Contributi ISTAT

Uso di XML e WEB Services per l'integrazione di sistemi informativi statistici attraverso lo standard SDMX

F. Rizzo, D. Camol, L. Vignola (*)

(*) ISTAT – Direzione Centrale per la Diffusione della cultura e dell'informazione statistica

1. Introduzione 4
2. Integrazione come necessità dei moderni sistemi informativi 5
3. La ricerca 9
3.1 Studio delle tecnologie a supporto 9
3.1.1 L'eXstensible Markup Language (XML) 10
3.1.1.1 Composizione di un documento XML. 11
3.1.2 La famiglia di tecnologie XML 13
3.1.2.1 XML Namespace 13
3.1.2.2 XML Schema Definition Language 14
3.1.2.3 XML Path (XPath) 15
3.1.2.4 XML Stylesheet Language – Transformation (XSLT) 15
3.1.2 I WEB Services 16
3.1.2.1 Simple Object Access Protocol (SOAP) 17
3.1.2.2 WEB Services Description Language (WSDL) 18
3.1.2.3 Universal Discovery and Integration (UDDI) 18
3.1.3 Standard for Data and Metadata Exchange (SDMX) 19
3.1.3.1 Gli obiettivi di SDMX 19
3.1.3.2 Forme di scambio e modalità di raccolta e accesso ai dati 20
3.1.3.3 Concetti base di SDMX versione 1.0 20
3.1.3.4 Formati supportati dalla versione 1.0 dell'SDMX 21
3.2 Analisi del sistema prototipale per l'integrazione delle banche dati di diffusione 22
3.2.1 Analisi dell'esistente 23
3.2.2 Analisi del prototipo 28
3.2.2.1 Integrazione in architettura data exchange 28
3.2.2.2 Integrazione in architettura data sharing 30
3.3 Progettazione di massima del prototipo 32
3.3.1 Interfaccia utente di ricerca dei dati 34
3.3.1.1 Dal sito WEB al Repository 36
3.3.1.2 Query-form per i dati con dimensione dominante temporale 40
3.3.1.3 Query-form per i dati con dimensione dominante territoriale 42
3.3.2 Database del Repository e logica d'interrogazione 43
3.2.2.1 Area del database in cui viene memorizzata la struttura dell'albero di ricerca 44
3.2.2.2 L'area del database in cui è stato registrato il Satellite Conistat 46
3.2.2.3 La logica d'interrogazione del Satellite Conistat 47
3.2.2.4 La struttura del database in cui è stato registrato il Satellite Demo 48
3.2.2.5 La logica d'interrogazione del Satellite Demo 49
3.3.3 Il documento sdmxQuery e il client SOAP 50
3.3.4 L'interfaccia di presentazione dei dati all'utente 51
3.3.5 I Satelliti 53
3.3.5.1 Il Satellite Conistat 54
3.3.5.2 Il Satellite Demo 54
3.4 Realizzazione del prototipo 56
3.4.1 Classi generalizzate 59
3.4.2 Funzionamento 62
3.4.2.1 Utente-Registry 62
3.4.2.2 Registry-Satellite 65
3.4.2.3 Satellite 65
3.4.2.4 Satellite-Registry 67
APPENDICE A 69

APPENDICE B.....70
APPENDICE C.....71

La stesura dei capitoli e dei paragrafi è da attribuire come segue:

F.Rizzo	Cap. 1, Cap. 2, Cap. 3
L.Vignola	Par. 3.1.3 e sottoparagrafi
D.Camol	Par. 3.4 e sottoparagrafi, Appendice B,C

1. Introduzione

Il sistema informativo dell'Istat negli ultimi anni ha subito diverse evoluzioni adeguando ai necessari cambiamenti tecnologici tutte le varie fasi del processo di produzione del dato statistico. In particolare a metà degli anni '90 è stata avviata la migrazione da un'architettura centralizzata, basata su mainframe, ad un'architettura distribuita basata sul modello client-server. In una prima fase il mainframe è stato sostituito da più minicomputer con sistema operativo Unix (in particolare la versione AIX-IBM) ai quali sono stati collegati terminali grafici (x-terminal); in una seconda fase tali minicomputer hanno assunto la funzione di server di data base e server di file system ai quali è stato possibile accedere da personal computer in modalità client. Il sistema informativo è stato quindi scomposto in una miriade di sistemi informativi dipartimentali identificabili, spesso, con il singolo Servizio di produzione. Tale cambiamento si è reso necessario non solo per rispondere più efficacemente alle crescenti esigenze informative dell'utenza, sia interna che esterna, ma anche per meglio adattarsi alle specificità di ciascun Servizio di produzione. Come logica conseguenza, anche nella parte del processo produttivo identificabile con la diffusione dei dati verso l'esterno, sono nate diverse banche dati, facenti capo a singoli settori dell'Istituto che le hanno progettate e realizzate e adesso provvedono alla loro gestione e manutenzione. Queste banche dati, se da un lato hanno permesso di diffondere in tempi brevi, generalmente su WEB, i dati prodotti, dall'altro hanno creato disomogeneità nelle modalità di accesso: gli utenti finali devono sapere in quale di loro le informazioni di interesse sono state memorizzate e come accedere a ciascuna.

Già da tempo all'interno dell'Istituto si sta discutendo su come rendere uniforme l'accesso ai dati. A evidenziare tale esigenza ha contribuito, di recente, la riorganizzazione del sito WEB istituzionale dell'Istat che di fatto propone una logica di accesso a portale, cioè un unico punto da cui l'utente possa accedere ai dati senza sapere la loro collocazione ed utilizzando sempre la stessa logica.

Inoltre, l'utenza attuale, esigente ed attenta ha contribuito affinché le politiche di diffusione andassero sempre più verso una semplificazione dell'accesso alle informazioni. Questo sta avvenendo non solo a livello dei singoli Istituti nazionali ma anche a livello degli organismi internazionali. A tale proposito varie organizzazioni, tra cui Eurostat, stanno cooperando nella definizione di una serie di standard in grado di ottimizzare lo scambio e la condivisione di dati e metadati. Questa cooperazione ha dato luogo al progetto SDMX (Standard for Data and Metadata Exchange).

Nell'ambito della Direzione Centrale per la Diffusione della cultura e dell'informazione statistica è nata, quindi, l'idea di realizzare una ricerca sull'uso di nuove tecnologie quali XML e WEB services per:

a) analizzare la loro corretta ed efficace applicazione nell'ambito dei sistemi statistici di diffusione;
b) considerare la loro applicabilità nell'ambito dell'integrazione di differenti sistemi statistici di diffusione;
c) studiare lo standard SDMX, verificando il suo utilizzo nello scambio e nella condivisione di dati e metadati di diffusione.

2. Integrazione come necessità dei moderni sistemi informativi

Nell'ultimo decennio molte organizzazioni hanno fatto consistenti investimenti nell'ambito dell'information technology guidati da due impellenti necessità:

1. migrare e/o completare le migrazioni da sistemi centralizzati, basate su mainframe o mini computer, a sistemi distribuiti;
2. confrontarsi con tutto ciò che si riferisce alle tecnologie legate ad Internet, in particolare WEB ed E-mail.

La prima necessità evidenziata ha sottinteso la crescita di una infrastruttura di rete e di elaborazione con lo scopo principale di facilitare lo scambio e la comunicazione di informazioni e dati. La conseguenza logica è stata la proliferazione, non solo di sistemi a tecnologia differente, ma anche di programmi software realizzati con logiche applicative differenti.

La seconda ha cambiato il modo di realizzare le applicazioni. Essendo le tecnologie applicative nate per Internet basate essenzialmente su standard aperti, risultano meno costose e facili da implementare.

Le architetture distribuite hanno comportato come immediata conseguenza disomogeneità sia a livello sistemistico che applicativo. La disomogeneità sistemistica, anche se non facilmente gestibile risulta almeno controllabile a livello di costi; quella a livello applicativo risulta molto più costosa e meno gestibile.

