```

Nella linea 1 il tag <?xml version=”1.0”?> è la dichiarazione che quello che segue è scritto in XML e viene indicata la versione utilizzata.

La linea 2 inizia con l’elemento rdf:RDF. Questo elemento indica che il codice XML che segue (inizia qui e finisce con il tag </rdf:RDF>) rappresenta delle dichiarazioni RDF. Di seguito all’elemento rdf:RDF, vengono elencate le dichiarazioni dei namespace XML e vengono introdotte dall’attributo xmlns del tag rdf:RDF. Questa dichiarazione specifica che tutti i tag che contengono un prefisso di rdf: sono parte del namespace identificato dall’URI http://www.w3.org/1999/02/22-rdf-syntax-ns#. Segue la dichiarazione un altro namespace XML. Questa volta al prefisso dc: viene associato l’URI http://purl.org/dc/elements/1.1/. Sulla riga successiva vi è una terza dichiarazione di un namespace. Al prefisso ex: viene associato l’URI http://example.org/stuff/1.0/. Il simbolo “>” alla fine della linea 4 indica la fine dell’apertura del tag rdf:RDF e la fine delle dichiarazioni dei namespace.

Le linee da 5 a 12 forniscono la descrizione del grafo in RDF/XML.

IL tag rdf:Description nella linea 5 indica l’inizio della descrizione della risorsa e l’attributo rdf:about identifica il soggetto della dichiarazione rappresentato dal suo riferimento URI. All’interno del tag vengono definite le sue due proprietà title e editor. Per la prima proprietà non ci sono problemi, viene descritta nella linea 6 tra i tag <dc:title> e </dc:title>, che identificano il predicato della dichiarazione, viene espresso il valore della proprietà “RDF/XML Syntax Specification (Revised)”. Nella riga 7, con il tag <ex:editor> viene enunciata la seconda proprietà. Al contrario degli esempi precedenti, questa volta la proprietà è a sua volta un grafo, ed inoltre il nodo collegato è un nodo vuoto. Il fatto che sia un nodo vuoto giustifica il fatto che nel teg di apertura di ex:editor sia riportato come attributo rdf:nodeID=”abc” dove abc è un nome fittizio dato al nodo vuoto per poterlo riferire successivamente. Il tag sulla linea 8 chiude la descrizione delle proprietà della pagina “http://www.w3.org/TR/rdf-syntax-grammar”.

Sulla linea 9 si apre un altro tag description che descrive le proprietà del nodo vuoto (da notare l’uso dell’identificatore del nodo abc dichiarato in precedenza). Sulla linea 10 e 11 vengono dichiarate le sue due proprietà.

Nelle righe successive vengono chiusi tutti i tag che erano rimasti aperti.

TIPIZZAZIONE

Come visto in precedenza, per ogni valore di proprietà può essere specificato il tipo. Consideriamo un esempio con un nodo che descrive una pagina web. La sua unica proprietà è la data di creazione. Un esempio di descrizione RDF/XML potrebbe essere la seguente:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                                xmlns:exterms="http://www.example.org/terms/">
        <rdf:Description rdf:about="http://www.example.org/index.html">
                <exterms:creation-date
```

```
                    rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1999-08-16
            </exterms:creation-date>
        </rdf:Description>
</rdf:RDF>
```

Nella linea 6, all'interno del tag di apertura exterms:creation-date, è stato aggiunto l'attributo rdf:datatype che indica il tipo di dato del valore contenuto all'interno del tag. Il riferimento URI del tipo di dato, oltre essere espresso come nell'esempio in forma estesa, visto che riferisce a un tipo di dato definito in XML schema poteva essere abbreviato con xsd:date.

Per usare l'abbreviazione bisogna dichiarare la corrispondenza fra xsd e http://www.w3.org/2001/XMLSchema#. Siccome la sintassi RDF/XML non permette di usare la sintassi abbreviata con namespace per i valori degli attributi di un tag (nel nostro caso è possibile usare l'abbreviazione con namespace per rdf:datatype ma non xsd:date) è necessario dichiarare esternamente l'abbreviazione. Per fare ciò si utilizza una dichiarazione del tipo:

```
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
```

che definisce che xsd rappresenta il namespace dei tipi di dato definiti in XML schema. Per utilizzare questa dichiarazione all'interno del documento è necessario riferirsi ad essa usando &xsd;.

Usando questa abbreviazione l'esempio precedente si trasformerebbe nel seguente modo:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                                xmlns:exterms="http://www.example.org/terms/">
        <rdf:Description rdf:about="http://www.example.org/index.html">
            <exterms:creation-date rdf:datatype="&xsd;date">1999-08-16</exterms:creation-date>
        </rdf:Description>
</rdf:RDF>
```

XML:LANG

RDF/XML permette tramite l'uso dell'attibuto xml:lang di identificare contenuti perticolari per una lingua. L'attributo xml:lang può essere usato in tutti i nodi o in tutte le proprietà il cui contenuto deve essere consultato in diverse lingue. Il valore xml:lang="" indica l'assenza di un identificatore di lingua. Un esempio può essere:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                                xmlns:dc="http://purl.org/dc/elements/1.1/">
        <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
            <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
            <dc:title xml:lang="en">RDF/XML Syntax Specification (Revised)</dc:title>
            <dc:title xml:lang="en-US">RDF/XML Syntax Specification (Revised)</dc:title>
        </rdf:Description>
        <rdf:Description rdf:about="http://example.org/buecher/baum" xml:lang="de">
            <dc:title>Der Baum</dc:title>
            <dc:description>Das Buch ist außergewöhnlich</dc:description>
            <dc:title xml:lang="en">The Tree</dc:title>
        </rdf:Description>
</rdf:RDF>
```

Nella prima risorsa descritta, l'attributo xml:lang viene usato per dare varie versioni della proprietà titolo. Nel primo caso è riportata la versione rivolta a tutte le lingue non specificate successivamente, il secondo caso riporta il titolo in versione inglese e nel terzo in versione inglese per stati uniti.

Nelle seconda risorsa descritta, l'attributo xml:lang viene utilizzato prima di tutto nel tag description per indicare la risorsa è stata definita in lingua tedesca poi in una seconda dichiarazione di titolo per indicare che esiste una versione dell'attributo titolo tradotta in inglese.

RDF:PARSETYPE=”RESOURCE”

Per dare una descrizione dei nodi vuoti senza bisogno del rdf:nodeID come visto precedentemente evitando così di spostare le proprietà del nodo in una nuova dichiarazione successiva, si può usare una dichiarazione <rdf:Description>, < / rdf:Description> specificando come attributo del tag di apertura rdf:parseType=”Resource”.

In questo modo, è possibile dare una descrizione di una risorsa senza specificare la risorsa stessa. Nell'esempio che segue, la proprietà ex:editor della risorsa descritta è un elemento non definito (nodo bianco) di cui sono definite delle proprietà.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ex="http://example.org/stuff/1.0/">
    <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
                                                dc:title="RDF/XML Syntax Specification (Revised)">
        <ex:editor rdf:parseType="Resource">
            <ex:fullName>Dave Beckett</ex:fullName>
            <ex:homePage rdf:resource="http://purl.org/net/dajobe/"/>
        </ex:editor>
    </rdf:Description>
</rdf:RDF>
```

OMETTERE NODI

Se tutte le proprietà di un nodo vuoto hanno lo stesso valore dell'attributo xml:lang, se è presente, e fra tutte le proprietà al massimo una specifica il tipo di dato contenuto (meglio nessuna per chiarezza), e nessuna proprietà ha attributi al suo interno, il nodo può essere abbreviato spostando la dichiarazione delle proprietà all'interno del tag di apertura.

Nell'esempio successivo è stata usata questa abbreviazione su un nodo vuoto che aveva come proprietà l'elemento ex:fullname. La traduzione in RDF / XML è la seguente:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ex="http://example.org/stuff/1.0/">
    <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
                                                dc:title="RDF/XML Syntax Specification (Revised)">
        <ex:editor ex:fullName="Dave Beckett" />
    </rdf:Description>
</rdf:RDF>
```

TIPI DI NODO

È comune nei grafi RDF avere un predicato del tipo rdf:type collegato a un nodo soggetto. Questo viene convenzionalmente chiamato tipo di nodo. RDF/XML permette di esprimere questa dichiarazione in maniera molto concisa. Basta sostituire a rdf:Description il riferimento URI contenuto nel nodo oggetto della dichiarazione del tipo.

Negli esempi successivi viene mostrato l'uso di rdf:type. Nel primo caso ne verrà fatto un uso esplicito, mentre nel secondo verrà usata un'abbreviazione.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ex="http://example.org/stuff/1.0/">
    <rdf:Description rdf:about="http://example.org/thing">
        <rdf:type rdf:resource="http://example.org/stuff/1.0/Document"/>
        <dc:title>A marvelous thing</dc:title>
    </rdf:Description>
</rdf:RDF>
```

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ex="http://example.org/stuff/1.0/">
    <ex:Document rdf:about="http://example.org/thing">
        <dc:title>A marvelous thing</dc:title>
    </ex:Document>
</rdf:RDF>
```

ABBREVIARE URI

RDF/XML permettere di abbreviare i riferimenti URI in due modi. XML fornisce l'attributo xml:base per specificare il riferimento base per risolvere riferimenti URI. Il riferimento base può essere utilizzato in tutti i valori degli attributi rdf:about, rdf:resource, rdf:ID e rdf:datatype nel codice RDF/XML .

L'attributo rdf:ID in un nodo può essere usato al posto di rdf:about e fornisce un riferimento URI equivalente concatenando la stringa "#" al valore dell'attributo. Per esempio, se rdf:ID="name", questo sarebbe equivalente a rdf:about="#name".

Rdf:ID fornisce un controllo addizionale che verifica che lo stesso valore dell'attributo appaia solo una volta all'interno del nemespace xml:base, quindi viene usato per definire un insieme di termini distinti all'interno dello stesso riferimento URI.

L'esempio successivo mostra l'abbreviazione di http://example.org/here/#snack usando xml:base="http://example.org/here/" e rdf:ID all'interno del tag rdf:description. L'uso di xml:base viene anche utilizzato all'interno del predicato ex:prop per associare al valore "fruit/apple" il riferimento URI http://example.org/here/fruit/apple.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                 xmlns:ex="http://example.org/stuff/1.0/" xml:base="http://example.org/here/">
    <rdf:Description rdf:ID="snack">
        <ex:prop rdf:resource="fruit/apple"/>
    </rdf:Description>
```

```
</rdf:RDF>
```

Per maggiori informazioni su RDF/XML vedi bibliografia [5].

3.4 ALTRE CAPACITA' RDF

RDF fornisce un certo numero di capacità addizionali, come dei tipi di dato e delle proprietà per rappresentare un gruppo di risorse o di dichiarazioni RDF, e capacità per la rappresentazione di frammenti XML come valori di proprietà. Queste capacità addizionali sono descritte nella seguente sezione.

CONTENITORI RDF

È spesso necessario descrivere un gruppo di cose: per esempio, per dire che un libro è stato creato da alcuni autori, o dexcrivere la lista degli studenti in corso, o elencare i moduli software di un pacchetto. RDF fornisce alcuni tipi e proprietà predefiniti che possono essere usati per descrivere questi gruppi.

Per primo, RDF fornisce un vocabolario di contenitori che contiene tre tipi predefiniti. Un contenitore è una risorsa che contiene cose. Le cose contenute sono chiamati membri del contenitore. Un membro di un contenitore è essere una risorsa, compreso un nodo vuoto. RDF definisce tre tipi di contenitori:

- rdf:Bag.
- rdf:Seq.
- rdf:Alt.

Un Bag rappresenta un gruppo si risorse, che può contenere membri duplicati, dove non ha significato l'ordine dei membri. Per esempio, un Bag può essere usato per descrivere un gruppo di parti dove l'ordine non ha importanza.

Una sequenza o Seq rappresenta un gruppo di risorse, che può contenere membri duplicati, dove l'ordine dei membri ha significato. Per esempio, una sequenza può essere usata per descrivere un gruppo da mantenere in ordine alfabetico.

Un Alt rappresenta un gruppo di risorse che sono alternative. Per esempio, un Alt può essere usata per descrivere delle lingue alternative in cui è disponibile la traduzione di un libro o per descrivere una lista di siti Internet dove è disponibile una risorsa.

Per descrivere una risorsa che è un contenitore, la risorsa avrà una proprietà rdf:type che avrà il valore di rdf:Bag o rdf:Seq o rdf:Alt. I membri del contenitori possono essere descritti da delle dichiarazioni aventi come soggetto il contenitore e come oggetto un membro. I predicati delle dichiarazioni avranno il nome di rdf:_n, dove n è un numero intero maggiore di zero (zero escluso).

Un uso di un contenitore è, per esempio, necessario per rappresentare la dichiarazione:

Il corso 6.001 è frequentato dagli studenti Amy, Mohamed, Johann, Maria e Phuong

Il corso può essere descritto da una proprietà studenti che è di tipo rdf:Bag. Il grafo per rappresentare tutto ciò è riportato in figura:

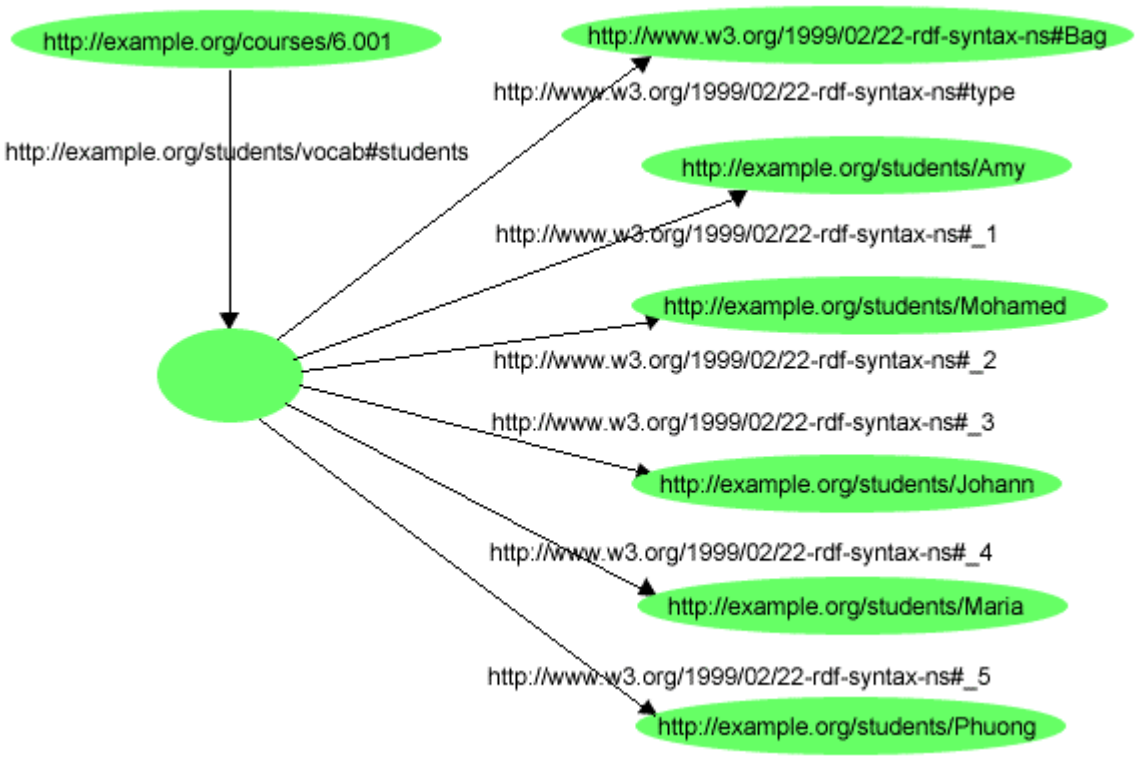

Figura 17 - RDF – Grafo con esempio di Bag

La relativa traduzione di RDF / XML è riportata di seguito:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                                    xmlns:s="http://example.org/students/vocab#">
    <rdf:Description rdf:about="http://example.org/courses/6.001">
        <s:students>
            <rdf:Bag>
                <rdf:li rdf:resource="http://example.org/students/Amy"/>
                <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
                <rdf:li rdf:resource="http://example.org/students/Johann"/>
                <rdf:li rdf:resource="http://example.org/students/Maria"/>
                <rdf:li rdf:resource="http://example.org/students/Phuong"/>
            </rdf:Bag>
        </s:students>
    </rdf:Description>
</rdf:RDF>
```

Nell'esempio riportato è stato usato rdf:li al posto di utilizzare un numero esplicito per ogni membro del contenitore. Da notare l'uso di <rdf:Bag> all'interno dell'elemento s:students come abbreviazione del nodo vuoto e della proprietà rdf:type.

Un esempio di rdf:Seq sarebbe analogo a quello di rdf:Bag, basta sostituire il tipo di nodo.

Per illustrare l'uso di un contenitore Alt, utilizziamo la dichiarazione:

Il codice sorgente di X11 può essere scaricato da ftp.exampe.org, ftp1.exampe.org o ftp2.exampe.org.