Se dal lato sistemi un'organizzazione può decidere se far propria la tesi di abbracciare una particolare piattaforma piuttosto che controllare l'inserimento di differenti tipi di piattaforme che vadano meglio a soddisfare le necessità del "business", dal lato applicativo risulta più complesso accedere alle informazioni così come sono memorizzate sui vari sistemi e renderle condivise con gli applicativi esistenti all'interno dell'organizzazione.

La condivisione delle informazioni all'interno dei sistemi informativi è divenuta, dunque, una delle esigenze maggiormente sentite nelle varie organizzazioni. Se, da un lato le nuove tecnologie, attraverso la distribuzione delle funzionalità elaborative, hanno facilitato lo sviluppo di applicazioni sempre più sofisticate e vicine alle esigenze dell'utenza, dall'altro questo ha comportato una dispersione dell'informazione sia in termini di memorizzazione che di gestione.

Si arriva quindi al paradosso di percepire la differenza tra cosa l'information technology è capace di fare e quello che allo stato attuale si riesce ad ottenere. Questa è stata la prima ragione per cui i progetti di integrazione sono in cima alla lista delle priorità di molte organizzazioni.

Nel corso degli ultimi anni, i requisiti e le modalità di lavoro imposti dalla maggior parte delle attività lavorative sono profondamente cambiati. Spesso senza rendercene conto si è passati dalla necessità di utilizzare un unico contenitore dove memorizzare tutti i dati di cui si ha bisogno all'interno di una organizzazione ad una miriade di contenitori. Infatti, attualmente, i dati vengono memorizzati in dbms distribuiti, in server di posta elettronica, in servizi di directory, in documenti Office, ecc.

Inoltre, spostando la nostra attenzione a un livello superiore, ovvero considerando un ambiente di elaborazione distribuito, ci rendiamo conto che un unico server localizzato in un'apposita sala macchina può essere, in realtà, solo una parte dell'intera infrastruttura informatica dell'organizzazione. Di conseguenza, le tecnologie di accesso ai dati devono essere in grado di lavorare con tipi differenti di deposito di dati.

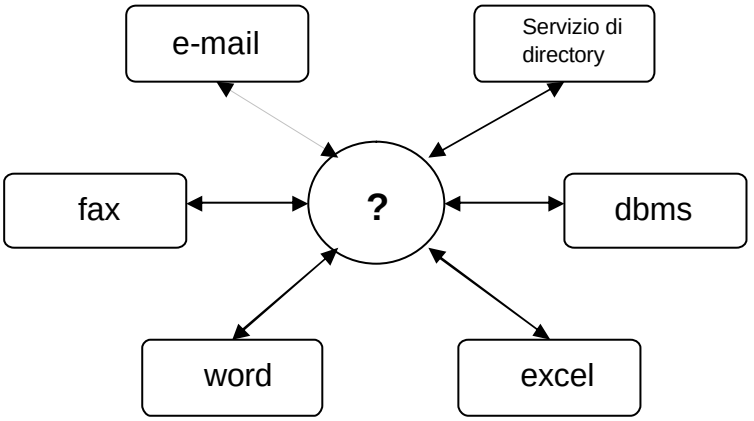


figura 2.1

Quindi, oramai da diversi anni, uno degli obiettivi principali nell'ambito dei sistemi informativi complessi è quello di integrare tutti i sottosistemi presenti al fine di:

- non duplicare i dati e le procedure;
- ridurre i rischi di trattare informazioni non corrette;
- sintetizzare informazioni provenienti da sistemi elaborativi comunque dislocati all'interno dell'organizzazione al fine di avere velocemente disponibili i dati necessari per prendere decisioni nel minor tempo possibile.

Negli anni quasi tutte le aziende di soluzioni informatiche hanno proposto metodologie e tecniche di integrazioni. Tali soluzioni sono risultate fortemente proprietarie, cioè troppo legate alle aziende che le hanno realizzate e troppo spesso difficili da applicare.

Con l'avvento di Internet e l'esplosione dell'informazione e dell'informatica "condivisa", molte tecniche e tecnologie sono state standardizzate.

Si è passati dalla semplice condivisione dell'infrastruttura (la rete), alla condivisione dell'informazione attraverso la facilità di accesso (WEB). È stato proprio l'accettazione del WEB su scala mondiale che ha innescato processi di standardizzazione che mai prima erano stati accettati sia dai produttori che dai consumatori.

Basta pensare a tutti gli anni in cui l'ISO[1] ha cercato di dettare degli standard sia a livello di protocolli di trasporto (TP4) che a livello applicativo (FTAM), mentre sul mercato la potenza di marketing delle società produttrici faceva diventare standard *de facto* i loro prodotti.

Con la nascita e lo sviluppo di Internet le cose sono cambiate: diversi standard legati a questa rete sono emersi e sono stati accettati dalla comunità. In particolare il WEB è stato il primo vero standard che è stato accettato sia dai produttori di software che dagli utenti finali con lo scopo di richiedere e ricevere informazioni sulla rete Internet. In seguito tale standard è stato adottato a livello delle singole organizzazioni per distribuire le informazioni al loro interno (Intranet) o tra organizzazioni (Extranet).

L'evolversi delle tecnologie e delle aspettative degli utenti ha però evidenziato il limite legato al protocollo che più ha reso popolare il WEB, cioè l'HTML.

L'HTML è nato come linguaggio per descrivere e rendere accessibili documenti pubblicati su Internet in forma di ipertesti ma nel tempo si è cercato di affidargli anche altre mansioni che hanno presto dimostrato i propri limiti. Per superare tali limiti il World Wide WEB Consortium (W3C) che è l'organismo indipendente che realizza e promuove gli standard utilizzabili su Internet, ha presentato un insieme di tecnologie che vanno sotto il nome di XML (eXtensible Markup Language).

Nel momento in cui si ha la necessità di trattare, memorizzare e trasferire dati, XML offre due grossi vantaggi:

a) è stato accettato come standard a livello industriale;
b) è costituito da puro testo.

Il primo vantaggio è dato dal fatto che il metodo di trasferimento e di presentazione delle informazioni utilizza un formato assolutamente indipendente dalla piattaforma, dal sistema operativo e dall'applicazione. Il secondo vantaggio è che non sarà più necessario preoccuparsi del

[1] ISO – International Standard Organization

metodo di memorizzazione e trasporto dei dati stessi. Infatti un file di testo può essere inviato attraverso Internet utilizzando il protocollo http (Hypertext Transport Protocol) e quindi essere trasparente ai firewall che generalmente presentano una porta aperta per questo protocollo. Inoltre un file di testo può essere scritto senza problemi su disco, oppure memorizzato in un data base all'interno di un campo testo.

È ovvio che questo modo di procedere genera un file più grosso rispetto alla sua rappresentazione binaria, ma le tecniche di compressione e la disponibilità di larghezza di banda sempre più ampia e meno costosa sono in grado di compensare tale aspetto.

3. La ricerca

La necessità di avere un unico accesso alle banche dati, attualmente disponibili su WEB, è divenuta un punto chiave per la diffusione dei dati statistici. Tale necessità ben si colloca nella logica di portale già tracciata nel nuovo sito dell'Istituto. Anche a livello europeo, Eurostat sta stimolando gli stati membri attraverso la realizzazione di vari progetti. In particolare oltre alla partecipazione diretta nel progetto SDMX, ha avviato l'iniziativa SODI (SDMX Open Data Interchange) che ha come scopo principale quello di rendere comparabili i dati dei vari paesi attraverso una presentazione uniforme e tempestiva: tutte caratteristiche necessarie per soddisfare gli operatori, istituzionali e non, che dalle informazioni statistiche fanno lo strumento più adeguato per la pianificazione e il controllo. Già da tempo nel nostro Istituto alcune iniziative stanno affrontando il tema dell'integrazioni dei sistemi in tutte le fasi di produzione dei dati. A tale proposito nella Direzione Centrale per la Diffusione della cultura e dell'informazione Statistica (DCDS) è stata realizzata una ricerca che ha avuto come scopo principale quello di sperimentare l'integrazione delle attuali banche dati di diffusione esistenti.

In particolare la ricerca si è occupata di:

- analizzare l'uso di nuove tecnologie quali XML e WEB Services e la loro corretta ed efficace applicazione nell'ambito dei sistemi statistici di diffusione;
- considerare la loro applicabilità nell'ambito dell'integrazione di differenti sistemi statistici di diffusione;
- studiare lo standard SDMX verificando il suo utilizzo nello scambio e la condivisione di dati e metadati di diffusione.