L'esempio può essere rappresentato dal seguente grafo:

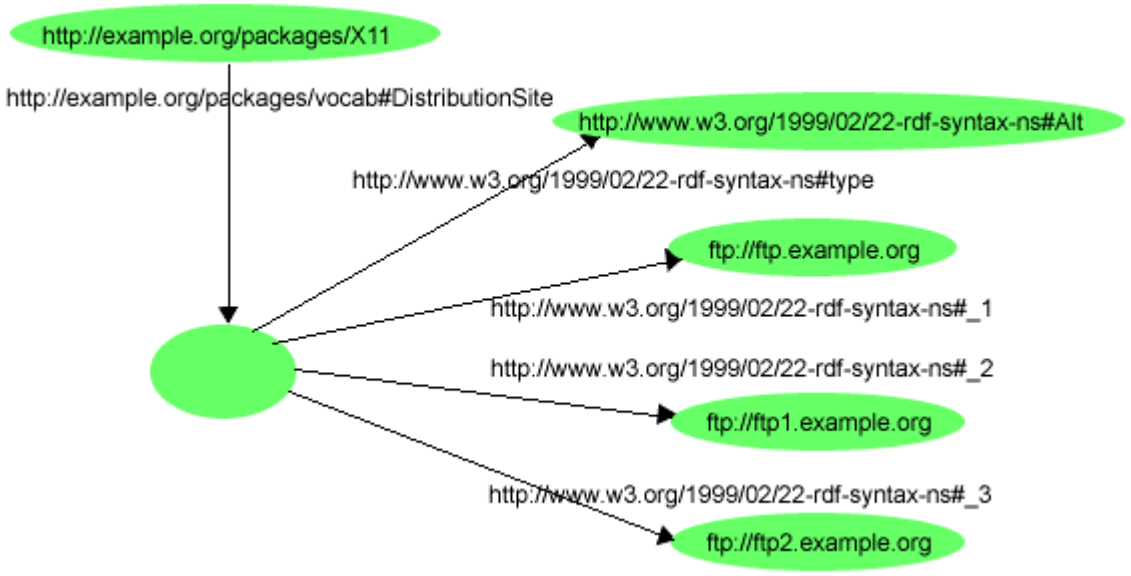

Figura 18 - RDF - Grafo di esempio con Alt

Il precedente grafo può essere tradotto in RDF / XML:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                                    xmlns:s="http://example.org/packages/vocab#">
    <rdf:Description rdf:about="http://example.org/packages/X11">
        <s:DistributionSite>
            <rdf:Alt>
                <rdf:li rdf:resource="ftp://ftp.example.org"/>
                <rdf:li rdf:resource="ftp://ftp1.example.org"/>
                <rdf:li rdf:resource="ftp://ftp2.example.org"/>
            </rdf:Alt>
        </s:DistributionSite>
    </rdf:Description>
</rdf:RDF>
```

Il contenitore Alt è stato creato per avere un membro, identificato dalla proprietà rdf:_1, che viene considerato come il valore di default. Per gli altri membri che non sono identificati da rdf:_1, l'ordine non è significativo.

COLLEZIONI RDF

Una limitazione dei contenitori descritti nella sezione precedente non c'è modo di specificare "questi sono tutti i membri del contenitore". Quindi, quando un grafo descrive alcuni membri, non si può escludere che non ci siano altri grafi che descrivono altri membri.

RDF fornisce un supporto per descrivere gruppi contenti solo i membri specificati. Si chiamano collezioni RDF. Una collezione RDF è un gruppo di cose rappresentante una lista strutturata. Questa lista strutturata è costituita da un tipo predefinito rdf:List e le proprietà rdf:first, rdf:rest e rdf:nil.

Per illustrare il suo uso utilizziamo il seguente esempio:

Gli studenti del corso 6.001 sono Amy, Mohamed e Johann

La precedente dichiarazione può essere rappresentata dal seguente grafo.

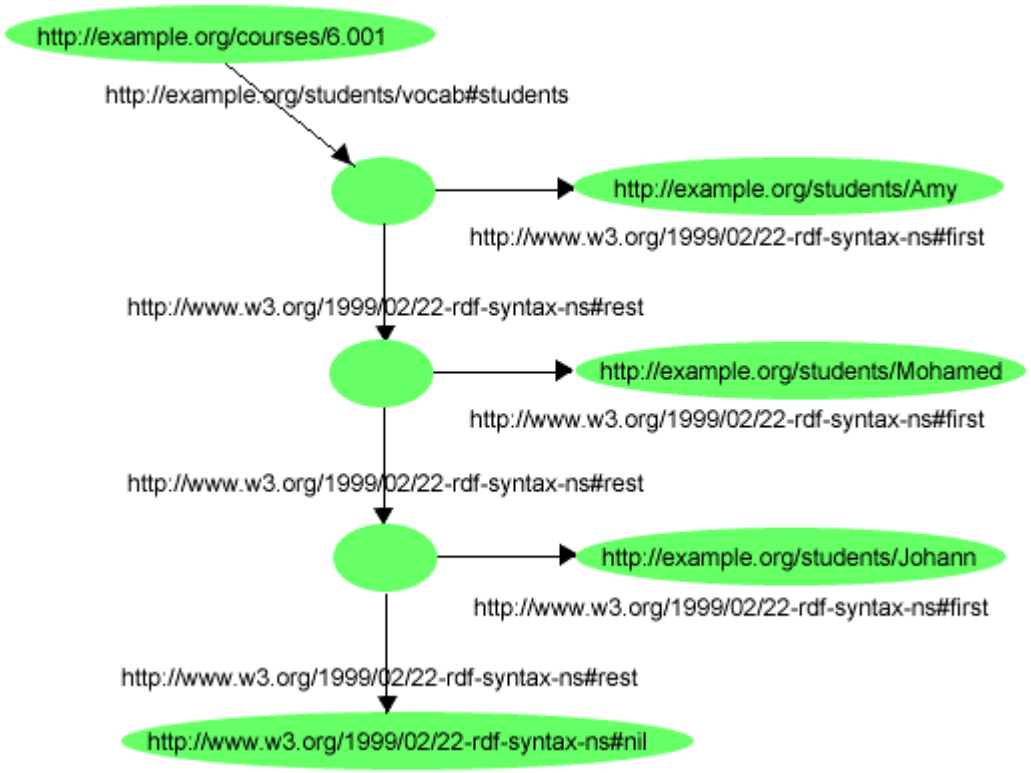

Figura 19 - RDF - Grafo di esempio Collection

In questo grafo, tutti i membri della collezione sono oggetti di una proprietà rdf:first dove il soggetto (nell'esempio un nodo vuoto) rappresenta la lista. Questa risorsa è collegata alle altre con la proprietà rdf:rest. La fine della lista è indicata dalla proprietà rdf:rest che ha come oggetto la risorsa rdf:nil.

RDF/XML fornisce una notazione speciale per descrivere semplicemente una collezione. In RDF/XML, una collezione può essere descritta da un elemento con l'attributo rdf:parseType="Collection" che contiene un gruppo di elementi che rappresentano i membri della collezione. Considerando l'esempio del grafo precedente, il corrispondente codice RDF/XML seguente:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                                    xmlns:s="http://example.org/students/vocab#">
    <rdf:Description rdf:about="http://example.org/courses/6.001">
        <s:students rdf:parseType="Collection">
            <rdf:Description rdf:about="http://example.org/students/Amy"/>
            <rdf:Description rdf:about="http://example.org/students/Mohamed"/>
            <rdf:Description rdf:about="http://example.org/students/Johann"/>
        </s:students>
    </rdf:Description>
</rdf:RDF>
```

L'uso di rdf:parseType="Collection" in RDF/XML definisce sempre una struttura come quella mostrata dal grafo in figura. Comunque, RDF non forza a utilizzare questo particolare modo per rappresentare la collezione ed è possibile usarne un altro modo per descriverla.

Il codice RDF/XML seguente rappresenta la modalità di rappresentare la collezione senza rdf:parseType=”Collection”.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                                    xmlns:s="http://example.org/students/vocab#">
        <rdf:Description rdf:about="http://example.org/courses/6.001">
                <s:students rdf:nodeID="sch1"/>
        </rdf:Description>
        <rdf:Description rdf:nodeID="sch1">
                <rdf:first rdf:resource="http://example.org/students/Amy"/>
                <rdf:rest rdf:nodeID="sch2"/>
        </rdf:Description>
        <rdf:Description rdf:nodeID="sch2">
                <rdf:first rdf:resource="http://example.org/students/Mohamed"/>
                <rdf:rest rdf:nodeID="sch3"/>
        </rdf:Description>
        <rdf:Description rdf:nodeID="sch3">
                <rdf:first rdf:resource="http://example.org/students/Johann"/>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
        </rdf:Description>
</rdf:RDF>
```

RDF REIFICATION

Le applicazioni RDF, alcune volte, necessitano di descrivere altre dichiarazioni RDF usando RDF. Per esempio consideriamo la seguente terna:

exproducts:item10245	exterms:weight	"2.4"^^xsd:decimal

RDF fornisce nel suo vocabolario le definizioni per descrivere questa dichiarazione RDF. Una descrizione di una dichiarazione consiste in un tipo rdf:Startement e in tre proprietà rdf:subject, rdf:predicate e rdf:object. Usando queste definizioni e assegnando alla definizione un nome come exproducts:triple12345 la dichiarazione può essere definita nel seguente modo:

exproducts:triple12345	rdf:type	rdf:Statement .
exproducts:triple12345	rdf:subject	exproducts:item10245 .
exproducts:triple12345	rdf:predicate	exterms:weight .
exproducts:triple12345	rdf:object	"2.4"^^xsd:decimal .

Questa dichiarazione identifica con il riferimento URI exproducts:triple12345 che si sta definendo una dichiarazione RDF, in cui il soggetto della dichiarazione è identificato da exproducts:item10245, il predicato della dichiarazione è identificato da exterms:weight, e l’oggetto della dichiarazione è 2.4 di tipo xsd:decimal.

Usando questa convenzione, si può memorizzare, ad esempio, il fatto che John Smith è il creatore della terna. In questo caso risulterebbe:

exproducts:triple12345	rdf:type	rdf:Statement .
exproducts:triple12345	rdf:subject	exproducts:item10245 .
exproducts:triple12345	rdf:predicate	exterms:weight .
exproducts:triple12345	rdf:object	"2.4"^^xsd:decimal .
exproducts:triple12345	dc:creator	exstaff:85740 .

IL grafo originale da cui si possono ricavare le terne riportate qui sopra è riportato nella figura seguente.

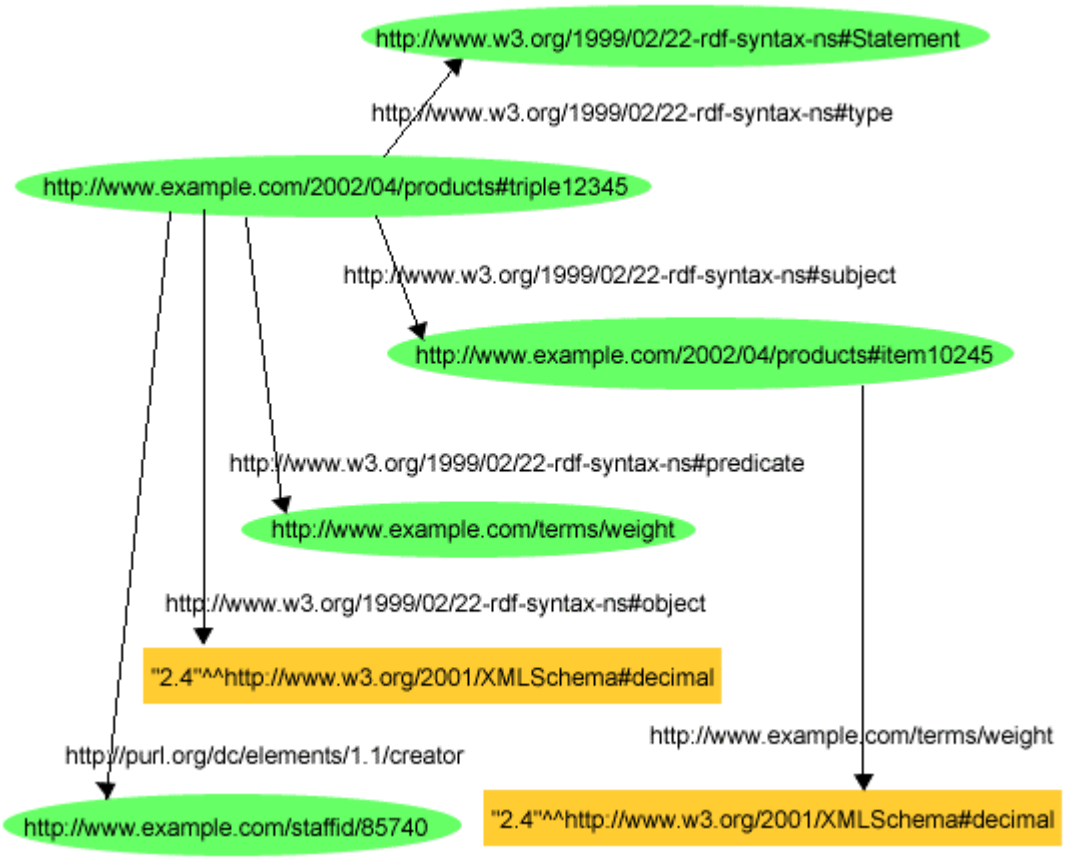

Figura 20 - RDF - Grafo esempio di Reification

Questo grafo può essere scritto in RDF/XML come riportato qui di seguito:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:exterms="http://www.example.com/terms/"
                                                  xml:base="http://www.example.com/2002/04/products">
        <rdf:Description rdf:ID="item10245">
                <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
        </rdf:Description>
        <rdf:Statement rdf:about="#triple12345">
                <rdf:subject rdf:resource="http://www.example.com/2002/04/products#item10245"/>
                <rdf:predicate rdf:resource="http://www.example.com/terms/weight"/>
                <rdf:object rdf:datatype="&xsd;decimal">2.4</rdf:object>
                <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
        </rdf:Statement>
</rdf:RDF>
```

XML NEI VALORI DI PROPRIETA'

Alcune volte, il valore di una proprietà necessita di contenere un frammento di XML o del testo che contiene XML markup. Per esempio, un editore deve registrare dei metadati che includono i titoli dei libri e degli articoli pubblicati. RDF/XML fornisce una notazione speciale per queste situazioni.