La ricerca è stata suddivisa in tre momenti ben delineati anche se fortemente relazionati:

- studio delle tecnologie a supporto
- analisi e progettazione del sistema
- realizzazione di un prototipo

3.1 Studio delle tecnologie a supporto

Questo momento è stato fondamentale per capire il reale utilizzo delle tecnologie coinvolte, verificare la loro efficacia nell'ambito del sistema trattato, definire alcune linee guida per il loro corretto utilizzo. I paragrafi seguenti hanno come scopo principale quello di esporre, in forma organica, gli argomenti studiati, in maniera tale che il lettore possa utilizzarli come base da cui partire per un ulteriore approfondimento. In appendice A vengono forniti gli indirizzi dei siti WEB nei quali il lettore potrà trovare la documentazione specialistica.

3.1.1 L'eXtensible Markup Language (XML)

L'XML è stato sviluppato originariamente come linguaggio per definire un nuovo formato per i documenti WEB. Deriva dall'SGML (Standard Generalized Markup Language), cioè lo standard internazionale per la descrizione della struttura e del contenuto di documenti elettronici di qualsiasi tipo. Sia SGML che XML hanno un formato "test-based" e possono essere considerati dei meta linguaggi, cioè dei linguaggi per descrivere altri linguaggi. XML risulta notevolmente semplificato rispetto all'SGML ed entrambi descrivono la struttura di un documento attraverso dei marcatori o tag[2]. Con il tempo l'XML si è evoluto da semplice descrittore di documenti su WEB a descrittore di dati strutturati. Esempi tipici di dati strutturati sono quelli contenuti nei data base, nei fogli elettronici, nei protocolli di rete, ecc.

XML è risultato molto più flessibile rispetto ai formati precedentemente utilizzati per descrivere dati strutturati perché può rappresentare con la stessa semplicità sia dati in formato tabellare (fogli elettronici e data base), sia dati in formato semi strutturati (documenti). Per esempio consideriamo alcuni formati divenuti nel tempo degli standard *de facto*: CSV (Comma Separated Value) e RTF (Rich Text File). Il primo ben rappresenta dati tabellari ma non semi strutturati, mentre il secondo fa l'opposto. Questa caratteristica ha permesso all'XML di divenire la *lingua franca* nello scambio delle informazioni.

Essendo l'XML un meta linguaggio, ha come caratteristica molto importante quella di essere estensibile. Questo concetto si manifesta in diversi modi. Prima di tutto, a differenza di HTML[3], XML non ha un vocabolario fisso: chiunque può definire vocabolari specifici per applicazioni varie. In secondo luogo applicazioni che creano o consumano file XML risultano essere più "resistenti" rispetto a quando utilizzano altri formati, nel momento in cui avvengono dei cambiamenti nei dati. Per esempio consideriamo un'applicazione che considera un elemento <Impresa> che ha come attributo il *codice ditta.* Nel caso in cui venga aggiunto, in seguito, allo stesso elemento un ulteriore attributo, per esempio l'*unità locale*, questo non andrà ad impattare sull'applicativo[4]. Questa flessibilità non è comune in altri formati di dati, mentre risulta essere un significativo beneficio nel momento in cui si usa XML.

Essendo XML un linguaggio per definire altri linguaggi, nel corso del tempo in vari settori della scienza e della tecnica sono nate delle vere e proprie *grammatiche* di settore. Alcuni esempio sono:

[2] Per marcatore si intende una parola racchiusa tra il carattere < e il carattere >.

[3] HTML (Hypertest Markup Language) deriva anch'esso da SGML. Risulta però composto da un certo numero di tag definiti a priori. Inoltre è stato pensato per descrivere principalmente come i dati dovranno essere presentati in un browser WEB, piuttosto che dare un significato semantico degli stessi.

[4] Nel caso si stesse utilizzando il formato CSV, tale operazione comporterebbe un adeguamento delle procedure al nuovo tracciato record dopo l'inserimento dell'attributo "unità locale".

- XBRL (eXtensible Business Exchange Markup Language) – per lo scambio di report e dati finanziari;
- UBL (Universal Business Language) – nell'ambito dei documenti che trattano argomenti economici;
- AML (Astronomical Markup Language) – nell'ambito della condivisione di dati astronomici;
- CML (Chemical Markup Language) – nell'ambito della condivisione dei dati chimici;
- ICE (Information and Content Exchange) – standard per le agenzie di stampa;
- cXML (Commercial XML) – nell'ambito del commercio elettronico.

Anche nell'ambito statistico è presente il progetto SDMX (Standard for Data and Metadata eXchange) la cui versione 1.0 è stata standardizzata da parte dell'ISO.

3.1.1.1 Composizione di un documento XML.

Come già evidenziato, XML possiede come caratteristica fondamentale la capacità di fornire una struttura ai documenti e di rendere i dati autodescrittivi. Quindi non definisce i marcatori, ma fornisce le regole per poterli definire. Questa libertà di definizione potrebbe generare anarchia se non venisse in qualche modo controllata attraverso il rispetto delle regole strutturali e grammaticali. A questo scopo lo standard prevede due principi fondamentali:

- i documenti XML devono essere ben formattati (*well-formed*);
- i documenti XML devono essere validi.

I documenti ben formattati devono rispettare determinate regole ortografiche per la definizione della struttura del documento e regole per l'uso corretto dei marcatori:

- ogni documento deve contenere un elemento di massimo livello (root) che contenga tutti gli altri elementi del documento. Le sole parti che rimangono fuori del livello di root sono i commenti e le direttive di elaborazione;
- ogni elemento deve avere sia un marcatore di apertura che uno di chiusura. Nel caso di elementi vuoti, è possibile utilizzare un unico marcatore nella forma abbreviata (< …. />);
- gli elementi devono essere correttamente nidificati, cioè i marcatori di chiusura devono seguire l'ordine inverso dei rispettivi marcatori di apertura;
- XML è sensibile alle maiuscole (case sensitive);
- i valori degli attributi devono sempre essere racchiusi tra singoli o doppi apici;

Le regole well-formed non pongono nessun vincolo sui marcatori che possono essere utilizzati in un determinato documento. In altre parole abbiamo bisogno di definire una grammatica per il linguaggio basato sui marcatori che il progettista inventa per un dato documento. Tale grammatica

dovrà indicare quali sono i vocaboli che possiamo utilizzare (marcatori) e con che struttura possiamo comporre le nostre frasi.

Un documento XML è formato da componenti denominati *elementi.* Ciascun elemento rappresenta un componente logico del documento e può contenere altri elementi (*sottoelementi*) o del testo.

Gli elementi possono avere associate altre informazioni che ne descrivono le proprietà. Queste informazioni sono chiamate *attributi.* L'organizzazione degli elementi segue un ordine gerarchico che prevede un elemento principale chiamato *root element* o semplicemente *root*, che contiene tutti gli elementi di un documento.

In genere la struttura di un documento XML può essere rappresentata graficamente tramite un albero noto come *document tree*.

Segue un esempio di documento XML che rappresenta una tabella statistica:

```
<?xml version="l.O" encoding="utf-8" ?>
<!--Rappresentazione XML di una tabella Statistica -->
<tabella Titolo="Indice del Fatturato">
    <nota>
      Per gli aspetti metodologici si veda la nota informativa del 28 marzo
      2003
    </nota>
    <pagina numero="l">
        <colonna titolo=="Totale generale">
            <osservazione anno="1996" periodo="01" valore="76,1" />
            <osservazione anno="1996" periodo="02" valore="83,9" />
            <osservazione anno="1996" periodo="03" valore="92,4" />
        </colonna>
        <colonna titolo="Estrazione di minerali">
            <osservazione anno="1996" periodo="01" valore="45,7" />
            <osservazione anno="1996" periodo="02" valore="42,8" />
            <osservazione anno="1996" periodo="03" valore="73,2" />
        </colonna>
        <colonna titolo="Attività manifatturiera">
            <osservazione anno="1996" periodo="01" valore="12,1" />
            <osservazione anno="1996" periodo="02" valore="25,9" />
            <osservazione anno="1996" periodo="03" valore="31,1" />
        </colonna>
    </pagina>
    <pagina numero="2">
        <colonna titolo="Alimentari">
            <osservazione anno="1996" periodo="01" valore="89,1" />
            <osservazione anno="1996" periodo="02" valore="81,7" />
            <osservazione anno="1996" periodo="03" valore="85,9" />
        </colonna>
        <colonna titolo="Tabacchi">
            <osservazione anno="1996" periodo="01" valore="89,1" />
            <osservazione anno="1996" periodo="02" valore="73,9" />
            <osservazione anno="1996" periodo="03" valore="75,1" />
        </colonna>
    </pagina>
</tabella>
```

3.1.2 La famiglia di tecnologie XML

XML rappresenta la tecnologia di base di una famiglia di tecnologie i cui componenti principali[5] sono:

- XML Namespace
- XML Schema Definition Language
- XML Path (XPath)
- XML Stylesheet Language – Transformation (XSLT)

In seguito tali tecnologie verranno descritte sommariamente, lasciando al lettore gli opportuni approfondimenti attraverso la bibliografia ed i link presenti in appendice A.

3.1.2.1 XML Namespace

I namespace XML offrono un metodo per riutilizzare le marcature nei documenti. In pratica i namespace hanno lo scopo di rendere i nomi degli elementi e degli attributi unici.

Il concetto di namespace è presente in quasi tutti i linguaggi di programmazione moderni. Volendo fare un esempio semplice da intendere, un namespace può essere visto come l'organizzazione delle directory e dei file nel filesystem. Ciascuna directory contiene al suo interno dei nomi di file o sottodirectory che risultano univoci. Quindi se esistono due file entrambi nominati "file1" ma posizionati in due directory differenti, per esempio dir1 e dir2, allora potranno essere identificati univocamente specificato il percorso di ciascuno di essi: dir1\file1 e dir2\file1. Come nell'organizzazione delle directory e dei file, i namespace possono essere organizzati su più livelli.

La denominazione dei namespace segue le regole definite nella RFC 2396[6], cioè le regole da seguire nel caso di un URI[7] (Uniform Resource Identifier). In accordo con tali specifiche, anche i namespace possono seguire la forma tipica degli URL (Uniform Resource Locators) o quella degli URN (Uniform Resourcers Names).

Seguono due esempi di namespace:

http://www.istat.it/sdmx	tipo URL
urn:www.istat.it:sdmx	tipo URN

I namespace vengono di solito utilizzati per descrivere gli elementi e gli attributi per un documento XML. Per esempio il namespace *http://www.w3.org/1999/XML/Transform* include i marcatori

[5] Altre tecnologie che fanno parte della famiglia non vengono citate nel documento in quanto non necessarie allo scopo del progetto.

[6] Le RFC (Request For Comment) sono documenti che descrivono i protocolli utilizzabili su Internet

[7] URI: metodo comune di identificazione dei servizi presenti in rete, ideato per unificare la sintassi necessaria per riferirsi a risorse di tipo diverso, comprende URL e URN.

utilizzabili per fare trasformazioni, mentre il namespace *http://www.w3.org/2000/10/XMLSchema* include marcatori per creare schemi XML.

Per utilizzare dei marcatori definiti da altri, è necessario importare i namespace di riferimento nei propri documenti. In generale un namespace è correlato ad un DTD[8] (Document Type Definition) o ad un XMLSchema. Tale importazione avviene attraverso l'attributo XML chiamato "xmlns". Tale speciale attributo che è parte integrante delle specifiche XML è disponibile in ogni elemento.

Segue un esempio di utilizzo di namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<CompactData xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v1_0/message"
             xmlns:istat_sts="http://www.istat.it/key_families/compact"
             xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
   <istat:DataSet>
      <istat:SiblingGroup REF_AREA="BE" ADJUSTMENT="W" BASE_YEAR="2000"
            UNIT_MULT="0" DECIMALS="2" TITLE="Indice grezzo della produzione"/>
         <istat:Series FREQ="M" ADJUSTMENT="R" TIME_FORMAT="P1M" COLLECTION="A"
               AVAILABILITY="A">
            <istat:Obs TIME_PERIOD="2004-10" VALUE="113.30" STATUS="A"/>
            <istat:Obs TIME_PERIOD="2004-09" VALUE="111.75" STATUS="A"/>
            <istat:Obs TIME_PERIOD="2004-08" VALUE="97.06" STATUS="A"/>
         </istat:Series>
   </istat:DataSet>
</CompactData>
```

3.1.2.2 XML Schema Definition Language

Come detto nel paragrafo 3.1.1.1, un documento XML deve essere validato. In genere un documento viene validato nel momento in cui risulta coerente con le regole definite in un DTD. Tali regole possono far parte del documento XML stesso o scritte in un altro documento. I DTD presentano però diverse limitazioni. Inoltre per scrivere le regole è necessario utilizzare una sintassi particolare, molto differente di quella utilizzata nei documenti XML. Il W3C ha pensato quindi di introdurre gli XMLSchema che hanno le stesse funzionalità dei DTD ma, essendo scritti in XML, hanno il notevole vantaggio di essere estensibili.

Segue un esempio di XMLSchema che definisce la testata di un documento SDMX:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace=http://www.SDMX.org/resources/schemas/SDMXML
         xmlns=http://www.SDMX.org/resources/SDMXML/schemas/draft/SDMXML
         xmlns:xsd=http://www.w3.org/2001/XMLSchema
                                    elementFormDefault="qualified">
      <xsd:element name="SDMXTimeSeries">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element name="Header" type="HeaderType"/>
                        <xsd:element name="StructureDefinition"
                                    type="StructureDefinitionType"/>
                        <xsd:element name="Data" type="DataType" minOccurs="0"/>
                  </xsd:sequence>
```

[8] Documento che specifica le componenti ammesse alla costruzione di un file XML

```
            </xsd:complexType>
      </xsd:element>
      <!-- Header Declarations -->
      <xsd:complexType name="HeaderType">
            <xsd:sequence>
                  <xsd:element ref="Name"/>
                  <xsd:element ref="Sender"/>
                  <xsd:element ref="Receiver"/>
                  <xsd:element ref="TestIndicator"/>
                  <xsd:element ref="Prepared"/>
            </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="Receiver" type="xsd:string"/>
      <xsd:element name="Sender" type="xsd:string"/>
      <xsd:element name="TestIndicator" type="xsd:boolean"/>
      <xsd:element name="Prepared" type="xsd:string"/>
</xsd:schema>
```

3.1.2.3 XML Path (XPath)

XPath è una tecnologia usata per identificare parti di documenti XML, ed è stato progettato per essere utilizzato da tecnologie quali XSL, XSLT, XPoint[9]. Adotta una sintassi compatta, non XML, che modella un documento XML come un albero di nodi e naviga attraverso la struttura logica astratta del documento stesso usando come punto di partenza un nodo arbitrario.

Segue un esempio di istruzioni XPath, utilizzata nel modulo software descritto nel paragrafo 3.3.4, che permette di scandire tutti i nodi “Series” del documento e compiere, quindi, determinate azioni.

```
<xsl:for-each select="//CompactData/DataSet/SiblingGroup/Series">
      . . . .
      . . . .
</xsl:for-each>
```

3.1.2.4 XML Stylesheet Language – Transformation (XSLT)

Uno degli aspetti più interessanti di XML è quello di costruire applicazioni che si basano sul principio chiave della *separazione tra contenuto e forma*. Mediante le trasformazioni di stile è possibile presentare l’informazione in modo adeguato alle caratteristiche dell’utilizzatore.

Durante il processo elaborativo, dall’origine alla fase di presentazione (rendering), è conveniente legare l’informazione alla sua forma finale il più tardi possibile, mantenendo così la possibilità di utilizzarlae anche per altri obiettivi durante l’intero processo.

In generale, nella restituzione di un documento XML si hanno due passi distinti di trasformazione di stile:

[9] XPoint è una tecnologia che fa parte della famiglia XML e che fornisce i metodi per indirizzare delle strutture interne ad un documento XML.