Queste situazioni richiedono l'uso dell'attributo rdf:parseType="Literal" che indica che il contenuto dell'elemento deve essere interpretato come un frammento XML. Per illustrare l'uso di rdf:parseType="Literal", guardiamo il codice RDF/XML riportato qui di seguito:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/" xml:base="http://www.example.com/books">
      <rdf:Description rdf:ID="book12345">
            <dc:title rdf:parseType="Literal">
            <span xml:lang="en"> The <em>&lt;br /&gt;</em> Element Considered Harmful.
                                                                              </span>
            </dc:title>
      </rdf:Description>
</rdf:RDF>
```

Il codice RDF/XML dell'esempio precedente descrive un grafo che contiene una terna con soggetto ex:book12345 e predicato dc:title. L'attributo rdf:parseType="Literal" indica che tutto il codice XML contenuto tra i teg <dc:title> e </dc:title> è un frammento di codice XML che è il valore della proprietà.

3.5 RDF SCHEMA

RDF fornisce un metodo per esprimere semplici dichiarazioni di risorse. La comunità degli utenti di RDF necessitava, però, di definire dei vocabolari che potessero essere usati nelle dichiarazioni. Per esempio, se la compagnia example.com volesse descrivere classi come exterms:Tent e usare delle proprietà come exterms:model, exterms:weightInKg e exterms:packedSize per descriverla, o allo stesso modo, persone interessate alla descrizione di informazioni bibliografiche vorrebbero descrivere delle classi come ex2:Book o ex2:MagazineArticle e usare proprietà come ex2:author, ex2:title e ex2:subject per descriverle, non potrebbero in quanto il vocabolario RDF non fornisce definizioni per ogni specifica applicazione. Classi e proprietà possono essere descritte in un vocabolario RDF usando una estensione fornita da RDF chiamata RDF schema.

RDF schema non fornisce un vocabolario per l'applicazione specifica ma fornisce gli strumenti necessari per descrivere classi e proprietà.

RDF, a differenza di molti sistemi che definiscono le classi in termini delle loro proprietà e delle loro istanze, ha un linguaggio di descrizione del vocabolario RDF che descrive le proprietà in termini della classe o della risorsa a cui appartengono. Questa è la regola del meccanismo del dominio e del range descritto in questa sezione. Per esempio, si definisce la proprietà eg:author che ha dominio eg:Document e range eg:Person, dove i classici linguaggi orientati agli oggetti tipicamente definiscono la classe eg:Book con l'attributo eg:author di tipo eg:Person. Un beneficio dell'approccio RDF è che permette a tutti di estendere la descrizione di risorse esistenti.

Il linguaggio presentato consiste in una collezione di risorse RDF che possono essere usate per descrivere proprietà di altre risorse applicate in un vocabolario RDF specifico.

Nei prossimi paragrafi, il vocabolario base sarà chiamato rdfs associato al riferimento URI http://www.w3.org/2000/01/rdf-schema# mentre il prefisso rdf sarà associato al riferimento URI http://www.w3.org/1999/02/22-rdf-syntax-ns#.

CLASSI

Le risorse possono essere divise in gruppi chiamati classi. I membri di una classe sono conosciuti come istanze della classe. Le classi sono delle risorse e sono spesso identificate da un riferimento URI RDF. Possono essere descritte usando le proprietà RDF. La proprietà rdf:type può essere usata per dichiarare di che tipo di risorsa è un'istanza di una classe.

RDF distingue tra la classe e l'insieme delle sue istanze. Associata a ogni classe c'è un insieme, chiamato estensione della classe, che è l'insieme delle istanze della classe. Due classi potrebbero avere lo stesso insieme di istanze ma essere differenti.

Un gruppo di risorse che sono classi di RDF Schema sono delle classi chiamate rdfs:Class.

Se una classe C è una sottoclasse della classe C', allora tutte le istanze di C saranno istanze anche di C'. La proprietà rdfs:subClassOf può essere usata per definire che una classe è una sottoclasse di un'altra classe. Il termine superclasse è usata col significato opposto a sottoclasse. Ad esempio, se la classe C' è la superclasse di C, allora tutte le istanze di C sono anche istanze di C'.

Classe	Descrizione
rdfs:Resource	Tutto ciò che viene descritto tramite RDF viene chiamato risorsa e è una istanza della classe rdfs:Resource. Questa classe contiene qualunque cosa. Tutte le altre classi saranno sottoclassi di questa classe. Rdfs:Resource è una istanza della classe rdfs:Class.
rdfs:Class	Questa è la classe che descrive tutte le risorse che sono altre classi RDF.
rdfs:Literal	La classe rdfs:Literal è la classe dei valori letterali come stringhe e valori interer. I membri di questa classe possono essere semplici o tipizzati. Un membro tipizzato è una istanza della classe datatype.
rdfs:Datatype	rdfs:Datatype è la classe dei tipi di dato. Tutte le istanze di rdfs:Datatype corrispondono ai tipi di dato definiti in precedenza. rdfs:Datatype è sia un'istanza sia una sottoclasse di rdf:Class. Tutte le istanze di rdfs:Datatype sono sottoclassi di rdfs:Literal.
rdf:XMLLiteral	La classe rdf:XMLLiteral è la classe che descrive i frammenti di codice XML. rdf:XMLLiteral è una istanza della classe rdfs:Datatype e è una sottoclasse della classe rdfs:Literal.
rdf:Property	Rdf:Property è la classe che descrive le proprietà RDF. Rdf:Property è una istanza della classe rdfs:Class.

PROPRIETA'

I concetti base di RDF descrivono il concetto di proprietà RDF come la relazione tra la risorsa soggetto e la risorsa oggetto.

La proprietà rdfs:subPropertyOf viene usata per dichiarare che una proprietà è una sottoproprietà di un'altra. Se la proprietà P è una sottoproprietà di P', allora tutte le coppie di risorse che sono associate da P sono associate anche da P'. Il termine super-propietà viene spesso usata con il significato inverso di sottoproprietà. Se la proprietà P' è la super-proprietà di P, allora tutte le coppie di risorse associate da P sono associate anche da P'.

Proprietà	Descrizione
rdfs:range	rdfs:range è una istanza di rdf:Property che viene usata per dichiarare che il valore di una proprietà è una istanza di una o più classi. La terna P rdfs:range C dichiara che P è una istanza della classe rdf:Property, che C è una istanza della classe rdf:Class e che le risorse denotate dalle terne che hanno come oggetto P sono delle istanze di C. Quando P ha più di una proprietà rdfs:range, allora le risorse denotate dagli oggetti delle terne sono istanze di tutte classi dichiarate dalla proprietà

	rdfs:range. La proprietà rdf:range può essere applicata a se stessa. Il rdfs:range di rdfs:range è la classe rdfs:Class. Questo significa che tutte le risorse che sono valori della proprietà rdfs:range sono istanze della classe rdfs:Class. La proprietà rdfs:range si può applicare alle proprietà. Questo può essere rappresentato in RDF usando la proprietà rdfs:domain. Il rdfs:domain di rdfs:range è la classe rdf:Property. Questo significa che tuute le risorse che sono valori della proprietà rdfs:range sono istanze della classe rdf:Property.
rdfs:domain	rdfs:domain è una istanza della classe rdf:Property che viene usata per dichiarare che tutte le risorse che hanno la proprietà sono istanze di una o più classi. La terna P rdfs:domain C dichiara che P è una istanza della classe rdf:Property, che C è una istanza della classe rdfs:Class e che la risorsa denotata dal soggetto della terna che ha per predicato P è una istanza della classe C. Quando la proprietà P ha più di una proprietà rdfs:domain, allora la risorsa denotata dal soggetto delle terne che hanno predicato P sono istanze di tutte le classi dichiarate dalla proprietà rdfs:domain. La proprietà rdfs:domain può essere applicata a se stessa. Il rdfs:domain di rdfs:domain è la classe rdf:Property. Questo significa che tutte le risorse con una proprietà rdfs:domain sono istanze di rdf:Property. Il rdfs:range di rdfs:domain è la classe rdfs:Class. Questo significa che tutte le risorse che sono valore della proprietà rdfs:domain è una istanza di rdfs:Class.
rdf:type	rdf:type è una istanza di rdf:Property che viene usata per dichiarare che una risorsa è una istanza di una classe. La terna R rdf:type C dichiara che C è una istanza di rdfs:Class e R è una istanza di C. Il rdfs:domain di rdf:type è la classe rdfs:Resource. Il rdfs:range di rdf:type è la classe rdfs:Class.
rdfs:subClassOf	La proprietà rdfs:subClassOf è una istanza di rdf:Property che viene usata per dichiarare che tutte le istanze di una classe sono istanze di un'altra. La terna C1 rdfs:subClassOf C2 dichiara che C1 è una istanza di rdfs:Class, C2 è una istanza di rdfs:Class e che C1 è una sottoclasse di C2. La proprietà rdfs:subClassOf è transitiva. Il rdfs:domain di rdfs:subClassOf è la classe rdfs:Class. Il rdfs:range di rdfs:subClassOf è la classe rdfs:Class.
rdfs:subPropertyOf	La proprietà rdf:subPropertyOf è una istanza della classe rdf:Property che viene usata per dichiarare che tutte le risorse associate da quella proprietà sono associate anche da un'altra. La terna P1 rdfs:subPropertyOf P2 dichiara che P1 è una istanza della classe rdf:Property, P2 è una istanza della classe rdf:Property e P1 è una sottoproprietà di P2. La proprietà rdfs:subPropertyOf è transitiva. Il rdfs:domain di rdfs:subPropertyOf è la classe rdf:Property. Il rdfs:range di rdfs:subPropertyOf è la classe rdf:Property.
rdfs:label	Rdfs:label è una istanza della classe rdf:Property che può essere usata per fornire una versione leggibile dagli umani del nome della risorsa. La terna R rdfs:label L dichiara che L è la versione leggibile dagli umano del nome della risorsa R. Il rdfs:domain di rdfs:label è la classe rdfs:Resource. Il rdfs:range di

	rdfs:label è la classe rdfs:Literal.
rdfs:comment	Rdfs:comment è una istanza della classe rdf:Property che può essere usata per fornire una descrizione leggibile da un utente umano di una risorsa. La terna R rdfs:comment L dichiara che L è la descrizione leggibile da un utente umano di R. Il rdfs:domain di rdfs:comment è la classe rdfs:Resource. Il rdfs:range di rdfs:comment è la classe rdfs:Literal. Un commento testuale aiuta a chiarire il significato di una classe e una proprietà RDF.

ALTRE CLASSI E PROPRIETA'

I contenitori RDF sono definiti dalle seguenti classi e proprietà:

Classe/Proprietà	Descrizione
rdfs:Container	La classe rdfs:Container è la superclasse di tutti i contenitori RDF.
rdf:Bag	La classe rdf:Bag è la classe per i contenitori di tipo Bag. È una sottoclasse della classe rdfs:Container. Nonostante non sia formalmente differente dalle classi rdf:Seq o rdf:Alt ma viene convenzionalmente usata per indicare che il contenitore è inteso come non ordinato.
rdf:Seq	La classe rdf:Seq è la classe per i contenitori di tipo Sequence. È una sottoclasse della classe rdfs:Container. Nonostante non sia formalmente differente dalle classi rdf:Bag o rdf:Alt ma viene convenzionalmente usata per indicare che il contenitore è inteso come ordinato.
rdf:Alt	La classe rdf:Alt è la classe per i contenitori di tipo Alternative. È una sottoclasse della classe rdfs:Container. Nonostante non sia formalmente differente dalle classi rdf:Seq o rdf:Bag ma viene convenzionalmente usata per indicare che sarà possibile selezionare solo uno dei membri del contenitore. Il primo membro (quello identificato da rdf:_1) è la scelta predefinita.
rdfs:Container MembershipProperty	La classe rdfs:ContainerMembershipProperty ha come istanze rdf:_1, rdf:_2, rdf:_3, … che sono usati per dichiarare che una risorsa è un membro di un contenitore. Rdfs:ContainerMembershipProperty è una sottoclasse della classe rdf:Property. Ogni istanza di rdfs:ContainerMembershipProperty è una sotto-proprietà della proprietà rdfs:member. Avendo un contenitore C, la terna C rdf:_nnn O dove nnn è un numero intero maggiore di 0 dichiara che O è un membro del contenitore C.
rdfs:member	Rdfs:member è una istanza di rdf:Property che è una super-proprietà di tutte le proprietà "è membro di" di tutti i contenitori.

Le collezione RDF sono definite dalle seguenti classi e proprietà:

Classe/Proprietà	Descrizione
rdf:List	rdf:List è una istanza della classe rdfs:Class che viene usata per creare la descrizione di una lista o di strutture simili.
rdf:first	Rdf:first è una istanza di rdf:Property che può essere usata per creare la descrizione di una lista e altre strutture simili. La terna L rdf:first O dichiara che c'è un primo elemento nella relazione tra L e O. Il rdfs:domain di rdf:first è la classe rdf:List. Il rdfs:range di rdf:first è la classe rdfs:Resource.
rdf:rest	rdf:rest è una istanza di rdf:Property che viene usata per creare la descrizione di una lista e altre strutture simili.

	La terna L rdf:rest O dichiara che c'è una relazione di "resto della lista" tra L e O. Il rdfs:domain di rdf:rest è la classe rdf:List. Il rdfs:range di rdf:rest è la classe rdf:List.
rdf:nil	La risorsa rdf:nil è una istanza di rdf:List che viene usata per rappresentare una lista vuota o una struttura simile. Una terna L rdf:rest rdf:nil dichiara che L è una istanza di rdf:List che ha uno e un solo elemento e che l'elemento può essere indicato usando la proprietà rdf:first.

Applicazioni RDF, alcune volte, necessitano di descrivere altre dichiarazioni RDF usando RDF. Questa descrizione può essere fatta con le seguenti classi e proprietà:

Classe/Proprietà	Descrizione
rdf:Statement	Rdf:Statement è una istanza della classe rdfs:Class. È intesa per rappresentare la classe di una dichiarazione RDF.
rdf:subject	Rdf:subject è una istanza della classe rdf:Property che viene usata per identificare il soggetto di una dichiarazione.
rdf:object	Rdf:object è una istanza della classe rdf:Property che viene usata per identificare l'oggetto di una dichiarazione.
rdf:predicate	Rdf:predicate è una istanza della classe rdf:Property che viene usata per identificare il predicato della dichiarazione.

Le seguenti classi e proprietà sono classi e proprietà di utilità.

Classe/Proprietà	Descrizione
rdfs:seeAlso	Rdf:seeAlso è una istanza della classe rdfs:Property che viene usata per indicare che una risorsa può fornire informazioni addizionali sulla risorsa soggetto. La terna S rdfs:seeAlso O dichiara che la risorsa O può fornire informazioni addizionali su S. Il rdfs:domain di rdfs:seeAlso è la classe rdfs:Resource. Il rdfs:range di rdfs:seeAlso è la classe di rdfs:Resource.
rdfs:isDefinedBy	rdfs:isDefinedBy è una istanza della classe rdf:Property che viene usata per indicare la risorsa che ha definito la risorsa soggetto. La terna S rdfs:isDefinedBy O dichiara che la risorsa O ha definito S. Il rdfs:domain di rdfs:isDefinedBy è la classe rdfs:Resource. IL rdfs:range di rdfs:isDefinedBy è la classe rdfs:Resource.

DESCRIVERE CLASSI

Il primo passo in ogni processo di descrizione è identificare i tipi di cose che devono essere descritte. Una classe in RDF Schema corrisponde al generico concetto di tipo o categoria. Le classi sono descritte usando le risorse di RDF Schema rdfs:Class e rdfs:Resource e le proprietà rdf:type e rdfs:subClassOf.

Per esempio, supponiamo che l'organizzazione example.org cerchi di usare RDF per fornire informazioni su differenti tipi di autoveicoli. In RDF Schema, example.org ritiene necessario per prima cosa rappresentare la categoria di cose "autoveicoli". Le risorse che appartengono alla classe sono chiamate istanze. In questo caso, example.org intende per istanze della classe risorse che sono autoveicoli.

In RDF Schema, una classe è una risorsa che ha la proprietà rdf:type con valore rdfs:Class. Quindi la classe autoveicolo può essere descritta assegnando alla classe il riferimento URI ex:MotorVehicle (usando il prefisso ex per rappresentare il riferimento URI http://www.example.org/schemas/vehicles che viene utilizzato come prefisso dei riferimenti URI che riferiscono all'interno del

vocabolario dell'organizzazione example.org) e descrivendola con la proprietà rdf:type che avrà valore rdfs:Class. Utilizzando il metodo a terne può essere così rappresentato:

ex:MotorVehicle	rdf:type	rdfs:Class

Dove la proprietà rdf:type è usata per indicare che la risorsa è una istanza di una classe. Quindi, avendo descritto la classe ex:MotorVehicle, la risorsa ex:companyCar che descrive un particolare autoveicolo può essere descritta dalla seguente dichiarazione.

exthings:companyCar	rdf:type	ex:MotorVehicle

Questa dichiarazione usa la convenzione commune che i nomi delle classi sono scritti con la prima lettera grande mentre i nomi di proprietà e di istanze sono scritti con la prima lettera piccola. Comunque, RDF Schema non richiede questa convenzione.

Dopo aver descritto la classe ex:MotorVehicle, l'organizzazione example.org può volere descrivere altre classi che rappresentano i vari tipi di specializzazioni degli autoveicoli come, per passeggeri, van, mimivan e altri. Queste classi possono essere descritte nello stesso modo della classe ex:MotorVehicle, assegnandogli un nuovo riferimento URI per ogni nuova classe e scrivendo la dichiarazione RDF che descrive quelle risorse. Ad esempio:

ex:Van	rdf:type	rdfs:Class
ex:Truck	rdf:type	rdfs:Class

Comunque, questa dichiarazione descrive solo le single classi. L'organizzazione example.org vuole indicare anche la speciale relazione con la classe ex:MotorVehicle.

Questo tipo di specializzazione della relazione tra due classi viene descritta utilizzando la proprietà predefinita rdfs:subClassOf per associare due classi. Per esempio, per dichiarare che ex:Van è un tipo specializzato di ex:MotorVehicle, l'organizzazione example.org potrebbe scrivere la seguente dichiarazione RDF:

ex:Van	rdfs:subClassOf	ex:MotorVehicle

Il significato della relazione rdfs:subClassOf è che tutte le istanze di ex:Van sono istanze anche di ex:MotorVehicle. Quindi, se la risorsa ex:companyVan è una istanza di ex:Van allora, in base alla dichiarazione della relazione rdfs:subClassOf, un software RDF interpreterà che l'informazione ex:companyVan sia anche una istanza di ex:MotorVehicle.

La proprietà rdfs:subClassOf è transitiva. Questo significa che, per esempio, in queste dichiarazioni RDF:

ex:Van	rdfs:subClassOf	ex:MotorVehicle
ex:MiniVan	rdfs:subClassOf	ex:Van

RDF Schema definisce ex:MiniVan è anche una sottoclasse di ex:MotorVehicle. La figura successiva mostra la gerarchia discorsa nell'esempio.

Figura 21 - RDF Schema - Esempio di gerarchia

Per semplificare, nel grafo tutte le proprietà rdf:type che associavano ogni classe alla classe rdfs:Class sono state omesse.

Lo schema precedente può essere descritto dalle seguenti terne:

ex:MotorVehicle	rdf:type	rdfs:Class .
ex:PassengerVehicle	rdf:type	rdfs:Class .
ex:Van	rdf:type	rdfs:Class .
ex:Truck	rdf:type	rdfs:Class .
ex:MiniVan	rdf:type	rdfs:Class .
ex:PassengerVehicle	rdfs:subClassOf	ex:MotorVehicle .
ex:Van	rdfs:subClassOf	ex:MotorVehicle .
ex:Truck	rdfs:subClassOf	ex:MotorVehicle .
ex:MiniVan	rdfs:subClassOf	ex:Van .
ex:MiniVan	rdfs:subClassOf	ex:PassengerVehicle

Questo esempio può essere tradotto in codice RDF/XML come segue :

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                                            xml:base="http://example.org/schemas/vehicles">
    <rdf:Description rdf:ID="MotorVehicle">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    </rdf:Description>
    <rdf:Description rdf:ID="PassengerVehicle">
```

```
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
    </rdf:Description>
    <rdf:Description rdf:ID="Truck">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
    </rdf:Description>
    <rdf:Description rdf:ID="Van">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
    </rdf:Description>
    <rdf:Description rdf:ID="MiniVan">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#Van"/>
        <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
    </rdf:Description>
</rdf:RDF>
```

Nel codice RDF/XML sono stari introdotto dei nomi, come MotorVehicle, per le risorse (Classi) che vengono descritte dove il valore della proprietà rdf:ID fornisce il nome della classe da utilizzare in altre descrizioni. Per esempio, la classe con rdf:ID=”MotorVehicle”, supponendo che il riferimento URI dello schema sia http://example.org/schemas/vehicles, può essere identificata dal riferiemento URI http://example.org/schemas/vehicles#MotorVehicle.

Per riferire a una classe durante una dichiarazione di una istanza, ad esempio per descrivere un singolo autoveicolo, si procede come nel seguente codice:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:ex="http://example.org/schemas/vehicles#" xml:base="http://example.org/things">
    <ex:MotorVehicle rdf:ID="companyCar"/>
</rdf:RDF>
```

DESCRIVERE PROPRIETA’

Oltre descrivere le specifiche classi di cose, la comunità di untenti necessitava anche di descrivere le specifiche proprietà che caratterizzano chiascuna classe. In RDF Schema, le proprietà vengono descritte usando la classe RDF rdf:Property e usando le proprietà rdfs:domain, rdfs:range e rdfs:subPropertyOf.

Tutte le proprietà possono essere descritte come istanze della classe rdf:Property. Quindi una nuova proprietà, come ad esempio exterms:weightInKg, viene descritta assegnando alla proprietà un riferimento URI e descrivendo questa risorsa con la proprietà rdf:type il cui valore sarà la classe rdf:Property.

```
exterms:weightInKg        rdf:type        rdf:Property .
```

RDF Schema fornisce anche un vocabolario per descrivere come proprietà e classi devono essere usati nel descrivere informazioni in RDF. Le informazioni più importanti sono dati dalle proprietà rdfs:range e rdfs:domain.

La proprietà rdfs:range viene usata per indicare che il valore di una particolare proprietà è progettato per essere di una specifica classe. Per esempio, l’organizzazione example.org vuole indicare

che la proprietà ex:author ha un valore che deve essere una istanza della classe ex:Person. Allora può scrivere le seguenti dichiarazioni RDF:

```
ex:Person      rdf:type      rdfs:Class .
ex:author      rdf:type      rdf:Property .
ex:author      rdfs:range    ex:Person .
```

Questa dichiarazione indica che ex:Person è una classe, ex:author è una proprietà e che una dichiarazione RDF che intenda usare la proprietà ex:author deve avere come oggetto una istanza della classe ex:Person.

Una proprietà, come ad esempio ex:hasMother, può avere nessuna, una o più di una proprietà rdf:range. Se ex:hasMother non ha nessuna rdf:range allora non si sa niente sul valore della proprietà. Se ex:hasMother ha una rdf:range, allora il valore di quella proprietà dovrà essere una istanza della classe specificata. Se ex:hasMother ha più rdf:range, allora il valore di quella proprietà dovrà essere una istanza che appartiene a tutte le classi indicate da rdf:range.

Ad esempio:

```
ex:hasMother    rdfs:range    ex:Female .
ex:hasMother    rdfs:range    ex:Person .
```

queste dichiarazioni RDF indicano che il valore della proprietà ex:Mother dovrà essere una istanza sia della classe ex:Female che una istanza della classe ex:Person.

La proprietà rdf:range può essere anche usata per indicare che il valore della proprietà deve essere di un determinato tipo di dato. Ad esempio:

```
ex:age      rdf:type      rdf:Property .
ex:age      rdfs:range    xsd:integer .
```

La proprietà rdf:domain è usata per indicare che una particolare proprietà è progettata per essere applicata a specifiche classi. Per esempio, se l'organizzazione exampe.org vuole indicare che la proprietà ex:author deve essere applicata alle istanze della classe ex:Book, può utilizzare le seguenti dichiarazioni RDF:

```
ex:Book      rdf:type      rdfs:Class .
ex:author    rdf:type      rdf:Property .
ex:author    rdfs:domain   ex:Book .
```

Questa dichiarazione indica che ex:Book è una classe, che ex:author è una proprietà e che una dichiarazione che usa la proprietà ex:author deve avere come soggetto una istanza della classe es:Book.

Un certa proprietà, come ad esempio esterms:weight, può avere nessuna, una o più di una proprietà rdfs:domain. Se exterms:weight non ha nessuna rdfs:domain allora non si sa niente sulla risorsa che utilizzerà la proprietà exterms:weight. Se exterms:weight ha una rdfs:domain, allora specifica che la proprietà exterms:weight può essere applicata solo alle istanze di una specifica classe. Se exterms:weight ha più di una rdfs:domain, allora la proprietà exterms:weight sarà applicata solo a quelle istanze che appartengono a tutte le classi specificate.

L'uso della proprietà range e domain può essere illustrato estendendo l'esempio del veicolo aggiungendo due proprietà ex:registeredTo e ex:rearSeatLegRoom, la nuova classe ex:Person e la descrizione esplicita del tipo di dato xsd:integer. La proprietà ex:registeredTo viene applicata a tutte le istanze della classe ex:MotorVehicle e il suo valore deve essere una istanza della classe ex:Person. Per questo esempio, la proprietà ex:rearSeatLegRoom viene applicata alle sole istanze della classe ex:PassengerVehicle e il suo valore è di tipo xsd:integer. Questo esempio può essere descritto dal seguente codice:

```
<rdf:Property rdf:ID="registeredTo">
    <rdfs:domain rdf:resource="#MotorVehicle"/>
```

```
        <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="rearSeatLegRoom">
        <rdfs:domain rdf:resource="#PassengerVehicle"/>
        <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>

<rdfs:Class rdf:ID="Person"/>

<rdfs:Datatype rdf:about="&xsd;integer"/>
```

Da notare che l'elemento <rdf:RDF> non è stato usato. Questo perché il codice precedente è stato scritto per essere inserito nell'esempio precedente con la descrizione della classe autoveicolo.

RDF Schema fornisce un metodo per specializzare delle proprietà per una classe. Questa relazione di specializzazione tra due classi viene descritta usando la proprietà predefinita rdfs:subPropertyOf. Per esempio, se ex:primaryDriver e ex:driver sono entrambe proprietà, l'organizzazione example.org può descrivere queste proprietà, e il fatto che ex:primaryDriver è la specializzazione di ex:driver, con le seguenti terne RDF:

```
ex:driver          rdf:type          rdf:Property .
ex:primaryDriver   rdf:type          rdf:Property .
ex:primaryDriver   rdfs:subPropertyOf ex:driver .
```

Il significato della relazione rdfs:subPropertyOf è che se una istanza dei exstaff:fred è un ex:primaryDriver di una istanza della classe ex:companyVan, allora RDF Schema definisce che exstaff:fred è anche un ex:driver dell'istanza della classe ex:companyVan. Il codice RDF/XML da inserire nell'esempio che descrive la classe autoveicoli che descrive queste proprietà è mostrato di seguito:

```
<rdf:Property rdf:ID="driver">
        <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Property>

<rdf:Property rdf:ID="primaryDriver">
        <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>
```

Una proprietà può essere una sottoproprietà di zero una o più di altre proprietà. Tutte le proprietà rdfs:range e rdfs:domain che sono applicate a una proprietà, viene applicata anche a tutte le sue sottoproprietà. Quindi, nell'esempio precedente, RDF Schema definisce la proprietà ex:primaryDriver che ha un rdfs:domain di valore ex:MotorVehicle perché è una sottoproprietà della proprietà ex:driver.

Il codice RDF/XML seguente descrive lo schema completo dell'esempio degli autoveicoli:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                                        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
                                            xml:base="http://example.org/schemas/vehicles">

        <rdfs:Class rdf:ID="MotorVehicle"/>
```

```
        <rdfs:Class rdf:ID="PassengerVehicle">
                <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
        </rdfs:Class>
        <rdfs:Class rdf:ID="Truck">
                <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
        </rdfs:Class>
        <rdfs:Class rdf:ID="Van">
                <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
        </rdfs:Class>
        <rdfs:Class rdf:ID="MiniVan">
                <rdfs:subClassOf rdf:resource="#Van"/>
                <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
        </rdfs:Class>
        <rdfs:Class rdf:ID="Person"/>
        <rdf:Property rdf:ID="registeredTo">
                <rdfs:domain rdf:resource="#MotorVehicle"/>
                <rdfs:range rdf:resource="#Person"/>
        </rdf:Property>
        <rdf:Property rdf:ID="rearSeatLegRoom">
                <rdfs:domain rdf:resource="#PassengerVehicle"/>
                <rdfs:range rdf:resource="&xsd;integer"/>
        </rdf:Property>
        <rdf:Property rdf:ID="driver">
                <rdfs:domain rdf:resource="#MotorVehicle"/>
        </rdf:Property>
        <rdf:Property rdf:ID="primaryDriver">
                <rdfs:subPropertyOf rdf:resource="#driver"/>
        </rdf:Property>
</rdf:RDF>
```

Il codice precedente mostra come descrivere classi e proprietà usando RDF Schema, l'esempio successivo mostra come descrivere una istanza della classe ex:PassengerVehicle.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:ex="http://example.org/schemas/vehicles#" xml:base="http://example.org/things">
        <ex:PassengerVehicle rdf:ID="johnSmithsCar">
                <ex:registeredTo rdf:resource="http://www.example.org/staffid/85740"/>
                <ex:rearSeatLegRoom rdf:datatype="&xsd;integer">127</ex:rearSeatLegRoom>
                <ex:primaryDriver rdf:resource="http://www.example.org/staffid/85740"/>
        </ex:PassengerVehicle>
</rdf:RDF>
```

Questo esempio assume che questa istanza venga descritta separatamente dal documento contenente lo schema RDF.

Da notare che la proprietà ex:registerTo è usata per descrivere questa istanza della classe ex:PassangerVehicles in quanto ex:PassangerVehicle è una sottoclasse della classe ex:MotorVehicle.

ALTRE INFORMAZIONI IN RDF SCHEMA

RDF Schema fornisce altre proprietà predefinite per descrivere le informazioni. Queste proprietà sono state descritte nel capitolo che tratta dei vocabolari RDF. Per esempio, la proprietà rdfs:comment viene usata per fornire una versione leggibile da un utente umano della descrizione della risorsa, la proprietà rdfs:label viene usata per fornire una versione leggibile da un utente umano del nome della risorsa.

Per maggiori informazioni su RDF Schema vedi bibliografia [3].

3.6 INTERPRETARE DICHIARAZIONI DI SCHEMI RDF

Da notare come RDF Schema ha molte similitudini con i linguaggi di programmazione orientati agli oggetti. Comunque esistono alcune importanti differenze.

La prima differenza importante è come descrivono le proprietà appartenenti a una classe. RDF Schema descrive le proprietà usando le proprietà range e domain. Per esempio, un tipico linguaggio di programmazione orientato agli oggetti definirebbe una classe book con un attributo author con valore di tipo Person. Il corrispondente RDF Schema definirebbe una classe ex:Book e, in una descrizione separata, una proprietà ex:author con rdfs:domain ex:Boox e rdfs:range exPerson.

Questa differenza potrebbe sembrare solo sintattica, ma in effetti è una differenza importante. Nei linguaggi di programmazione orientati agli oggetti la descrizione dell'attributo è parte della descrizione della classe e l'attributo può essere applicato solo alle istanze della classe descritta. Un'altra classe, che avesse lo stesso attributo sarebbe considerato come attributo differente. In altre parole, lo scopo di una dichiarazione di un attributo nei linguaggi di programmazione orientati agli oggetti è ristretta alla classe in cui è definita. In RDF, le descrizioni delle proprietà sono, di default, indipendenti dalla definizione di una classe, e hanno un scopo globale.

Il risultato è che in RDF Schema si possono definire proprietà senza definire il suo dominio. Questa proprietà potrà quindi essere usata per descrivere istanze di tutte le classi in cui ha senso definire tale attributo. Uno dei benefici dell'approccio di dichiarazione delle proprietà in RDF è che diventa estremamente facile estendere l'uso di una definizione di proprietà a situazioni che non erano state preventivate nel progetto originale. Allo stesso tempo, però, il beneficio va usato con cautela, in quanto un uso incauto potrebbe portare all'applicazione di proprietà in situazioni inadeguate.

Un altro risultato del metodo di descrizione delle proprietà di RDF è che non è possibile definire in RDF Schema specifiche proprietà che hanno specifici range a seconda della classe a cui sono applicati. Ad esempio, nel definire la proprietà ex:hasParent, si può desiderare che sia applicata per descrivere risorse della classe ex:Human e che abbia come valore una istanza della classe ex:Human. Una volta definita la proprietà in questo modo non è possibile descrivere il fatto che la proprietà potesse essere applicata a classi di tipo ex:Tiger e, in questo caso debba avere valore ex:Tiger.

Un'altra importante differenza è che le descrizioni in RDF Schema non sono necessariamente vincolanti al contrario delle dichiarazioni nei linguaggi di programmazione orientati agli oggetti. Ad esempio, se in un linguaggio di programmazione si dichiara la classe Book con un attributo author che ha come valore un'istanza della classe Person, questo viene interpretato, di solito, come un vincolo. Il linguaggio, infatti, non permetterà di creare una istanza della classe Book senza l'attributo author, e non permetterà di creare una istanza della classe Book con l'attributo author che non abbia un'istanza della classe Person come valore. In più, se author è l'unico attributo definito nella classe Book, il linguaggio non permetterà di creare istanze della classe Book con qualche altro attributo.

RDF Schema, al contrario, fornisce informazioni addizionali per la descrizione di risorse, ma non vincola l'applicazione a rispettare questa descrizione. Per esempio, supponiamo che una

descrizione RDF Schema dichiari che una proprietà ex:author abbia un rdfs:range di tipo ex:Person. Questa semplice terna dichiara che dichiarazioni RDF contenenti la proprietà ex:author avranno un'istanza di ex:Person come oggetto.

Queste informazioni addizionali sono usate in differenti maniere.

- Una prima applicazione può interpretare questa dichiarazione come una parte di una specifica dichiarazione di dati RDF e usarla per assicurare che la proprietà ex:author abbia come valore un'istanza della classe indicata. Questa applicazione interpreta la descrizione come un vincolo come farebbe un linguaggio di programmazione.
- Una seconda applicazione potrebbe interpretare questa dichiarazione come un supplemento di informazione su dei dati ricevuti, informazioni che non erano esplicitamente forniti nei dati originali. Per esempio, questa seconda applicazione potrebbe ricevere dei dati RDF che contengono la proprietà ex:author che ha un valore di una classe non specificata e usare la dichiarazione per concludere che la risorsa deve essere un'istanza della classe ex:Person.
- Una terza applicazione potrebbe ricevere alcuni dati RDF che includono la proprietà ex:author che ha come valore un'istanza della classe ex:Corporation e usare la dichiarazione dell'esempio per avvertire che "questa informazione è inconsistente qui, ma da un'altra parte potrebbe non esserlo". Da qualche altra parte, infatti, potrebbe esistere una dichiarazione che risolve l'apparente inconsistenza, ad esempio la dichiarazione che una corporazione è una persona legale.

In conclusione, a seconda di come una applicazione interpreta le descrizioni di proprietà, una descrizione di una istanza può essere considerata valida senza che definisca proprietà specificate nello schema (ad esempio un'istanza della classe ex:Book senza la proprietà ex:author, quando la proprietà ex:author è stata descritta con un dominio di tipo ex:Book) o con proprietà addizionali (ad esempio, un'istanza della classe ex:Book con la proprietà ex:technicalEditor, quando lo schema descrive la classe Book ma non la proprietà technicalEditor):

In altre parole, dichiarazioni in RDF Schema sono sempre descrizioni. Esse non sono vincolanti, ma solo se l'applicazione le interpretano come tali lo diventano.

4. APPLICAZIONE REALIZZATA

In questo capitolo vengono descritte le modilità con cui è stata realizzata l'applicazione e quali sono state le difficoltà riscontrate. Il problema affrontato consiste nel creare delle pagine Web dinamiche per la progettazione concettuale ER di database e la traduzione degli stessi in RDF. Dato lo schema ER in figura:

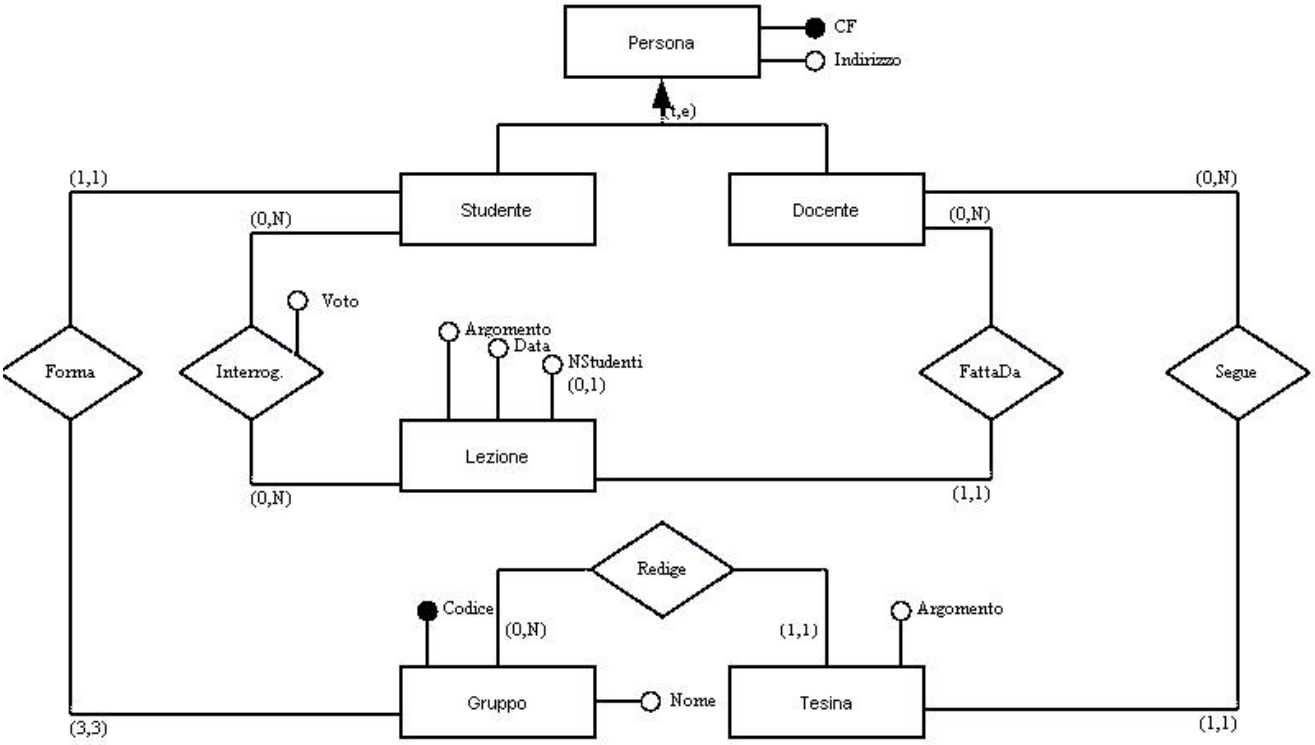

Figura 22 - Esempio schema ER

l'applicazione deve fornire all'utente un'interfaccia grafica per il suo disegno e deve fornirne una sua rappresentazione in RDF. L'applicazione deve permettere inoltre la traduzione dello schema ER disegnato in codice RDF/XML che contenga anche le informazioni di posizionamento degli elementi dello schema all'interno della finestra.

In particolare lo schema precedente, tradotto senza informazioni di posizionamento, genera il seguente codice RDF/XML:

```
<?xml version="1.0"?>
<! DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns# xmlns:er="http://localhost/TESI/RDF/Vocabulary/"
                              xmlns:dt="http://localhost/TESI/RDF/Datatype/" xml:base="http://localhost/TESI/RDF">
  <rdf:description rdf:about="#Persona">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
          <er:haAttributi>
                <rdf:description rdf:about"#Persona:Indirizzo">
                   <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
                </description>
                <rdf:description rdf:about"#Persona:CF">
                   <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
```

```
                </description></er:haAttributi>
        <er:haIdentificatori><er:Identificatore rdf:resource="Persona:CF"/></er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#Studente">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
        <er:haAttributi></er:haAttributi><er:haIdentificatori></er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#Docente">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
        <er:haAttributi></er:haAttributi><er:haIdentificatori></er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#Lezione">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
        <er:haAttributi>
           <rdf:description rdf:about"#Lezione:Argomento">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
           </description>
           <rdf:description rdf:about"#Lezione:Data">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
           </description>
           <rdf:description rdf:about"#Lezione:NStudenti">
             <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#AttributoMultiplo"/>
             <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard">
                                                                        0</er:MinimaCardinalita>
            <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard">
                                                                        1</er:MassimaCardinalita>
           </description>
        </er:haAttributi><er:haIdentificatori></er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#Tesina">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
        <er:haAttributi>
           <rdf:description rdf:about"#Tesina:Argomento">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
           </description>
        </er:haAttributi><er:haIdentificatori></er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#Gruppo">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#entita"/>
        <er:haAttributi>
            <rdf:description rdf:about"#Gruppo:Codice">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
            </description>
            <rdf:description rdf:about"#Gruppo:Nome">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
           </description>
        </er:haAttributi>
        <er:haIdentificatori>
           <er:Identificatore rdf:resource="Gruppo:Codice"/>
        </er:haIdentificatori>
</rdf:description>
<rdf:description rdf:about="#FattaDa">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#AssociazioneBinaria"/>
        <er:haAttributi></er:haAttributi>
        <er:primaPartecipazione>
```

```
            <rdf:description rdf:about"#FattaDa:Docente:">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                <er:dellaEntita rdf:resource="#Docente"/>
                <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                                    ">0</er:MinimaCardinalita>
               <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                                  ">N</er:MassimaCardinalita>
            </rdf:description>
        </er:primaPartecipazione>
        <er:secondaPartecipazione>
            <rdf:description rdf:about"#FattaDa:Lezione:">
                    <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                    <er:dellaEntita rdf:resource="#Lezione"/>
                    <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                                    ">1</er:MinimaCardinalita>
               <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                                  ">1</er:MassimaCardinalita>
            </rdf:description>
        </er:secondaPartecipazione>
</rdf:description>
<rdf:description rdf:about="#Interrog.">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#AssociazioneBinaria"/>
        <er:haAttributi>
            <rdf:description rdf:about"#Interrog.:Voto">
                    <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Attributo"/>
            </description>
        </er:haAttributi>
        <er:primaPartecipazione>
            <rdf:description rdf:about"#Interrog.:Studente:">
                    <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                    <er:dellaEntita rdf:resource="#Studente"/>
                    <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                                    ">0</er:MinimaCardinalita>
              <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                                  ">N</er:MassimaCardinalita>
            </rdf:description>
        </er:primaPartecipazione>
        <er:secondaPartecipazione>
            <rdf:description rdf:about"#Interrog.:Lezione:">
                    <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                    <er:dellaEntita rdf:resource="#Lezione"/>
                    <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                                    ">0</er:MinimaCardinalita>
             <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                                  ">N</er:MassimaCardinalita>
           </rdf:description>
        </er:secondaPartecipazione>
</rdf:description>
<rdf:description rdf:about="#Redige">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#AssociazioneBinaria"/>
        <er:haAttributi></er:haAttributi>
        <er:primaPartecipazione>
            <rdf:description rdf:about"#Redige:Gruppo:">
                    <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                    <er:dellaEntita rdf:resource="#Gruppo"/>
```

```
                <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                        ">0</er:MinimaCardinalita>
            <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                      ">N</er:MassimaCardinalita>
            </rdf:description>
        </er:primaPartecipazione>
        <er:secondaPartecipazione>
            <rdf:description rdf:about"#Redige:Tesina:">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                <er:dellaEntita rdf:resource="#Tesina"/>
                <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                        ">1</er:MinimaCardinalita>
            <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                      ">1</er:MassimaCardinalita>
            </rdf:description>
        </er:secondaPartecipazione>
</rdf:description>
<rdf:description rdf:about="#Segue">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#AssociazioneBinaria"/>
                <er:haAttributi></er:haAttributi>
        <er:primaPartecipazione>
            <rdf:description rdf:about"#Segue:Docente:">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                <er:dellaEntita rdf:resource="#Docente"/>
                <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                        ">0</er:MinimaCardinalita>
            <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                      ">N</er:MassimaCardinalita>
            </rdf:description>
        </er:primaPartecipazione>
        <er:secondaPartecipazione>
            <rdf:description rdf:about"#Segue:Tesina:">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                <er:dellaEntita rdf:resource="#Tesina"/>
                <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                        ">1</er:MinimaCardinalita>
            <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                      ">1</er:MassimaCardinalita>
            </rdf:description>
        </er:secondaPartecipazione>
 </rdf:description>
<rdf:description rdf:about="#Forma">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#AssociazioneBinaria"/>
        <er:haAttributi></er:haAttributi>
        <er:primaPartecipazione>
            <rdf:description rdf:about"#Forma:Studente:">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                <er:dellaEntita rdf:resource="#Studente"/>
                <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                        ">1</er:MinimaCardinalita>
            <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                      ">1</er:MassimaCardinalita>
            </rdf:description>
        </er:primaPartecipazione>
        <er:secondaPartecipazione>
```

```
            <rdf:description rdf:about"#Forma:Gruppo:">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Partecipazione"/>
                <er:dellaEntita rdf:resource="#Gruppo"/>
                <er:minimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#minCard
                                                                    ">3</er:MinimaCardinalita>
              <er:massimaCardinalita rdf:datatype="http://localhost/TESI/RDF/Datatype/#maxCard
                                                                    ">3</er:MassimaCardinalita>
            </rdf:description>
        </er:secondaPartecipazione>
    </rdf:description>
    <rdf:description rdf:about"#Gen:Persona:Studente:Docente">
        <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#Generalizzazione"/>
        <er:copertura rdf:datatype="http://localhost/TESI/RDF/Datatypes/#TipoDiCopertura">(t,e)</er:copertura>
        <er:listaFigli>
            <rdf:description rdf:about="#Gen:Persona:Studente:Docente:Figlio1">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#FiglioGeneralizzazione"/>
                <er:Entita rdf:resource="#Studente">
            </rdf:description>
            <rdf:description rdf:about="#Gen:Persona:Studente:Docente:Figlio1">
                <rdf:type rdf:resource="http://localhost/TESI/RDF/Vocabulary/#FiglioGeneralizzazione"/>
                <er:Entita rdf:resource="#Docente">
             </rdf:description>
        </er:listaFigli>
     </rdf:description>
</rdf:RDF>
```

I prossimi paragrafi saranno dedicati alla descrizione di come è stata realizzata l'applicazione. In particolare i primi due paragrafi si occuperanno della scrittura del codice necessario per la progettazione di database attraverso una pagina Web dinamica; il terzo e il quarto paragrafo della scrittura delle definizioni RDF necessarie per l'applicazione e, infine, il quinto paragrafo si occuperà della pagina Web dinamica per la traduzione dello schema ER disegnato in codice RDF / XML.

4.1 REALIZZAZIONE DI CLASSI PER LA RAPPRESENTAZIONE DEGLI ELEMENTI DI UNO SCHEMA ER

Per realizzare una pagina Web dinamica per il disegno di schemi ER è necessario creare classi che rappresentino tutti i tipi di elementi che possono essere disegnati in uno schema ER. Durante questa fase della realizzazione dell'applicazione le problematiche maggiori sono state quelle riguardanti l'identificazione degli elementi di uno schema ER. Un esempio sono le classi *Partecipazione* e *Associazione.* Dal punto di vista dell'utente è normale creare 2 o più partecipazioni per ogni associazione. La soluzione più immediata sarebbe stata quella di rappresentare partecipazioni e associazioni come un elemento unico, memorizzando le loro caratteristiche in un unico oggetto. Questa soluzione sarebbe stata però troppo complicata per l'utente che sarebbe stato costretto a selezionare almeno tre punti sullo schermo (uno per posizionare l'associazione e almeno due per selezionare i punti di congiunzione con le entità) e specificare molte informazioni tutte in una volta (quali vertici del rombo dell'associazione utilizzare, quali sono le cardinalità e quali le etichette). Inoltre, sarebbe stato troppo complicato anche dal punto di vista del codice necessario per disegnare l'intero elemento (se una partecipazione ha già 16 combinazioni diverse a seconda del vertice a cui si collega all'associazione e dal verso con cui si collega all'entità, figuriamoci quante ce ne vorrebbero per disegnare 2 o più partecipazioni e una associazione in un colpo solo).

Per questo motivo si è scelto di creare più classi di quelle strettamente necessarie, ognuna delle quali rappresenta un singolo e semplice elemento di uno schema ER. Sono state così definite la classe *Entità*, la classe *Attributo*, la classe *Associazione*, la classe *Partecipazione*, la classe *Subset*, la classe *Gerarchia*, la classe *AttributoComposto*, la classe *IdentificazioneEsterna* e la classe *Identificatore*.

Figura 23 - Elementi di uno schema ER

Un secondo problema affrontato è stato trovare un metodo per disegnare tutti gli oggetti sullo schermo ad ogni round trip. La classe *System.Web.UI.Control* contiene un metodo chiamato *Render* che viene automaticamente chiamato ogni volta che l'oggetto deve essere disegnato che è adatto allo scopo. Per questo tutte le classi che rappresentano gli elementi sono stati derivate da questa.

Come esempio, scegliamo la classe *AttributoComposto* (scegliamo questa in quanto è una classe abbastanza semplice e che mostra i meccanismi con cui sono state scritte tutte le classi) e consideriamo lo schema riportato in figura. Per ottenerlo è stato necessario disegnare quattro elementi: una entità, un attributo composto e due attributi. I quattro elementi sono, in realtà, quattro oggetti distinti, ognuno dei quali conterrà al proprio interno, oltre alle proprie caratteristiche, il nome dell'oggetto a cui è associato. Considerando il nostro esempio, all'interno dell'oggetto che rappresenta l'attributo composto *a1*, una proprietà specificherà che l'attributo composto è associato all'entità, mentre all'interno degli oggetti che rappresentano gli attributi *b1* e *b2*, una proprietà specificherà che gli attributi sono associati all'attributo composto.

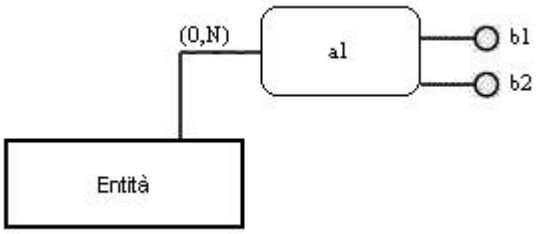

Figura 24 - Attributo comosto

Riportiamo qui di seguito il codice della classe *AttributoComposto* dove, per quanto appena detto, ci aspettiamo di trovare una proprietà che contenga il nome dell'enità a cui è associato e nessuna informazione su quali sono gli attributi che compongono l'attributo composto.

```
namespace LibreriaDiClassi
{
```

Il frammento di codice riportato qui sopra specfica che la classe definita dovrà essere inserita nel namespace *LibreriaDiClassi*. Tutte le classi che sono state create per questa applicazione sono state inserite in questo namespace.

```
    public class AttributoComposto:System.Web.UI.Control
    {
```

Il frammento di codice precedente dichiara l'inizio del codice che descrive la classe *LibreriaDiClassi.AttributoComposto* che è una sottoclasse della classe *System.Web.UI.Control*.

```
        public string nome;
        public int posX;
        public int posY;
        public string entita;
        public string verso;
        public string minCard;
```

```
public string maxCard;
public int x;
public int y;
```

In questa sezione vengono dichiarate gli attributi della classe *AttributoComposto*. Le proprietà sono state definite di tipo pubblico in modo che siano accessibili direttamente dall'esterno. In particolare l'attributo *nome* contiene il nome dell'attributo composto, l'attributo *entità* specifica il nome dell'entità a cui ha associato l'attributo composto, gli attributi *minCard* e *maxCard* rappresentano la cardinalità minima e massima dell'attributo composto e i restanti cinque attributi servono per il posizionamento dell'elemento all'interno della pagina Web dinamica, in particolare, *posX* e *posY* contengono le coordinate della posizione in cui l'attributo composto appoggia sull'entità, *x* e *y* sono le coordinate del centro del rettangolo smussato che contiene il nome dell'attributo composto e *verso* contiene il verso in cui sta l'attributo composto rispetto all'entità.

```
public void Imposta(int xp, int yp, string nome, string entita, string verso,
                                                    string minCard, string maxCard)
{
        this.posX = xp;
        this.posY = yp;
        this.nome = nome;
        this.entita = entita;
        this.verso = verso;
        this.maxCard = maxCard;
        this.minCard = minCard;
        if(verso=="NE"){ x=posX+80; y=posY-21-22;}
        if(verso=="NO"){ x=posX-80; y=posY-21-22;}
        if(verso=="SE"){ x=posX+80; y=posY+21+23;}
        if(verso=="SO"){ x=posX-80; y=posY+21+23;}
        if(verso=="E"){ x=posX+80; y=posY;}
        if(verso=="O"){ x=posX-80; y=posY;}
}
```

Il metodo *Imposta* serve per impostare tutti gli attributi della classe. Tutti gli attributi vengono copiati da quelli in ingresso tranne *x* e *y* che vengono calcolati a seconda di *verso*, *posX* e *posY*.

```
public char VersoAttributo(int xp, int yp)
{
        if((xp>=x-40 && xp<=x+40) && (yp<=y-22) &&
                        (verso=="NE" || verso=="NO" || verso=="E" || verso=="O" ))
                return 'N';
        if((xp>=x-40 && xp<=x+40) && (yp>=y+22) &&
                        (verso=="SE" || verso=="SO" || verso=="E" || verso=="O" ))
                return 'S';
        if((xp>=x+40) &&  (yp>=y-23 && yp<=y+23)
                                && (verso=="E" || verso=="NE" || verso=="SE"))
                return 'E';
        if((xp<=x-40) && (yp>=y-23 && yp<=y+23) &&
                                (verso=="O" || verso=="NO" || verso=="SO"))
                return 'O';
        return ' ';
```

```
        }
```

Il metodo *VersoAttributo* serve per calcolare quale verso avrà un attributo associato all'attributo composto di questo oggetto a seconda delle coordinate *xp* e *yp* in cui l'utente ha fatto click. Oltre a verficare la direzione, il metodo verifica anche che la direzione sia giusta rispetto al posizionamento dell'entità. Ad esempio, se posiziono un attributo composto sul lato est di una entità, non si potrà aggiungere un attributo sul lato ovest dell'attributo composto altrimenti questo sicuramente finirà sull'entità. In caso di errore il metodo restituisce il carattere ' '.

Metodi simili sono contenuti in tutte le classi che prevedono la loro relazione con un atributo, naturalmente i controlli sul corretto posizionamento varieranno a seconda della classe.