- la *trasformazione* dell'istanza del vocabolario XML in una nuova istanza coerente con il vocabolario della semantica di restituzione;
- la *formattazione* dell'istanza del vocabolario di restituzione nello user agent (processori posizionati su WEB server, WEB browser, palmari, cellulari, eccetera).

Per far fronte a queste esigenze il W3C ha messo a punto due raccomandazioni:

- la Recommendation XSL Transformations (XSLT) – descrive un vocabolario per trasformare la struttura dell'informazione prodotta nell'ambito dell'organizzazione in un'altra struttura, adeguata per i successivi passi elaborativi;
- la Recommendation Extensible Stylesheet Language (XSL) – descrive un vocabolario riconosciuto dall'agente di rendering per trasformare i formati espressi in forma astratta in particolari mezzi di trasformazione

Un esempio dell'uso di XSLT e XSL è quello di trasformare un documento XML in HTML e formattarlo per una adeguata presentazione in un WEB browser. Nel caso di un'architettura a più livelli, è possibile inviare l'informazione strutturata al WEB browser utilizzando XSLT sul *server*, sull'*user agent* o su *entrambi*. Da un punto di vista tecnico, il server può distribuire il carico elaborativo ad agenti XSL/XSLT inviando una combinazione di stylesheet e informazione sorgente da trasformare sul lato client; oppure effettuare tutte le trasformazioni direttamente sul server, in modo da poter supportare user agent che interpretano solo vocabolari HTML o HTML/CSS (Cascade Style Sheet).

Indipendentemente dalle caratteristiche del browser, la scelta se trasferire documenti con markup XML o con solo markup HTML (semanticamente meno ricchi) può dipendere da considerazioni legate all'opportunità di rendere disponibili a tutti le informazioni contenute nel documento XML completo. Una scelta potrebbe essere quella di trasformare il contenuto del documento in base alle caratteristiche del potenziale fruitore e lasciare all'user agent posizionato sul client, di personalizzare la formattazione.

Nel paragrafo 3.3.4, dedicato al layout di presentazione dei dati agli utenti, è stato scelto di utilizzare l'user agent presente sul server per eseguire sia la trasformazione che la formattazione ed inviare quindi al browser l'informazione direttamente in HTML.

3.1.2 I WEB Services

La tecnologia che si riferisce ai WEB services è attualmente una delle più discusse negli ambienti di settore.

Il termine WEB services descrive la prossima generazione di tecnologie utilizzabili sul WEB e rappresenta le fondamenta dell'elaborazione distribuita su Internet. Storicamente Internet, attraverso il WEB, è stato utilizzato per la pubblicazione di contenuti multimediali usufruibili direttamente

dalle persone. La funzione base del WEB, fino a questo memento, è stata quella di fornire dei contenuti che potessero essere visualizzati attraverso un browser. I WEB services rappresentano un ulteriore passo nell'uso del WEB, in cui la fornitura dei contenuti viene separata dall'uso che si intende fare dei contenuti stessi: l'utente "target" sarà in realtà un altro computer.

L'idea di considerare il WEB come un mezzo per presentare dei contenuti ad un alto computer, piuttosto che ad una persona, non è una idea nuova. Essa è sempre stata alla base della teoria sulla elaborazione distribuita. Nel corso degli anni ci sono state molte implementazioni dell'elaborazione distribuita. Due esempi significativi sono DCOM (Distributed Common Object Model), in ambiente Windows, e CORBA (Common Object Request Broker Architecture) principalmente in ambiente Unix. Quello che è cambiato è che con l'avvento di Internet e delle tecnologie ad essa direttamente collegate, l'implementazione dell'elaborazione distribuita è divenuta non proprietaria, economica e semplice da realizzare.

I WEB services sono la tecnologia alla base della elaborazione distribuita in ambiente Internet. Il primo grosso vantaggio, rispetto a tecnologie quali DCOM e CORBA, è che tale tecnologia specifica cosa fare senza dare nessuna indicazione su quali linguaggi o piattaforme utilizzare. In pratica si potrebbe dire che i WEB services hanno come scopo quello di far comunicare software differenti, realizzati su piattaforma differenti, attraverso linguaggi di programmazione differenti.

Alla base della tecnologia WEB services esistono una serie di protocolli:

- SOAP
- WSDL
- UDDI

3.1.2.1 Simple Object Access Protocol (SOAP)

SOAP è un protocollo, basato su XML, che consente a due applicazioni (un client e un server) di comunicare tra loro sul WEB. Pubblicato come Working Draft dal W3C nel dicembre 2001[10], SOAP definisce il formato dei messaggi che due applicazioni possono scambiarsi utilizzando i protocolli di Internet, come ad esempio HTTP, per fornire dati e richiedere elaborazioni. Il protocollo è indipendente dalla piattaforma hardware e software ed è indipendente dal linguaggio di programmazione utilizzato per sviluppare le applicazioni comunicanti.

In pratica SOAP rappresenta un documento XML che descrive una richiesta di elaborazione o il risultato di una elaborazione e risulta costituito dai seguenti elementi:

- Envelope – rappresenta il contenitore del messaggio e costituisce l'elemento root del documento XML;

[10] http://www.w3.org/TR/soap12-part0, http://www.w3.org/TR/soap12-part1, http://www.w3.org/TR/soap12-part2

- Header – è un elemento opzionale che contiene informazioni globali sul messaggio (per esempio la lingua di riferimento, la data di invio, eccetera);
- Body – rappresenta la richiesta di elaborazione o la risposta derivata da una elaborazione;
- Fault – se presente, fornisce informazioni sugli errori che si possono verificare durante l'elaborazione. Tale elemento è presente solo nei messaggi di risposta.