```
        protected override void Render(System.Web.UI.HtmlTextWriter writer)
        {
            writer.Write("<IMG alt=\"\" src=\"Immagini\\AttributoComposto.gif\"
                                                    style=\"LEFT: "+(x-40).ToString() +
                    "px; POSITION: absolute; TOP: " + (y-23).ToString() + " px\">");
            writer.Write("<DIV style=\"FONT-SIZE: 8pt; WIDTH: 80px;
                    BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none;
                        BORDER-LEFT-STYLE: none; HEIGHT: 15px; BORDER-
                        BOTTOM-STYLE: none; LEFT: "+(x-40).ToString() + "px;
                            POSITION: absolute; TOP: " + (y-8).ToString() + " px\"
                                    align=\"center\">" + this.nome + "</DIV>");
            if(verso=="NE")
            {
                writer.Write("<IMG alt=\"\" src=\"Immagini\\LineaV.gif\"
                    width=\"14\" height=\""+ (posY-y).ToString()  +"\" style=\"LEFT:
                        "+(posX-7).ToString() + "px; POSITION: absolute; TOP: " +
                                                        (y).ToString() + " px\">");
                writer.Write("<IMG alt=\"\" src=\"Immagini\\LineaO.gif\"
                                height=\"14\" width=\""+ (x-posX-40).ToString()  +"\"
                            style=\"LEFT: "+ (posX).ToString() + "px; POSITION:
                                    absolute; TOP: " + (y-7).ToString() + " px\">");
                writer.Write("<DIV style=\"FONT-SIZE: 8pt; WIDTH: 40px;
                    BORDER-TOP-STYLE: none; BORDER-RIGHT-STYLE: none;
                        BORDER-LEFT-STYLE: none; HEIGHT: 15px; BORDER-
                        BOTTOM-STYLE: none; LEFT: "+(posX).ToString() + "px;
                            POSITION: absolute; TOP: " + (y-16).ToString() + " px\"
                        align=\"left\">(" + this.minCard +","+this.maxCard + ")</DIV>");
            }
            //Atri valori dell'attributo verso
        }
    }
}
```

Il metodo *Render* è un metodo definito nella classe *System.Web.UI.Control* e viene richiamato tutte le volte che deve essere disegnata la parte grafica dell'oggetto. In questo caso, il metodo viene utilizzato per scrivere i tag HTML necessari per posizionare correttamente tutte le immagini e le etichette che compongono la parte grafica di un attributo composto. L'oggetto *writer* di tipo *HtmlTextWriter* è un oggetto specifico per scrivere il codice HTML generato dal controllo. Questo metodo, nel nostro caso,

restituirà il codice HTML di due etichette (una che contiene il nome dell'attributo e una che contiene la cardinalità) e tre immagini (un rettangolo con gli angoli smussati e due linee) posizionati in modo da formare l'immagine di un attributo composto.

Come indicato dal commento, il codice comprende anche le altre cinque possibilità sul verso dell'attributo composto che qui non sono state riportate per brevità.

Tutte le altre classi che rappresentano gli elementi di uno schema ER sono costruite nello stesso modo, anche se ogni classe contiene qualche peculiarità nel codice.

4.2 REALIZZAZIONE DELLA PAGINA WEB PER IL DISEGNO DI SCHEMI ER

La pagina Web dinamica realizzata con teconologia ASP.NET è strutturata in maniera molto semplice. In alto, la prima riga contiene tutti i pulsanti che permettono di scegliere quale elemento dello schema ER disegnare. La seconda riga contiene uno spazio libero in cui appaiono i controlli per gli input dei dati necessari per creare un elemento dello schema. Sotto, la pagina contiene una immagine ciccabile su cui verranno disegnati gli elementi selezionati dall'utente. L'immagine è contenuta all'interno di un pannello, in modo da sfruttare la sua proprietà *Controls* che contiene una collezione di controlli aggiunti dinamicamente. Questa proprietà è quella che contiene tutti gli oggetti creati dall'utente.

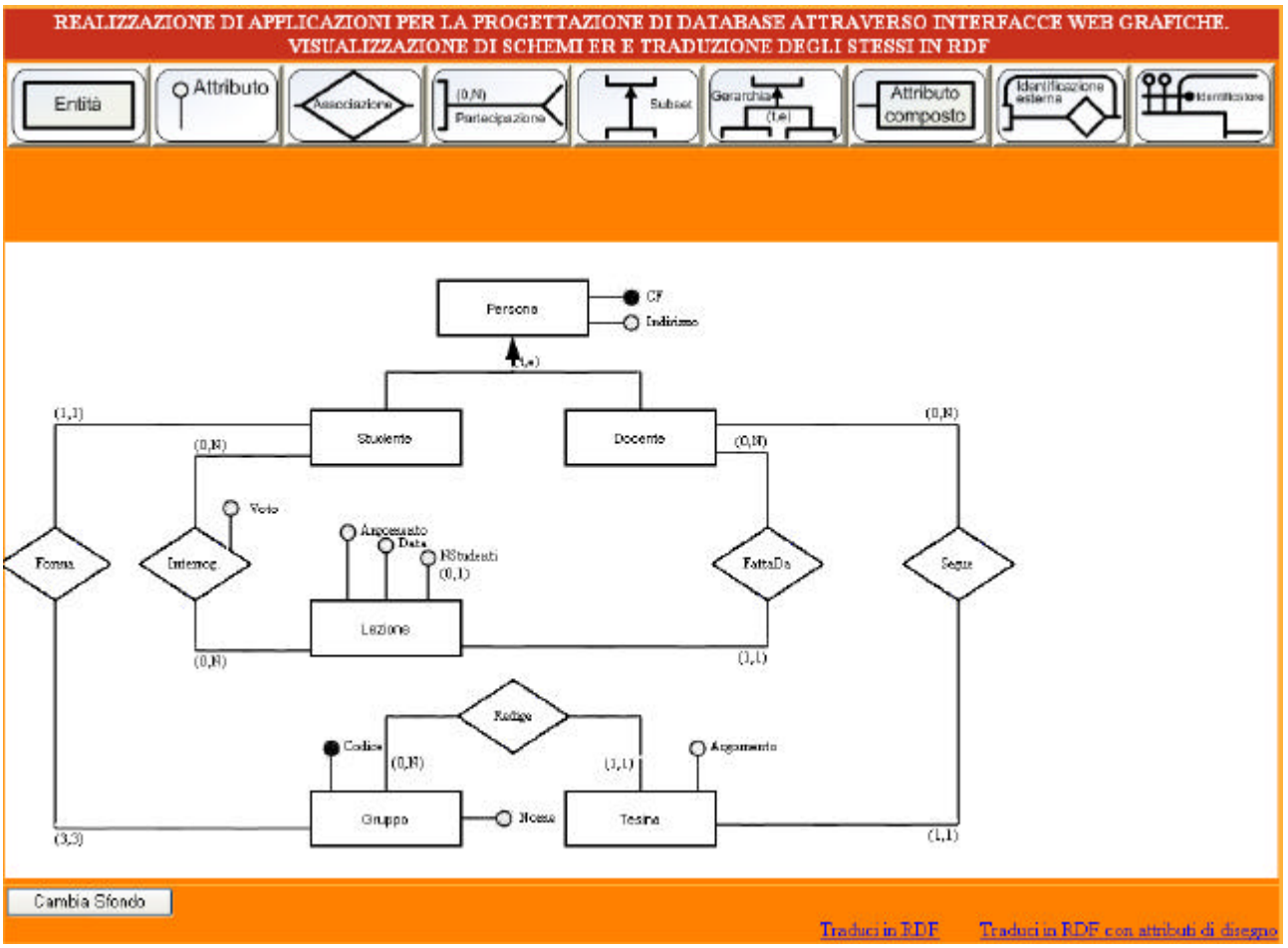

Figura 25 - Schermata dell'applicazione

Il primo problema da affrontare è stato che la proprietà *Controls* del pannello non viene salvata mediate il meccanismo view state, quindi i controlli aggiunti al pannello vengono persi tra un round trip e il successivo. Questo è stato un grosso problema, in quanto è fondamentale mantenere gli elementi che l'utente ha gia posizionato visto che ad ogni click viene generato un nuovo round trip e che per posizionare un nuovo elemento servono almeno tre click.

Tra le possibili soluzioni del problema (discusse nel paragrafo 2.1.5) si è scelto di usare una variabile sessione che contenga al suo interno un vettore con gli oggetti creati dall'utente. A ogni round trip i controlli contenuti nella proprietà *Controls* del pannello vengono inseriti in un nuovo vettore creato per contenere tanti elementi quanti quelli contenuti nel pannello e il vettore viene salvato in una variabile sessione. Dopo il round trip l'applicazione si occupa di ricaricare gli oggetti salvati in precedenza all'interno del pannello. Visto che le variabili sessione rimangono disponibili anche se non vengono ricreate ad ogni round trip, non è necessario salvare i controlli ad ogni roun trip, ma solo quando viene aggiunto un nuovo elemento allo schema. Qui di seguito vengono riportati i due metodi creati all'inteno della pagina per salvare e caricare i componeti.

```
public void SalvaComponenti()
{
        Control[] cl = new Control[Panel1.Controls.Count];
        Panel1.Controls.CopyTo(cl,0);
        Session["Controls"] =  cl;
}
```

```
public void CaricaComponenti()
{
        if (Session["Controls"]!=null)
        {
                Control[] cc = (Control[])Session["Controls"];
                foreach(Control c in cc)
                {
                        if ((c as Control)!=null)
                        {
                                if (c.GetType().ToString().IndexOf("LibreriaDiClassi")>=0)
                                        Panel1.Controls.Add(c);
                        }
                }
        }
}
```

Il secondo problema da affrontare è stato quello di tenere traccia dello stato del disegno, cioè tenere traccia di qual è stata l'azione precedente dell'utente. Se, ad esempio, l'utente preme il pulsante per posizionare una entità, nel round trip successivo un click sull'immagine di sfondo del disegno dovrà reagire in modo differente dal solito, cioè dovrà richiamare la procedura per la creazione di un'entità.

Per memorizzare questa informazione è stata creata una classe *StatoDisegno* ed è stato inserito nella pagina un oggetto *StatoDisegno1* di questo tipo che contiene al suo interno un attributo stato di tipo stringa che contiene una stringa associata all'ultimo pulsante premuto.

Il codice della classe *StatoDisegno* è riportato qui di seguito.

```
namespace LibreriaDiClassi
{
```

```
	public class StatoDisegno: System.Web.UI.Control
	{
		public string stato
		{
			get{return (string)ViewState["stato"];}
			set{ViewState["stato"]=value;}
		}
		protected override void Render(System.Web.UI.HtmlTextWriter writer)
		{
			writer.RenderBeginTag(System.Web.UI.HtmlTextWriterTag.B );
			writer.Write(stato);
			writer.RenderEndTag();
		}
	}
}
```

Da notare che nel metodo *Render* è stato inserito il codice per visualizzare nella posizione del controllo il valore dell'attributo stato scritto in grossetto. Normalmente questo non è visibile perché il controllo viene nascosto attraverso l'attributo visibile ereditato dalla classe *Control*.

Da notare anche, che per questa classe, il salvataggio dell'attributo stato della classe con il metodo view state è stato scritto in modo esplicito anche se non era necessario visto che viene ereditato dalla classe *System.Web.UI.Control*.

Consideriamo ora come vengono creati, in generale, nuovi elementi dello schema ER. Premendo il pulsante in alto relativo all'elemento che si vuole creare, appare subito sotto un pannello che contiene tutti i controlli per l'input delle caratteristiche dell'elemento. Ad esempio, se selezioniamo il pulsante per creare un attributo composto verrà eseguito il seguente metodo:

```
private void cmdAttributoComposto_Click(object sender, System.Web.UI.ImageClickEventArgs e)
{
	this.CaricaComponenti();
	this.PulisciPannello();
	StatoDisegno1.stato = "Attributo composto";
	this.AzzeraCompinenti();
	PanelAttributoComposto.Visible = true;
	if (Session["Controls"]!=null)
	{
		Control[] cc = (Control[])Session["Controls"];
		foreach(Control c in cc)
		{
			if ((c as Control)!=null)
			{
				if (c.GetType().ToString().Equals("LibreriaDiClassi.Entita"))
				{
			cmbAttributoCompostoEntita.Items.Add(((LibreriaDiClassi.Entita)c).name);
				}
			}
		}
	}
```

```
}
```

Come si può notare dal codice, la prima riga richiama la procedura per caricare i componenti all'interno del pannello del disegno, che è stata già descritta precedentemente.

La seconda riga, richiama una procedura che serve a eliminare alcuni elementi dello schema il cui posizionamento non è stato portato a termine. Per capirne l'utilità consideriamo, ad esempio, il caso in cui l'utente decida di creare una gerarchia. Per poter posizionare correttamente la gerarchia è necessario indicare un punto di congiunzione con l'entità padre e tanti punti di congiunzione quante sono le entità figlie. Se l'utente dopo aver indicato alcuni punti, ma senza averli indicati tutti, preme sul pulsante per creare un attributo composto (o su uno qualsiasi dei pulsanti della prima riga della pagina) all'interno della collezione di oggetti del pannello sarebbe rimasta una gerarchia creata a metà. Questa procedura elimina tutti gli elementi come questa gerarchia.

La terza riga di codice imposta la variabile stato dell'oggetto *StatoDisegno1* con il valore associato all'ultimo pulsante premuto. L'utilità di questa operazione è già stata presa in considerazione precedentemente.

La quarta riga chiama la procedura *AzzeraComponenti*. Per capire l'utilità di questa procedura bisogna analizzarla con la riga successiva che mostra il pannello *PanelAttributoComposto*. L'utilità della procedura è questa: i controlli contenuti all'interno del pannello, quando vengono mostrati ci si aspetta che siano vuoti (per quanto riguarda i campi testo) e che le liste siano aggiornate. Questo è vero se è la prima volta che si aggiunge un attributo composto, ma se precedentemente si era gia aggiunto un attributo composto i campi saranno ancora occupati dai valori precedenti che vengono automaticamente salvati con il metodo view state dal framework. Inoltre tutti i controlli saranno disabilitati dal pulsante posiziona (come si vedrà in seguito). Per riportare tutti i controlli allo stato originale, cioè vuoti e abilitati è stata creata questa procedura. Per semplicità si è preferito creare una procedura unica che azzeri tutti i componenti dei pannelli che contengono gli input per creare elementi, anzichè di crearene 9, ognuna specifica per ogni pulsante della prima riga della pagina. La procedura AzzeraComponenti, inoltre, nasconde anche tutti i pannelli di input (trenne quello che contiene gli elementi disegnati) che sono eventualmente rimasti visibili.

Le ultime righe del codice leggono la variabile sessione che contiene tutti i controlli aggiunti al disegno, li scorre tutti alla ricerca di oggetti di tipo *LibreriaDiClasse.Entita*, grazie al metodo *GetType* della classe *Control* che permette di conoscere il tipo di una classe, e aggiunge i nomi delle entità a una lista che servirà all'utente per selezionare a quale entità associare l'attributo composto.

A questo punto nella pagina sarà stato visualizzato il pannello *PanelAttributoComposto* che si presenterà così:

Figura 26 - Applicazione – PanelAttributoComposto

A questo punto l'utente inserirà il nome dell'attributo composto e la sua cardinalità e selezionerà a quale entità associare l'attributo composto. Al termine dell'inserimeto dei dati cliccherà sul pulsante Posiziona attributo composto richiamerà il seguente metodo:

```
private void cmdPosizionaAttributoComposto_Click(object sender, System.EventArgs e)
{
        this.CaricaComponenti();
        //Controlli degli input
```

```
        txtNomeAttributoComposto.Enabled = false;
        cmbAttributoCompostoEntita.Enabled = false;
        txtMinCardAttributoComposto.Enabled = false;
        txtMaxCardAttributoComposto.Enabled = false;
        StatoDisegno1.stato="Posiziona Attributo Composto";
}
```

Da notare che al posto del commento nel codice originale vi sono tutti i controlli su ciò che ha inserito l'utente che vanno dalla verifica che non ci sia un altro attributo nella stessa entità con lo stesso nome, alla verifica che le cardinalità siano corrette.

A questo punto l'applicazione attende che l'utente clicchi su l'immagine di sfondo su cui disegnare i componenti. In risposta a questo click, la procedura scatenata richiama una procedura diversa a seconda del valore dell'attributo stato dell'oggetto *StatoDisegno1*, cioè a seconda di quale elemento si sta cercando di posizionare. In questo caso viene richiamata la seguente procedura:

```
public void PosizionaAttributoComposto(int x, int y)
{
        LibreriaDiClassi.AttributoComposto ac= new LibreriaDiClassi.AttributoComposto();
        LibreriaDiClassi.Entita ent=new LibreriaDiClassi.Entita();
        LibreriaDiClassi.Entita app;
        if (Session["Controls"]!=null)
        {
                Control[] cc = (Control[])Session["Controls"];
                foreach(Control c in cc)
                {
                        if ((c as Control)!=null)
                        {
                                if (c.GetType().ToString().Equals("LibreriaDiClassi.Entita"))
                                {
                                        app=((LibreriaDiClassi.Entita)c);
                                       if(app.name==cmbAttributoCompostoEntita.SelectedItem.Text)
                                                ent=app;
                                }
                        }
                }
        }
        if(ent.VersoAttributo(x,y)=='N' && x>ent.PosX+60)
                ac.Imposta(x,ent.PosY,txtNomeAttributoComposto.Text,ent.name,"NE",
                        txtMinCardAttributoComposto.Text,txtMaxCardAttributoComposto.Text);
        if(ent.VersoAttributo(x,y)=='N' && x<=ent.PosX+60)
                ac.Imposta(x,ent.PosY,txtNomeAttributoComposto.Text,ent.name,"NO",
                        txtMinCardAttributoComposto.Text,txtMaxCardAttributoComposto.Text);
        if(ent.VersoAttributo(x,y)=='S' && x>ent.PosX+60)
                ac.Imposta(x,ent.PosY+45,txtNomeAttributoComposto.Text,ent.name,"SE",
                        txtMinCardAttributoComposto.Text,txtMaxCardAttributoComposto.Text);
        if(ent.VersoAttributo(x,y)=='S' && x<=ent.PosX+60)
                ac.Imposta(x,ent.PosY+45,txtNomeAttributoComposto.Text,ent.name,"SO",
                        txtMinCardAttributoComposto.Text,txtMaxCardAttributoComposto.Text);
```

```
        if(ent.VersoAttributo(x,y)=='E')
                ac.Imposta(ent.PosX+120,y,txtNomeAttributoComposto.Text,ent.name,"E",
                        txtMinCardAttributoComposto.Text,txtMaxCardAttributoComposto.Text);
        if(ent.VersoAttributo(x,y)=='O')
                ac.Imposta(ent.PosX,y,txtNomeAttributoComposto.Text,ent.name,"O",
                        txtMinCardAttributoComposto.Text,txtMaxCardAttributoComposto.Text);
        Panel1.Controls.Add(ac);
        StatoDisegno1.stato="";
        PanelAttributoComposto.Visible = false;
}
```

Come si può vedere la procedura riportata qui sopra crea un nuovo attributo composto, posizionandola a seconda del click dell'utente (parametri *x* e *y* della procedura) e a seconda della posizione dell'entità scelta. Alla fine l'attributo composto viene aggiunto al pannello, viene nascosto il pannello per creare un nuovo attributo e viene azzerato l'attributo stato dell'oggetto *StatoDisegno1*.

Per gli altri elementi che compongono uno schema ER, i passi da compiere per inserirli nel disegno è molto simile. Cambiano i dati da inserire, i controlli da effettuare sia sui dati inseriti dall'utente sia sul posizionamento degli elementi, e il numero di posizioni da specificare per posizionere l'elemento.

4.3 DICHIARAZIONE DI NUOVI TIPI DI DATO RDF

Per poter scrivere le definizioni RDF dei vari elementi che compongono uno schema ER è stato necessario scrivere alcune dichiarazioni di nuovi tipi di dato che non erano stati definiti tra quelli predefiniti di RDF Schema.

Il tipo di dato *minCard*, che rappresenta tutti i possibili valori che può assumere una proprietà cardinalità minima, non è altro che la rinominazione del tipo di nonNegativeInteger di RDFS anche se viene ottenuto derivando quest'ultimo con restrizione.

```
<xsd:simpleType name="minCard">
        <xsd:restriction base="xsd:nonNegativeInteger"/>
</xsd:simpleType>
```

Il tipo di dato *maxCard*, , che rappresenta tutti i possibili valori che può assumere una proprietà cardinalità massima, è stato ottenuto come unione di due tipi di dato. Il primo è il tipo di dato predefinito positiveInteger che contiene tutti i numeri interi positivi non nulli e il secondo tipo di dato è una restrizione del tipo string alle stringhe "N" e "n". Quindi una proprietà che contenga valori di tipo maxCard può assumere come valore tutti i numeri interi positivi non nulli e le stringhe "N" e "n".

```
<xsd:simpleType name="maxCard">
        <xsd:union>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:positiveInteger"/>
                </xsd:simpleType>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                                <xsd:enumeration value="N"/>
                                <xsd:enumeration value="n"/>
                        </xsd:restriction>
```

```
            </xsd:simpleType>
        </xsd:union>
</xsd:simpleType>
```

Il terzo nuovo tipo di dato definito è quello che rappresenta tutti i valori che può assumere una proprietà tipo copertura di una gerarchia in uno schema ER. Esso è stato ottenuto derivando il tipo di dato predefinito string con restrizione alle sole stringhe che verificano una espressione regolare.

```
<xsd:simpleType name="TipoDiCopertura">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="([tp],[eo])"/>
        </xsd:restriction>
</xsd:simpleType>
```

Gli altri tipi di dato (*listaDiAttributi*, *listaDiIdentificatori*, *listaDiPartecipazioni*, *listaDiFigliGeneralizzazione*, *listaDiAssociazioni*) sono tipi di dato ottenuti da una lista di classi definiti nel vocabolario RDF descritto nel paragrafo successivo.

```
<simpleType name="listaDiAttributi">
        <list itemType="http://localhost/RDF/Vocabulary/#Attributo"/>
</simpleType>
<simpleType name="listaDiIdentificatori">
        <list itemType="http://localhost/RDF/Vocabulary/#Identificatore"/>
</simpleType>
<simpleType name="listaDiPartecipazioni">
        <list itemType="http://localhost/RDF/Vocabulary/#Partecipazione"/>
</simpleType>
<simpleType name="listaDiFigliGeneralizzazione">
        <list itemType="http://localhost/RDF/Vocabulary/#FiglioGeneralizzazione"/>
</simpleType>
<simpleType name="listaDiAssociazioni">
        <list itemType="http://localhost/RDF/Vocabulary/#AssociazioneBinaria"/>
</simpleType>
```

4.4 COSTRUZIONE DI UN VOCABOLARIO RDF PER LA DESCRIZIONE DI SCHEMI ER

Dopo aver descritto i nuovi tipi di dato utilizzati nell'applicazione, è necessario descrivere tutte le classi RDF e le proprietà di queste classi per la rappresentazione di schemi ER.

Come descritto nel paragrafo su RDF Schema (3.5), se una definizione di proprietà contiene più rdfs:domain che definiscono il dominio di applicazione della proprietà stessa, allora le istanze che contengono la proprietà in questione devono essere istanze di tutte le classi indicate. Ad esempio, consideriamo la proprietà "*haAttributo*" definita su tre domini la classe *Entità*, la classe *Associazione* e la classe *AttributoComposto*. Il risultato ottenuto è che le uniche istanze che hanno la proprietà *haAttributo* sono le istanze che sono istanze contemporaneamente sia della classe *Entità*, sia della classe *Associazione*, sia della classe *AttributoComposto*. È ovvio che, in questo caso, non è il risultato voluto.

Per ottenere il risultato voluto, cioè una proprietà di nome applicabile a una qualsiasi istanza della classe *Entità* o a una qualsiasi istanza della classe *Associazione* o a una qualsiasi istanza della classe *AttributoComposto*, ci sono due possibili modi. Il primo consiste nel non specificare il dominio di una

proprietà. In questa situazione, la dichiarazione della proprietà viene intesa come applicabile a una istanza di una classe qualsiasi. Sebbene questo metodo sia lecito, è formalmente scorretto in quanto la proprietà può essere applicata a istanze di classi che non hanno nulla a che fare con essa e perdere di significato, oppure essere applicata a una istanza di una classe in cui ha significato, ma un significato diverso rispetto a quello per cui era stata progettata. Per questo motivo, in questa applicazione, si è scelto il secondo modo, cioè quello di creare una nuova classe *ElementoConAttributi* che generalizza le classi *Entità*, *Associazione* e *AttributoComposto* e di specificare questa come dominio della proprietà haAttributo. In questo caso, il risultato ottenuto è proprio quello che si voleva ottenere in quanto le istanze delle classi *Entità*, *Attributo* e *AttributoComposto* sono anche istanze della classe *ElementoConAttributi* e quindi ad esse è applicabile la proprietà *haAttributo*.

In maniera analoga sono state create altre classi che generalizzano le classi che rappresentano gli elementi di uno schema ER per definirne le proprietà comuni.

Questo approccio è stato utilizzato per tutte le proprietà utili per descrivere le caratteristiche di elementi di uno schema ER, ma non per le proprietà che sono state aggiunte per il disegno degli stessi all'interno della pagina Web dinamica. Per queste ultime è stato usato il seguente criterio: se si era definita già una generalizzazione delle classi per altre proprietà, questa veniva utilizzata, altrimenti si preferiva cambiare nome alla proprietà a seconda della classe a cui si riferiscono. Ad esempio: sia le istanze della classe *Entità*, che della classe *Associazione*, che della classe *Attributo*, sia della classe *AttributoComposto*, sia della classe *AttributoMultiplo*, necessitano di due proprietà che descrivano il loro posizionamento assoluto all'interno della pagina. In questo caso, essendo stata definita una classe *Elemento* che direttamente o indirettamente le generalizza tutte, sono state create le proprietà *posizioneX* e *posizioneY* con dominio la classe *Elemento*. Invece, per le istanze delle classi *Partecipazione*, *Identificatore* e *IdentificazioneEsterna*, caratterizzate tutte da due coppie di coordinate e non avendo definito nessuna generalizzazione che le contenesse tutte, non si è definta la classe *ElementoConDueCoordinate* che non avrebbe avuto significato nel contesta di schemi ER, ma si sono create proprietà con nomi diversi che si applicano a una delle classi sopra indicate.

Un'altra nota tecnica è che XML versione 1.0 non supporta caratteri accentanti. Quindi nelle descrizioni degli elementi si sono utilizzati i relativi codici, mentre nei nomi delle classi si sono evitati gli accenti per riuscire a riferirli più semplicemente. Per questo, ad esempio la classe *Entità* è stata chiamata *Entita*.

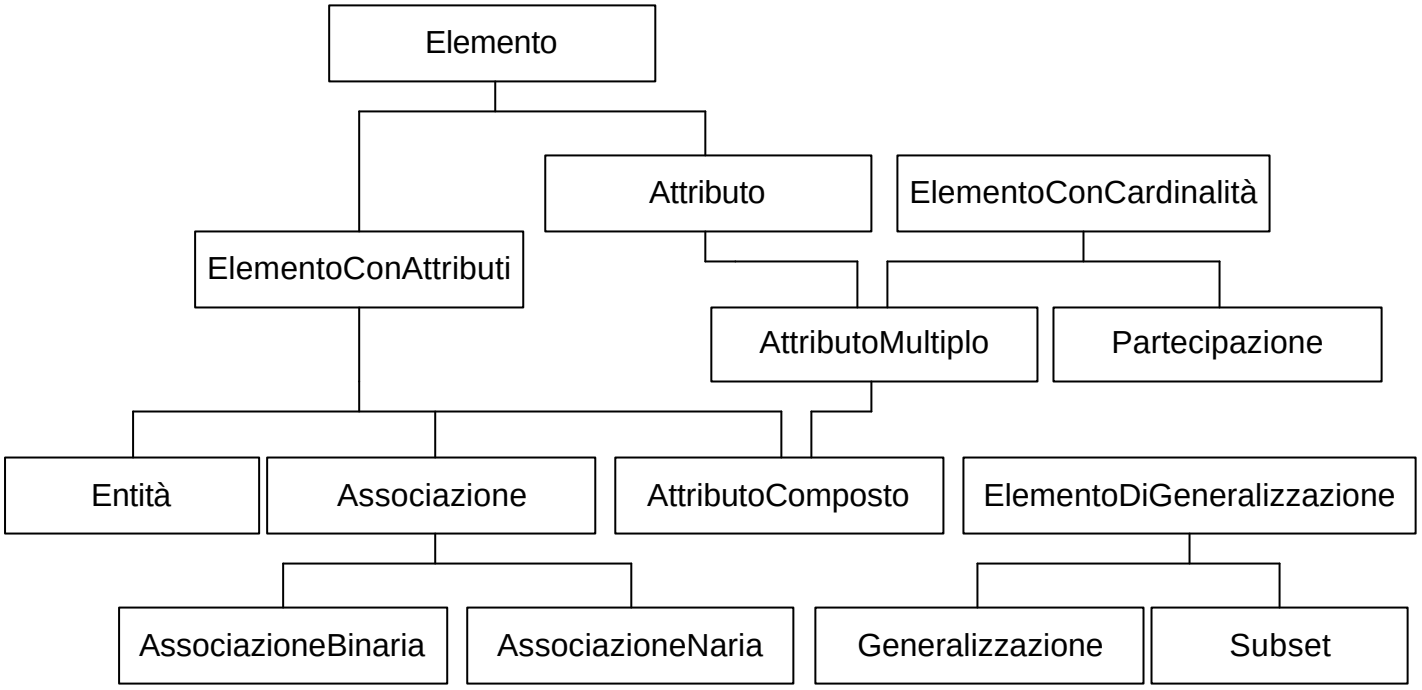

Figura 27 - Applicazione - Gerarchia delle classi

Oltre alle classi mostrate in figura sono state create altre tre classi (*Identificatore*, *IdentificazioneEsterna*, *FiglioGeneralizzazione*) che non hanno però vincoli di parentela con altre classi.

Da notare come in RDF, al contrario dei linguaggi di programmazione ad oggetti classici, sia possibile definire una classe che discende direttamente da due classi. Ad esempio, la classe *AttributoComposto* è figlia contemporaneamente di *ElementoConAttributi* e di *AttributoMultiplo*. Ciò significa che una istanza della classe *AttributoComposto* è sia una istanza della classe *ElementoConAttributi*, sia una istanza della classe *AttributiMultiplo*. Nei linguaggi di programmazione ad oggetti classici, normalmente vi è il vincolo che una classe possa essere figlia di una sola altra classe e che possa implementare più interfacce. In RDF non è così.

Il codice scritto in RDF/XML usato per descrivere il vocabolario dell'applicazione è diviso in due sezioni. Nella prima vengono definite le nuove classi di oggetti utilizzate dall'applicazione. Da notare l'utilizzo della proprietà rdfs:label per dare la versione leggibile dei nomi con accenti (i caratteri accentati vengono codificati) e della proprietà rdfs:comment per dare una descrizione delle classi utilizzate.

```
 <rdf:Description rdf:about="#Elemento">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:about="#ElementoConAttributi">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#Elemento"/>
</rdf:Description>
<rdf:Description rdf:about="#ElementoConCardinalita">
        <rdfs:label>Elemento con cardinalit&#224;</rdfs:label></rdf:Description>
<rdf:Description rdf:about="#Attributo">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#Elemento"/>
```

```
        <rdfs:comment>L'attributo rappresenta propriet&#224; elementari di entit&#224; o
associazioni </rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#Entita">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#ElementoConAttributi"/>
        <rdfs:label>Entit&#224;</rdfs:label>
        <rdfs:comment>Una entit&#224; rappresenta un insieme di oggetti della realt&#224; di cui
si individuano propriet&#224; comuni</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#Associazione">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#ElementoConAttributi"/>
        <rdfs:comment>Una associazione rappresenta un legame logico tra due o pi&#249;
entit&#224;</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#AttributoMultiplo">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#Attributo"/>
        <rdfs:subClassOf rdf:resource="#ElementoConCardinalita"/>
        <rdfs:comment>Un attributo &#232; multiplo se il numero di valori dell'attributo associati a
ogni istanza di un'entit&#224; o di un'associazione pu&#242; essere diverso da 1</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#AttributoComposto">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#AttributoMultiplo"/>
        <rdfs:subClassOf rdf:resource="#ElementoConAttributi"/>
        <rdfs:comment>Un attributo composto &#232; costituito da un gruppo di attributi che
hanno affinit&#224; nel significato o nell'uso</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#AssociazioneBinaria">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#Associazione"/>
        <rdfs:comment>Una associazione rappresenta un legame logico tra due
entit&#224;</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#AssociazioneNaria">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#Associazione"/>
        <rdfs:label>Associazione N-aria</rdfs:label>
        <rdfs:comment>Una associazione rappresenta un legame logico tra pi&#249; di due
entit&#224;</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#Partecipazione">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#ElementoConCardinalita"/>
```

```
</rdf:Description>
<rdf:Description rdf:about="#Identificatore">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:comment>Un identificatore di un'entit&#224; E &#232; una collezione di attributi o
di entit&#224; in associazione con E che individua in modo univoco tutte le istanze di
E</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#ElementoDiGeneralizzazione">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:about="#Generalizzazione">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#ElementoDiGeneralizzazione"/>
        <rdfs:comment>Un'entit&#224; E &#232; una generalizzazione di un gruppo di
entit&#224; se ogni istanza di queste entit&#224; &#232; anche un'istanza dell'entit&#224;
E</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#Subset">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#ElementoDiGeneralizzazione"/>
        <rdfs:comment>Un subset &#232; una gerarchia di generalizzazione con una sola
entit&#224; generalizzata</rdfs:comment>
</rdf:Description>
<rdf:Description rdf:about="#IdentificazioneEsterna">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
</rdf:Description>
<rdf:Description rdf:about="#FiglioGenerlizzazione">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Class"/>
</rdf:Description>
```

Nella seconda sezione del vocabolario dell'applicazione vengono definite le proprietà delle classi sopra descritte. In particolare le proprietà si dividono in due gruppi: il primo descrive le proprietà che ciascun elemento ha per com'è definito (Ad esempio un AttributoMultiplo ha un nome, una cardinalità minima e una cardinalità massima), il secondo gruppo descrive le proprietà necessarie per posizionare l'elemento descritto all'interno dell'applicazione. Per brevità riportiamo qui di seguito solo le definizioni del primo gruppo di proprietà:

```
<rdf:Description rdf:about="#haAttributi">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#ElementoConAttributi"/>
        <rdfs:range rdf:resource="http://localhost/RDF/datatypes/#ListaDiAttributi"/>
</rdf:Description>
<rdf:Description rdf:about="#haIdentificatori">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Entita"/>
        <rdfs:range rdf:resource="http://localhost/RDF/datatypes/#ListaDiIdentificatori"/>
</rdf:Description>
<rdf:Description rdf:about="#primaPartecipazione">
```

```
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AssociazioneBinaria"/>
        <rdfs:range rdf:resource="#Partecipazione"/>
</rdf:Description>
<rdf:Description rdf:about="#secondaPartecipazione">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AssociazioneBinaria"/>
        <rdfs:range rdf:resource="#Partecipazione"/>
</rdf:Description>
<rdf:Description rdf:about="#haPartecipazioni">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#AssociazioneNaria"/>
        <rdfs:range rdf:resource="http://localhost/RDF/datatypes/#ListaDiPartecipazioni"/>
</rdf:Description>
        <rdf:Description rdf:about="#minimaCardinalita">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#ElementoConCardinalita"/>
        <rdfs:range rdf:resource="http://localhost/RDF/datatypes/#minCard"/>
</rdf:Description>
<rdf:Description rdf:about="#massimaCardinalita">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#ElementoConCardinalita"/>
        <rdfs:range rdf:resource="http://localhost/RDF/datatypes/#maxCard"/>
</rdf:Description>
<rdf:Description rdf:about="#etichetta">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Partecipazione"/>
        <rdfs:range rdf:resource="&xsd;string"/>
</rdf:Description>
<rdf:Description rdf:about="#dellaEntita">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Partecipazione"/>
        <rdfs:range rdf:resource="#Entita"/>
</rdf:Description>
<rdf:Description rdf:about="#entitaPadre">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#ElementoDiGeneralizzazione"/>
        <rdfs:range rdf:resource="#Entita"/>
</rdf:Description>
<rdf:Description rdf:about="#entitaFiglio">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Subset"/>
        <rdfs:range rdf:resource="#Entita"/>
</rdf:Description>
<rdf:Description rdf:about="#listaFigli">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
```

```
        <rdfs:domain rdf:resource="#Generalizzazione"/>
        <rdfs:range rdf:resource="#http://localhost/RDF/datatypes/#ListaDiFigliGeneralizzazione"/>
</rdf:Description>
<rdf:Description rdf:about="#copertura">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Generalizzazione"/>
        <rdfs:range rdf:resource="#http://localhost/RDF/datatypes/#TipoDiCopertura"/>
</rdf:Description>
<rdf:Description rdf:about="#entita">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#IdentificazioneEsterna"/>
        <rdfs:range rdf:resource="#Entita"/>
</rdf:Description>
<rdf:Description rdf:about="#tramiteAssociazione">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#IdentificazioneEsterna"/>
        <rdfs:range rdf:resource="#AssociazioneBinaria"/>
</rdf:Description>
<rdf:Description rdf:about="#parteInterna">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Identificatore"/>
        <rdfs:range rdf:resource="#http://localhost/RDF/datatypes/#listaDiAttributi"/>
</rdf:Description>
<rdf:Description rdf:about="#parteEsterna">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#Identificatore"/>
        <rdfs:range rdf:resource="#http://localhost/RDF/datatypes/#listaDiAssociazioni"/>
</rdf:Description>
<rdf:Description rdf:about="#tramite">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#IdentificazioneEsterna"/>
        <rdfs:range rdf:resource="#Associazione"/>
</rdf:Description>
<rdf:Description rdf:about="#entitaEsterna">
        <rdf:type rdf:resource="http://www.w3.org/2001/01/rdf-schema#Property"/>
        <rdfs:domain rdf:resource="#IdentificazioneEsterna"/>
        <rdfs:range rdf:resource="#Entita"/>
</rdf:Description>
```

4.5 REALIZZAZIONE DELLA PAGINA WEB PER LA TRADUZIONE DELLO SCHEMA ER IN RDF

La pagina Web dinamica che traduce lo schema ER in RDF è strutturata in maniera molto semplice: durante il caricamento della pagina viene letta la variabile sessione che contiene gli oggetti che

rappresentano gli elementi dello schema disegnato, vengono tradotti in RDF/XML e il codice generato viene visualizzato in una TextBox multilinea.

La prima procedura che è stata scritta all'interno di questa pagina è una procedura che aggiunge un nuova linea alla TextBox con la stringa indicata come parametro e spostata verso destra di tanti tabulatori quanti indicati da un parametro di tipo intero. Il frammento di codice che definisce questa procedura viene riportato qui di seguito.