E' opportuno evidenziare che SOAP definisce soltanto la struttura dei messaggi non il loro contenuto. I tag per descrivere una richiesta di elaborazione o un risultato vengono definiti in un XMLSchema ed utilizzati all'interno della struttura SOAP sfruttando il meccanismo dei namespace. Segue un esempio di messaggio SOAP:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"
            soap:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding">
      <soap:Body xmlns:istat=http://www.istat.it/soap>
            <istat:GetTimeSeries>
                  <istat:codice> e-ip-11-CA11</istat:codice>
            </istat:GetTimeSeries>
      </soap:Body>
</soap:Envelope>
```

3.1.2.2 WEB Services Description Language (WSDL)

WSDL[11] è un documento XML che descrive un messaggio SOAP e come tale messaggio deve essere scambiato. Essendo WSDL un documento XML risulta leggibile ed editabile attraverso qualunque editore di testo anche se, in molti casi, esso viene generato e letto automaticamente da un software. Per capire l'importanza del WSDL, immaginiamo che un *provider* metta a disposizione dell'utenza un WEB service. Un utente che vuole accedere a tale WEB service deve richiedere informazioni su come fare. Per fornire tali informazioni il *provider* pubblica un documento in cui vengono fornite tali specifiche, cioè pubblica un WSDL.

Esistono molti tools capaci di leggere documenti WSDL e generare in automatico il codice necessario per comunicare con il WEB service corrispondente (per esempio wsdl.exe in ambiente .NET e WSDL2Java utilizzando Apache Axis in ambiente Java).

3.1.2.3 Universal Discovery and Integration (UDDI)

UDDI[12] può essere pensato come un luogo in cui segnalare l'esistenza di un dato WEB service e dove un utente può rivolgersi per avere notizie sull'esistenza o meno di un WEB service che

[11] http://www.w3.org/TR/wsdl
[12] http://www.uddi.org/about.html

soddisfi le proprie esigenze. Non è necessario che un WEB service venga registrato in un UDDI, ma la sua registrazione lo renderà visibile alla comunità. Una registrazione su un UDDI rappresenta una "directory entry" in un file XML che descrive il tipo di WEB service e ciò che offre. In generale la "directory entry" è composta da tre parti:

- white pages - descrive il provider che offre il servizio: denominazione, indirizzo, contatti, eccetera;
- yellow pages - descrive la categoria alla quale il provider appartiene (in genere viene utilizzato il sistema di classificazione "North American Industry Classification System");
- green pages - descrive l'interfaccia al servizio.

Le modalità di ricerca delle informazioni, presenti in un UDDI, seguono le WS-Inspection specification.

3.1.3 Standard for Data and Metadata Exchange (SDMX)

SDMX si riferisce ad una iniziativa di cooperazione tra diverse organizzazione internazionali volta a sviluppare ed impiegare processi più efficienti nello scambio di dati e metadati tra le stesse organizzazioni internazionali e i paesi membri. Il progetto iniziato nel 2001 è sponsorizzato da: Bank for International Settlements (BIS), European Central Bank (ECB), Eurostat, International Monetary Fund (IMF), Organisation for Economic Co-operation and Development (OECD), United Nations (UN) e World Bank (WB).

I paragrafi seguenti forniscono al lettore una panoramica sullo standard in maniera tale che possa facilmente orientarsi nella lettura dei documenti specialistici. Molta documentazione sull'argomento è disponibile nel sito WEB all'indirizzo: http://www.sdmx.org.

3.1.3.1 Gli obiettivi di SDMX

Alla base del progetto SDMX c'è la necessità di standardizzare lo scambio e la condivisione di dati e metadati. Tale necessità è diventata più pressante con lo sviluppo di Internet attraverso il quale lo scambio e la condivisione delle informazioni sono più semplici da effettuare, frequenti ed importanti. Quindi l'obiettivo principale è arrivare ad un insieme di standard riconosciuti da tutti gli attori che permettano l'accesso non solo ai dati statistici comunque essi siano dislocati, ma anche ai metadati che rappresentano il mezzo più efficiente per rendere i dati stessi facilmente interpretabili ed usabili. Tali standard aiuteranno gli Istituti nazionali a meglio interfacciarsi verso utenti e partner ed in particolare con gli organismi internazionali. Tra le altre cose essi permetteranno la facilitazione dell'uso di Internet per accedere a data base presenti sulla rete in maniera tale da ricercare i dati appena essi vengono resi disponibili.

Quindi oltre a migliorare le funzioni di reporting dagli stati membri verso gli organismi internazionali, SDMX potrà essere utilizzato efficacemente dagli Istituti nazionali per relazionarsi con altri organismi nazionali e in tutti quei caso dove sono state realizzate iniziative di decentramento dei sistemi statistici.

3.1.3.2 Forme di scambio e modalità di raccolta e accesso ai dati

Le possibilità di scambio di dati e metadati tra le varie organizzazioni sono diverse, esse dipendono dal numero dei partner coinvolti e dalla natura degli accordi:

- **bilateral exchange**: in questo caso si tratta di uno scambio tra due organizzazioni. In tal caso le organizzazioni stesse decidono i formati, la frequenza e la modalità di comunicazione;
- **gateway exchange**: in questo caso si tratta di uno scambio tra più di due organizzazioni. In tal caso è necessario un accordo tra le organizzazioni per definire un unico formato, solo in questo modo è possibile ridurre la gestione di diversi scambi di tipo bilaterale;
- **data-sharing exchange**: in questo caso non si tratta di un vero e proprio scambio di dati ma una condivisione. Le organizzazioni interessate, siano esse "providers" (quelli che forniscono i dati) che "consumers" (quelli che utilizzano i dati), dovranno aderire a standard comuni. Un possibile scenario in questi casi è la registrazione, da parte dei provider, in un unico sistema di raccolta, delle informazioni relative ai dati che vogliono mettere a disposizione nei loro sistemi, per cui gli utenti interessati possono accedere direttamente al sistema di raccolta che interrogherà i sistemi interessati restituendo all'utente i dati richiesti.

All'interno di queste forme di scambio, la modalità di accesso ai dati può essere di due tipi:

- **pull**: in cui le organizzazioni che diffondono i dati, li mettono a disposizione su un server a cui le organizzazioni utilizzatrici dovranno accedere. In questo caso è previsto un tipo di scambio data-sharing;
- **push**: in cui i produttori inviano periodicamente i dati agli utilizzatori interessati, utilizzando trasferimento di file o e-mail.

3.1.3.3 Concetti base di SDMX versione 1.0

Definiamo alcuni concetti base dell'SDMX che verranno utilizzati in seguito:

- **structural metadata**: rappresentano quei metadati che servono ad identificare e descrivere i dati. Ad esempio nel caso di "Popolazione residente per regione e sesso", le dimensioni regione e sesso rappresentano dei metadati strutturali;
- **reference metadata**: rappresentano quei metadati che descrivono i contenuti e la qualità dei dati statistici. I reference metadata possono essere a loro volta suddivisi in:
    - **conceptual metadata**: descrivono i concetti utilizzati e le loro relazioni;

    - **methodological metadata**: descrivono le metodologie usate per la creazione dei dati;
    - **quality metadata**: descrivono la qualità del risultato statistico;
- **observation**: rappresenta il singolo valore di una variabile osservato in un determinato istante di tempo;
- **time series**: rappresenta i valori assunti da una determinata variabile in diversi istanti temporali;
- **cross sectional**: rappresenta i valori ottenuti da più variabili in uno stesso istante temporale;
- **data group**: rappresenta un raggruppamento di più variabili in time series o variabili in cross sectional;
- **data set**: rappresenta un raggruppamento di più data group;
- **attachment level**: rappresenta il livello a cui vengono legati i metadati strutturali. I dati possono essere raggruppati, infatti, in modi differenti a seconda della loro tipologia. Se si vogliono inviare nello stesso documento SDMX più serie storiche o più section, le informazioni comuni alle serie o alle section verranno legate a livello di gruppo altrimenti se si vogliono diffondere più gruppi il livello di attachment sarà il dataset e così via. I valori che l'attachment level può assumere sono quattro: observation, time series o section, group e data set;