```
public void AggiungiRiga(int tab, string text)
{
        for(int i=0;i<tab;i++)
                txtRDF.Text= txtRDF.Text+"\t";
        txtRDF.Text= txtRDF.Text+text+"\n";
}
```

Questa procedura è alla base di tutta la pagina.

Un'altra caratteristica della pagina è quella di leggere un parametro dalla QueryString chiamato *esteso* che specifica se la traduzione RDF deve contenere anche le informazioni sul posizionamento degli elementi o meno. Il frammento di codice che riceve il parametro è il seguente:

```
        bool estesa;
        if (Request.QueryString.Get("estesa")=="true")
                estesa=true;
        else
                estesa=false;
```

Per tradurre lo schema ER sono state create tante procedure quanti sono i tipi di elementi di uno schema ER. Ogni procedura riceve un intero che definisce di quanti tabulatori deve essere spostato verso destra il codice, un oggetto che rappresenta l'elemento da tradurre e un booleano che rappresenta se devono essere riportati anche le informazioni di posizionamento dell'elemento o meno. Il codice, ad esempio, per tradurre un attributo composto è il seguente:

```
public void AttributoComposto(int tab, LibreriaDiClassi.AttributoComposto att, bool estesa)
{
        this.AggiungiRiga(tab,"<rdf:description rdf:about\"#"+att.entita + ":" + att.nome+"\">");
        tab=tab+1;
        this.AggiungiRiga(tab,"<rdf:type
                        rdf:resource=\"http://localhost/TESI/RDF/Vocabulary/#AttributoComposto\"/>");
        this.AggiungiRiga(tab,"<er:minimaCardinalita
                        rdf:datatype=\"http://localhost/TESI/RDF/Datatype/#minCard\">" + att.minCard +
                                                                "</er:MinimaCardinalita>");
        this.AggiungiRiga(tab,"<er:massimaCardinalita
                        rdf:datatype=\"http://localhost/TESI/RDF/Datatype/#maxCard\">" + att.maxCard +
                                                                "</er:MassimaCardinalita>");
        this.AggiungiRiga(tab,"<er:haAttributi>");
        if(estesa)
        {
                this.AggiungiRiga(tab,"<er:posizioneX>"+att.posX+"</er:PosizioneX>");
                this.AggiungiRiga(tab,"<er:posizioneY>"+att.posY+"</er:PosizioneY>");
                this.AggiungiRiga(tab,"<er:verso>"+att.verso+"</er:Verso>");
        }
        if (Session["Controls"]!=null)
```

```
        {
                Control[] cc = (Control[])Session["Controls"];
                foreach(Control c in cc)
                {
                        if (c.GetType().ToString().Equals("LibreriaDiClassi.Attributo") &&
                            ((LibreriaDiClassi.Attributo)c).tipocontenitore=="attributo composto" &&
                                ((LibreriaDiClassi.Attributo)c).contenitore==att.entita+":"+att.nome)
                                this.Attributo(tab+1,((LibreriaDiClassi.Attributo)c),estesa);
                }
        }
        this.AggiungiRiga(tab,"</er:haAttributi>");
        tab=tab-1;
        this.AggiungiRiga(tab,"</description>");
}
```

Questo codice rappresenta la tipica struttura delle procedure di traduzione degli elementi in RDF. Nella prima parte vengono tradotte le caratteristiche dell'elemento, nella seconda vengono tradotte, se richiesto, le caratteristiche di posizionamento dell'elemento e nella terza parte viene richiamata la procedura per la traduzione dei sottoelementi, in questo caso gli attributi dell'attributo composto.

5. CONCLUSIONI E LAVORO FUTURO

Il lavoro svolto ha portato alla realizzazione di una applicazione Web che permette di progettare database tramite il disegno di schemi ER all'interno di una pagina dinamica scritta con al tecnologia ASP.NET. L'applicazione permette di disegnare tutti i 9 elementi che compongono gli schemi (Entità, Associazioni, Attributi, Attributi composti, Partecipazioni, Subset, Generalizzazione, Identificazioni esterne e Identificatori).

Le parti dell'applicazione che possono essere migliorate relative all'espressione più generale possibile di schemi ER sono di tipo grafico: ad esempio non tutte le combinazioni di identificazioni esterne sono supportate e non vengono fatti tutti i controlli sul posizionamento degli elementi. L'applicazione non controlla, ad esempio, che due entità vengano posizionate parzialmente sovrapposte o se due entità che non si sovrappongono non sono abbastanza lontane da permettere di aggiungere degli attributi nello spazio che c'è tra di loro o a che distanza dall'entità creare un attributo composto perché non si sovrapponga a un altro elemento. Nonostante questi limiti grafici, l'applicazione permette di disegnare qualsiasi schema ER senza particolari problemi visto che per evitare i limiti grafici basta che un utente che ne tenga conto e posizioni gli elementi abbastanza distanti tra di loro. Proprio per questo, e per dare all'utente la possibilità di ottenere un miglior allineamento è stata inserita come sfondo una immagine che contiene una griglia.

L'applicazione permette inoltre la traduzione dello schema ER in RDF sia come semplice traduzione delle informazioni espresse dallo schema, sia aggiungendo alle informazioni tradotte le informazioni di posizionamento degli elementi. Questa ultima possibilità, in particolare, è stata inserita per poter, in un futuro, permettere a un utente di disegnare uno schema, salvarne le informazioni in formato RDF/XML e poter in un secondo momento ricaricare lo schema disegnato in precedenza.

La scelta di aggiungere le informazioni di posizionamento alla traduzione in RDF e salvare il codice RDF non era l'unica possibilità per permettere a un utente di salvare il lavoro effettuato e poterlo riprenderlo in seguito, era però l'unica soluzione che permette di sfruttare il lavoro già fatto per la traduzione dallo schema al codice RDF; è bastato aggiungere la traduzione delle informazioni di posizionamento. Inoltre questa soluzione permette di recuperare abbastanza agevolmente le informazioni sugli elementi, in quanto sono inserite in un codice RDF che è scritto in XML. Un altro vantaggio è che una volta recuperate le informazioni, basta ricreare i relativi oggetti, inserirli nella variabile sessione che contiene normalmente gli oggetti che rappresentano gli elementi e richiamare la pagina di disegno.

Un possibile sviluppo dell'applicazione è quello della cancellazione e della modifica dei componenti. Queste due operazioni potranno essere introdotte in modo semplice in quanto per effettuarle basta modificare o cancellare gli oggetti contenuti nella variabile sessione della pagina.

L'applicazione, oltre alle funzionalità già descritte, effettua numerosi controlli sui dati immessi dall'utente e sul posizionamento di alcuni componenti. Esempi di controlli sui dati immessi dall'utente vanno dal semplice controllo che impedisce di inserire due entità con lo stesso nome, al controllo che impedisce di creare una gerarchia di generalizzazione se in un'entità figlio esiste un attributo con lo stesso nome di un attributo in un'entità padre. Esempi di controlli sul posizionamento sono quelli che impediscono di posizionare due partecipazioni su uno stesso vertice di una associazione, quelli che impediscono di creare degli identificatori che contengono attributi multipli o attributi composti o quelli che impediscono di creare identificazioni esterne che non siano supportate dalla versione attuale dell'applicazione.

Sono lasciati per sviluppi futuri la scrittura di controlli più avanzati da effettuare al momento della traduzione dello schema come, ad esempio, la verifica della connessione dello schema.

6. BREVE GLOSSARIO

Temine	Definizione
.NET Framework	Il framework .NET è una piattaforma per la realizzazione della nuova generazione di applicazioni e servizi Web XML distribuiti. Espone un modello di programmazione coerente ed indipendente dal linguaggio e condiviso da tutti gli strati di un'applicazione e consente di interoperare in modo trasparente e di migrare facilmente a partire dalle tecnologie esistenti.
Alternative RDF	Una lista di risorse o letterali che rappresentano delle alternative al valore (singolo) di una proprietà. Il contenitore Alternative potrebbe essere impiegato, ad esempio, per indicare traduzioni in altre lingue del titolo di un'opera, o per fornire una lista di siti Internet mirror nei quali è reperibile la risorsa. Un'applicazione che usa una proprietà il cui valore è un raccolta di tipo Alternative sa di poter selezionare uno fra gli oggetti presenti nella lista
Anello	Un'anello è una associazione binaria tra un'entità e se stessa. Il ruolo di un'entità è indicato tramite una etichetta.
Arco RDF	Una rappresentazione di una proprietà in forma grafica.
ASP.NET	La tecnologia Active Server Pages per il framework .NET
Asserzione RDF	Una determinata risorsa, insieme ad una proprietà definita e al valore relativo alla proprietà stessa, costituiscono una asserzione RDF. Queste tre parti della asserzione sono dette, rispettivamente, il soggetto, il predicato, e l'oggetto. L'oggetto di una asserzione (ovvero il valore della proprietà) può essere a sua volta un'altra risorsa, oppure può essere un letterale: ovvero una risorsa (definita da un URI) o una semplice stringa di caratteri o altro tipo di dato primitivo definito da XML. In termini RDF un letterale può comprendere nel suo contenuto anche markup XML, ma esso non è elaborato da un processore RDF.
Associazione	Una associazione rappresenta un legame logico tra due o più entità.
Attributo	L'attributo rappresenta proprietà elementari di entità o associazioni
Attributo composto	Un attributo composto è costituito da un gruppo di attributi che hanno affinità nel significato o nell'uso.
Attributo RDF	Una caratteristica di un oggetto.

Bag RDF	Lista non ordinata di risorse o letterali. Si usano per dichiarare che una proprietà ha valori multipli e che l'ordine con cui questi valori sono inseriti non ha alcun significato. Il contenitore bag potrebbe essere usato per fornire una lista di numeri di sezioni dove l'ordine di elaborazione di queste sezioni non è importante. Sono ammessi valori duplicati.
Cardinalità	Cardinalità minima di C1 in A: min-card(C1,A): è il minimo numero di corrispondenze nell'aggregazione A alle quali ogni membro di C1 deve partecipare. Cardinalità massima di C1 in A: max-card(C1,A): è il massimo numero di corrispondenze nell'aggregazione A alle quali ogni membro di C1 può partecipare.
Classe RDF	Un concetto, una categoria o una classificazione generale. Spesso viene utilizzata per classificare cose. Formalmente, in RDF, è una risorsa che ha rdfs:Class come valore della proprietà rdf:type.
Copertura	Totale (t): ogni elemento della classe generica è in relazione con almeno un elemento delle classi generalizzate. Parziale (p): esistono alcuni elementi della classe generica che non sono in relazione con alcun elemento delle classi generalizzate. Esclusiva (e): ogni elemento della classe generica è in relazione con al massimo un elemento delle classi generalizzate. Sovrapposta (o): esistono alcuni elementi della classe generica che sono in relazione con elementi di due o più classi generalizzate.
Dominio di un attributo	è l'insieme di valori legali per l'attributo. Un attributo è detto semplice se è definito su un solo dominio.
Elemento	Questo termine si riferisce ad uno specifico costrutto sintattico XML; ovvero,il materiale tra i tag XML di apertura e di chiusura.
Entità	Una entità rappresenta un insieme di oggetti della realtà di cui si individuano proprietà comuni.
From Web	Le form Web sono una tecnologia ASP.NET utilizzabile per realizzare pagine Web programmabili. Le form Web possono visualizzare informazioni all'utente, utilizzando un qualsiasi linguaggio di marcatura, in qualsiasi browser ed utilizzare il codice lato server per implementare la logica applicativa
Garbage collection	Il processo che permette di tener traccia transitivamente di tutti i puntatori agli oggetti attualmente in uso per individuare tutti quelli referenziabili per riutilizzare tutte le porzioni di memoria del heap non individuate in questa fase.
Generalizzazione	Definisce una relazione di sovrainsieme tra una classe padre e altre classi figlie (sottoclassi).

Grafo RDF	è una rappresentazione di un insieme di triple RDF.
HTTP	Hyper Text Transfer Protocol, protocollo Internet standard per trasferire informazioni tra client e server e tra server e server.
Identificatore	Un identificatore di un'entità E è una collezione di attributo o di entità in associazione con E che individua in modo univoco le istanze di E.
Intermediate Language	L'intermediate Language (IL) è un linguaggio utilizzato come output per numerosi compilatori e come input per un compilatore JIT. L'IL definisce un'architettura di esecuzione astratta e basata sullo stack.
Nodo RDF	Una rappresentazione di una risorsa o un letterale in forma grafica.
Oggetto RDF	Identifica il valore della proprietà nella dichiarazione modellata.
Predicato RDF	Identifica la proprietà originale nella dichiarazione modellata. Il valore del predicato è una risorsa che rappresenta la specifica proprietà nella dichiarazione.
Proprietà RDF	Una proprietà consiste in un aspetto specifico, una caratteristica, un attributo, o una relazione usata per descrivere una risorsa. Ogni proprietà ha un significato specifico, definisce i valori ammessi, i tipi di risorse a cui può riferirsi, e la sua relazione con altre proprietà.
Risorsa RDF	Tutte le cose descritte con espressioni RDF vengono dette risorse. Una risorsa può essere un'intera pagina Web; ad esempio il documento "http://www.w3.org/Overview.html". Una risorsa può anche essere una parte di una pagina Web; ad esempio uno specifico elemento HTML o XML all'interno di un documento. Una risorsa può anche essere un'intera collezione di pagine; per esempio un intero sito Web. Una risorsa può anche essere un oggetto non direttamente accessibile via Web; ad esempio un libro stampato. Le risorse sono sempre definite da URI con eventuali anchor id. Qualsiasi cosa può avere associato un URI; l'estensibilità degli URI permette l'introduzione di identificatori per qualsiasi entità immaginabile.

Schema RDF	Uno schema rappresenta la sede in cui vengono documentate le definizioni e i vincoli di uso delle proprietà. Allo scopo di evitare confusioni tra definizioni dello stesso termine provenienti da fonte diversa e potenzialmente contrastanti, RDF adopera le funzionalità dei Namespace XML. I Namespace sono semplicemente un modo per associare un uso specifico di una parola contestualmente al dizionario schema) in cui si intende la definizione debba essere trovata. In RDF, ogni predicato usato in una asserzione deve essere identificato esattamente con un namespace, o schema. In ogni caso, un elemento Description può contenere asserzioni con predicati provenienti da diversi schemi.
Sequence RDF	Lista ordinata di risorse o letterali. Il contenitore Sequence si usa per dichiarare che una proprietà ha valori multipli e che l'ordine dei valori è significativo. Il contenitore Sequence potrebbe essere usato, per esempio, per conservare un ordine alfabetico dei valori. Sono ammessi valori duplicati.
Soggetto RDF	Identifica la risorsa che viene descritta nella dichiarazione, quindi il soggetto è la risorsa relativamente alla quale era stato formulata la dichiarazione originale
Subset	Un subset è una gerarchia di generalizzazione con una sola entità generalizzata. La copertura di un subset è parziale.
Terna RDF	Una rappresentazione di una asserzione usata da RDF, composta dalla proprietà, dall'identificatore della risorsa e dal valore della proprietà in questo ordine.
XML	Extensible Markup Language, uno standard WC3 per la formattazione di documento e dati strutturati sul web.

7. BIBLIOGRAFIA RAGIONATA

RDF (RESOURCE DESCRIPTION FRAMEWORK)

Tutto il materiale riguardante RDF è stato tratto dal sito della W3C (http://www.w3.org), in particolare le pagine che sono state particolarmente utili sono:

[1] RDF Primer (http://www.w3.org/TR/2004/REC-rdf-primer-20040210/) che riassume tutto quello che c'è da sapere su RDF. Ho usato questa pagina come guida per come strutturare il discorso.

[2] RDF Semantics (http://www.w3.org/TR/2004/REC-rdf-mt-20040210/) che precisa la semantica e il corrispondente sistema di regole per RDF e RDF Schema.

[3] RDF Vocabulary Description Language 1.0: RDF Schema (http://www.w3.org/TR/2004/REC-rdf-schema-20040210/) per la parte che riguarda i vocabolari RDF e le classi e le proprietà predefinite da RDF Schema.

[4] RDF/XML Syntax specification (http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/) per la parte che riguarda le regole di traduzione dei grafi RDF e delle terne in codice RDF/XML.

[5] RDF Concepts and abstract syntax (http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/) che definisce i la sintassi astratta su cui si basa RDF e come collegare la sintassi concreta di una dichiarazione alla semantica formale di RDF. Questo documento è stato usato in particolare per la parte di descrizione dei grafi RDF e delle dichiarazioni con il metodo delle terne.

ASP.NET

Per la parte di ASP.NET è stato utilizzato il seguente testo:

[6] "Practical ASP.NET"
Steven Smith
JACKSON

Sono state utilizzate le seguenti due pagine tratte dal sito della Microsoft su come ASP.NET gestisce round trip e gestione dello stato:

[7] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconwebformspageprocessingpage.asp per la parte che riguarda le modalità di elaborazione delle pagine ASP .NET e i round trip.

[8] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconintroductiontowebformsstatemanagement.asp per la parte che riguarda la gestione dello stato delle pagine e i metodi per mantenerlo tra un round trip e il successivo.