- **code list**: rappresenta una lista di valori assunti da un concetto statistico con i codici relativi. Una stessa code list può essere associata a più concetti statistici, ad esempio la code list associata al concetto "provincia di nascita dello sposo" è una lista con valori e codici che può essere utilizzata anche come code list per indicare la "provincia in cui è stato contratto il matrimonio". Una code list viene gestita (diffusa e aggiornata) da una unica istituzione (Agency);
- **key-family**: rappresenta un insieme di concetti che descrivono ed identificano un set di dat. Definisce quali concetti sono le dimensions (identificano e descrivono) e quali sono attributes (descrivono solamente) e definisce un attachment level per ciascun concetto basando ciò sul sistema di packaging (Data set, Group, Series, Observation);
- dimension: rappresenta un concetto statistico codificato ed assegnato ad una key-family;
- attribute: rappresenta un concetto statistico utilizzato per qualificare un dato.

3.1.3.4 Formati supportati dalla versione 1.0 dell'SDMX

L'SDMX nasce come evoluzione del formato standard già definito per lo scambio di dati GESMES (Generic Standard Message):

- SDMX-EDI rappresenta la versione GESMES/TS 3.0. Viene utilizzato maggiormente per l'invio di file batch contenenti serie storiche. Tale formato prevede la presenza di una "Structure definition" per esprimere i metadati strutturali e di un "Compact data" per lo scambio di dati e

codici che si riferiscono alla Structure definition e che presenta una struttura ottimizzata per l'invio di grosse quantità di dati;

- SDMX-ML rappresenta la versione XML di GESMES. In questo caso dati e metadati vengono scambiati in un formato XML. Le tipologie di documenti che sono previsti da questa versione sono:
    - **structure definition message**: contiene i metadati strutturali che servono ad interpretare i dati. Descrive le key family, code list e concetti;
    - **generic data message**: contiene sia i dati che i metadati. Può essere elaborato senza altri documenti aggiuntivi;
    - **compact data message**: viene utilizzato per lo scambio di grandi mole di dati. A differenza del generic data message, per essere interpretato correttamente, il sistema ricevente deve possedere le informazioni relative ai metadati strutturali (structure definition message);
    - **utility data message**: viene utilizzato oltre che per trasferire le key-family del data set, anche per supportare la validazione ed altre funzionalità degli schemi XML;
    - **cross sectional message**: è simile al compact data message ma, a differenza di quest'ultimo, permette il trasferimento dei dati che non sono organizzate in serie storiche, come ad esempio dati che rappresentano più variabili osservate nello stesso istante di tempo;
    - **query message**: rappresenta il formato SDMX per l'invio di query a database disponibili su WEB o a WEB services che riconoscono il formato SDMX.

3.2 Analisi del sistema prototipale per l'integrazione delle banche dati di diffusione

La progettazione e la realizzazione di un sistema capace di integrare, a livello applicativo, altri sistemi diventa tanto più complessa quanto più i sistemi coinvolti sono differenti sia in termini di informazioni trattate che in termini di tecnologie hardware e software utilizzate. Essendo molte le variabili che entrano in gioco, risulta diffide individuarle tutte ed esaminarle.

Lo scopo della ricerca non è stato quello di realizzare una soluzione completa, ma verificare una sua fattibilità. Quindi sono state analizzate le variabili principali e di conseguenza la progettazione è stata concentrata solo su alcune funzionalità. In particolare l'analisi si è concentrata:

- sull'esistente, con particolare riferimento ai due sistemi Conistat e Demo. Tali sistemi anche se non completamente rappresentativi di tutte le banche dati presenti in Istituto, hanno

caratteristiche completamente differenti. In particolare Conistat tratta dati in serie storiche, Demo dati organizzati per territorio;

- sulle minime funzionalità essenziali atte a soddisfare l'idea di base: uniformità di accesso ai dati senza sapere in quale banca dati sono memorizzati.

3.2.1 Analisi dell'esistente

L'attuale diffusione dei dati statistici da parte dell'Istat avviene attraverso una serie di banche dati esposte su Internet. Tali banche dati che è possibile definire settoriali perché in genere trattano un'area o un singolo settore statistico, presentano caratteristiche differenti.

Di seguito vengono riproposte alcune definizioni presenti sul sito WEB dell'Istituto:

"Le banche dati e gli altri sistemi informativi statistici sono a carattere tematico e forniscono una visione globale e accurata del fenomeno indagato. L'accesso è libero e gratuito.
Le *banche dati* sono magazzini in cui l'utente può scegliere in base alle proprie esigenze il tipo di dati e il loro livello di dettaglio e costruire le proprie tabelle in maniera personalizzata.
I *sistemi informativi* contengono informazioni e dati strutturati in tavole preconfezionate e scaricabili su foglio elettronico. Ogni banca o collezione di dati è corredata di metainformazioni (metodologie, classificazioni, definizioni) relative all'argomento trattato"[13].

Quindi all'interno del sito WEB vengono collocate sotto la voce "banche dati" sia le banche dati vere e proprie che i sistemi informativi. Inoltre vengono considerate le collezioni di tavole, cioè tavole statistiche prodotte generalmente alla conclusione di alcune indagini, come forma preliminare di diffusione dei dati prodotti.

In generale è possibile classificare ciascuna banca dati servendosi del seguente schema:

- per la frequenza con cui tali dati vengono raccolti:
    - congiunturali;
    - strutturali;
    - censuari;
- per dimensione dominante:
    - tempo;
    - territorio;
    - classificazione;
- per i tipi di dati trattati:
    - economici;
    - finanziari;

[13] Dal sito Istat all'indirizzo: http://www.istat.it/db_siti/

    - demografici;
    - sociali;
- per il tema interessato:
    - orizzontale (più indagini);
    - verticale (singola indagine);
- per architettura hardware e software utilizzata
    - unix con java;
    - unix con php;
    - windows con asp;
- per modalità di organizzare e memorizzare i dati:
    - in data base management system
        i. Oracle
        ii. Access
        iii. mySql
    - in file
        i. html
        ii. ascii formattato
        iii. zip
        iv. excel

In seguito viene esposto un elenco di banche dati così come proposto sul sito web:

- Censimento popolazione 2001

  Popolazione residente, stranieri, famiglie, persone che vivono in convivenze, grado di istruzione e condizione professionale dei cittadini; consistenza numerica e caratteristiche strutturali di edifici e abitazioni. Il dettaglio è fino al livello comunale.
- Censimento industria e servizi 2001

  Imprese, istituzioni pubbliche e non profit, unità locali e addetti, suddivisi per attività economica, classe di addetti e forma giuridica. Sono disponibili confronti con i censimenti dal 1951 in poi. Il dettaglio è fino al livello comunale.
- Censimento Agricoltura 2000

  Aziende agricole, nuove attività (colture biologiche, agriturismo, artigianato), nuove tecnologie, utilizzazione dei terreni, irrigazione, allevamenti, mezzi meccanici, forza lavoro e approcci al mercato.
- Censimento intermedio industria

L'universo delle imprese è stato individuato nella prima fase (31 dicembre del 1996); nella seconda fase (31 dicembre 1997) è stata approfondita l'analisi delle caratteristiche strutturali delle imprese attive, degli artigiani, dei lavoratori autonomi.

- Censimento istituzioni Non Profit 1999

  Numero delle istituzioni attive nel 1999 per forma giuridica, struttura organizzativa, tipo di assetto sociale, dimensione economica e settori di attività.

- Demo: demografia in cifre

  Popolazione residente per età, sesso e stato civile. Sono disponibili anche informazioni sui principali fenomeni demografici: nascite, permessi di soggiorno, indice di vecchiaia, età media, mortalità, previsioni della popolazione residente.

- Sistema di indicatori territoriali

  Indicatori di tipo demografico, sociale, ambientale ed economico riferito a ripartizioni, regioni, province e capoluoghi. Il sistema è articolato in 15 aree informative.

- Coeweb: statistiche del commercio estero

  Merci importate ed esportate da e nei Paesi che commerciano con l'Italia. I dati, aggiornati mensilmente, sono disponibili dal 1991 ad oggi.

- ConIstat: statistiche congiunturali

  Più di 15 mila serie storiche su prezzi, attività delle imprese, occupazione, retribuzioni, commercio estero e conti economici. L'interrogazione è possibile in base a settore di attività economica, gruppo di prodotto, voce del Sistema europeo dei conti.

- Disabilità in cifre

  Numero di disabili, istruzione e integrazione scolastica, turismo accessibile, sindrome di Down e altri aspetti della disabilità. Il sistema è utilizzabile anche dalle persone diversamente abili.

- Indicatori socio-sanitari regionali

  Sistema sanitario e salute nelle regioni italiane: salute, stili di vita e fattori di rischio; risorse impegnate dal SSN, domanda di assistenza sanitaria; indicatori di contesto demografico, sociale ed economico; stato di salute dell'ambiente.

- Health for All – Italia

  Oltre 4.000 indicatori su sanità e salute: contesto socio-demografico, mortalità per causa, malattie croniche e infettive, condizioni di salute e speranza di vita, disabilità, assistenza sanitaria, attività ospedaliera, risorse sanitarie.

- Dati sull'agricoltura e zootecnia.

  Coltivazioni, macellazione carni rosse e bianche, import-export bestiame, mezzi di produzione, pesca, caccia, floricoltura, forestali, lattiero caseario.

- Indicatori regionali di sviluppo.

  Indicatori statistici regionali sulla base dei quali viene condotta l'attività di valutazione e programmazione degli interventi da attuarsi nelle regioni Obiettivo 1, nell'ambito dei Fondi strutturali 2000-2006.

- Fondo monetario internazionale.

  Pagina riassuntiva contenente i dati più aggiornati e significativi prodotti da Istat, Banca d'Italia, Ministero dell'Economia e delle Finanze, Ufficio Italiano dei Cambi secondo gli standards richiesti dal DSBB del Fondo monetario internazionale.

- Cultura in cifre.

  Principali istituzioni culturali, pubbliche e private e servizi erogati con riferimento al settore dell'editoria, biblioteche, archivi, musei, gallerie, monumenti e scavi, sport, spettacoli, cinema, radio e televisione.

- SIA: sistema di indagini sulle acque.

  Statistiche sulle acque prodotte dall'Istat e, in particolare, dati ricavati dal Sistema di indagini sulle acque 1999. Il riferimento minimo territoriale è rappresentato da comuni e bacini idrografici.

- Sistema Informativo Territoriale sulla Giustizia.

  Tutte le statistiche sulla giustizia prodotte dall'Istat. Sono presenti le principali pubblicazioni, schede informative sulle indagini, glossario dei termini statistici utilizzati, normativa di riferimento.

- Rivalutazioni monetarie.

  Indici dei prezzi al consumo per le famiglie di operai e impiegati (FOI) al netto dei consumi di tabacchi: coefficienti di rivalutazione monetaria e variazioni percentuali.

- Dati congiunturali.

  Si rendono disponibili i principali indicatori statistici diffusi dall'Istituto in corrispondenza della emissione dei comunicati stampa sulla congiuntura economica.

Segue uno schema di classificazione delle sole *banche dati*, senza considerare i *sistemi informativi*:

Banca Dati	**Frequenza**	**Dimensione dominante**	**Tipo**	**Tema**	**Hw e Sw**
Censimento popolazione 2001	censuario	territorio	demografico	verticale	Unix Oracle Java
Censimento industria e servizi 2001	censuario	territorio	economico	verticale	Unix Oracle

					Java -BO
Censimento Agricoltura 2000	censuario	territorio	economico	verticale	Windows Oracle ASP -BO
Censimento intermedio industria	censuario	territorio	economico	verticale	Windows Oracle ASP -BO
Censimento istituzioni Non Profit 1999	censuario	territorio	economico	verticale	Windows Oracle ASP -BO
Demo: demografia in cifre	strutturale	territorio	demografico	verticale	Unix Postgres PHP
Sistema di indicatori territoriali	strutturale	Territorio tempo	economico demografico sociale finanziario	orizzontale	Unix Oracle JSP
Coeweb: statistiche del commercio estero	congiunturale strutturale	territorio tempo	economico	verticale	Windows Oracle ASP
ConIstat: statistiche congiunturali	congiunturale	tempo	economico	orizzontale	Windows Access ASP
Disabilità in cifre	strutturale	territorio tempo	sociale	verticale	Windows Access ASP
SIA: sistema di indagini sulle acque	strutturale	territorio	sociale	verticale	Unix Oracle Java
Sistema Informativo Territoriale sulla Giustizia	strutturale	territorio	sociale	verticale	Unix Oracle Java

3.2.2 Analisi del prototipo

L'idea progettuale ha come obiettivo principale la semplificazione degli accessi ai dati statistici attraverso l'uniformità di ricerca e presentazione delle informazioni. Per raggiungere tale obiettivo sono state seguite le seguenti specifiche:

- integrare l'acceso al sistema proposto, con la logica gerarchica (area, settore, argomento) presente nel nuovo sito WEB dell'Istituto;
- rendere uniforme la ricerca per tipi di dati differenti, siano essi a dominanza temporale che territoriale;
- presentare i dati ricercati in un numero limitato di layout (per esempio con il tempo in fiancata e le variabili in testata, oppure con il territorio in fiancata e le variabili in testata);
- realizzare un giusto compromesso tra ciò che i sistemi esistenti attualmente offrono e ciò che si potrà offrire attraverso l'integrazione, garantendo tempi di accesso e di risposta comparabili con quelli dei sistemi esistenti.

L'analisi del sistema ha cercato di individuare una soluzione realizzabile nel medio periodo e che, oltre a soddisfare i requisiti minimi necessari, comportasse dei vantaggi tali da giustificare le scelte effettuate. Gli scenari esaminati sono stati ricondotti a due schemi generali:

a) integrazione in architettura "data exchange";
b) integrazione in architettura "data sharing".

3.2.2.1 Integrazione in architettura data exchange

Tale architettura prevede un sistema centrale in cui i Servizi produttori depositano periodicamente i dati prodotti e i relativi metadati. Segue uno schema di tale architettura.

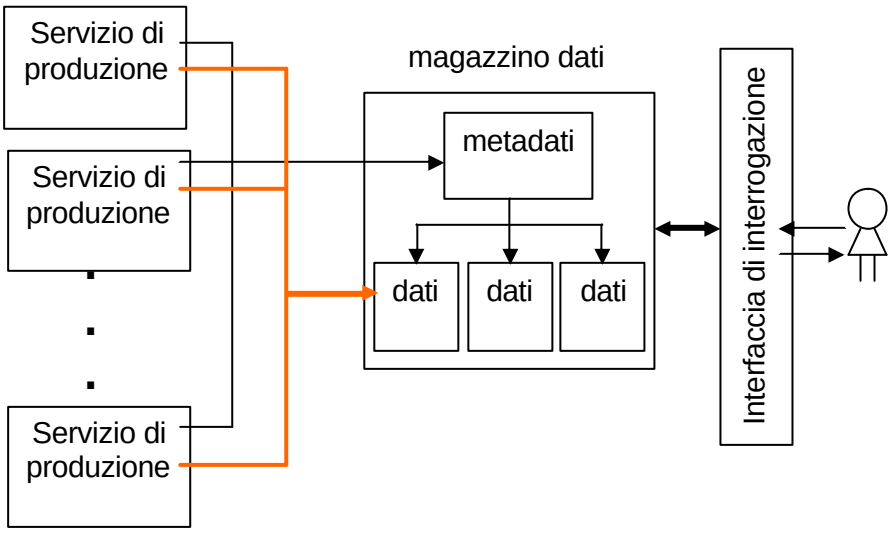


figura 3.1

Le funzioni che i vari soggetti coinvolti svolgono sono di seguito elencate:

1) i Servizi produttori inviano al sistema centrale la metainformazione che potrà essere validata centralmente;
2) i Servizi produttori inviano periodicamente, al sistema centrale, i dati e i rispettivi aggiornamenti[14];
3) il sistema centrale ripristina situazioni congruenti al verificarsi di problemi che possono avvenire durante i passi 1 e 2;
4) il sistema centrale provvede all'organizzazione ed ad eventuali elaborazioni dei dati raccolti;
5) i Servizi di produzione validano l'informazione inserita dopo ogni aggiornamento e prima che i dati vengano esposti su Internet;
6) il sistema centrale espone i nuovi dati su Internet.

I vantaggi di tale architettura risultano più evidenti nel momento in cui esiste poco decentramento della produzione dei dati, mentre iniziano a perdere di valore nel momento in cui il decentramento diventa più accentuato. Questo accade perché ciascuna unità produttiva, nel tempo, crea organizzazioni del lavoro e sistemi informativi differenti dalle altre unità. L'esperienza dimostra che tale tendenza può essere contrastata ma non annullata e, in ogni caso, le operazioni di contrasto risultano più efficaci proporzionalmente agli investimenti in risorse umane e strumentali che si intendono impiegare. Riassumendo, è possibile dire che il punto di forza di tale architettura consiste nell'integrazione nativa dei dati, che avviene a livello del contenitore centralizzato, mentre lo svantaggio principale consiste nella necessità di dover reingegnerizzare una parte consistente dell'organizzazione del lavoro e dei flussi informativi esistenti. Tale reingegnerizzazione risulta tanto più problematica quanto più unità produttive risultano coinvolte.

Nella specificità dell'attuale organizzazione dell'Istat, tale architettura, inserendosi in un contesto di banche dati già esistenti, dovrebbe prevedere una loro sostituzione. Ciò comporterebbe un ripensamento di tutta una serie di funzioni attualmente svolte dalle singole banche dati[15] al fine di garantire almeno gli attuali standard qualitativi. L'accentramento di alcune responsabilità avrebbe come logica conseguenza la necessità della creazione di un gruppo permanente di persone con il compito di realizzare centralmente molto di ciò che prima veniva svolto in periferia.

Questa soluzione, coinvolgendo sia la parte informatica che la parte organizzativa dei processi, oltre a comportare un aumento di richieste di mezzi finanziari e di personale, potrà essere realizzata solo nel lungo periodo.

[14] Tale funzione è tanto più critica quanto maggiore risulta la frequenza di aggiornamento.
[15] Tali funzioni sono quelle sintetizzabili con il termine *reporting* cioè raccolta, validazione e aggiornamento

3.2.2.2 Integrazione in architettura data sharing

L'architettura data sharing prevede un disegno ad "hub" in cui un sistema centrale colloquia con dei sistemi Satelliti facendo da tramite alle richieste degli utenti ed organizzando centralmente le modalità di presentazione delle risposte. Segue uno schema di tale architettura.

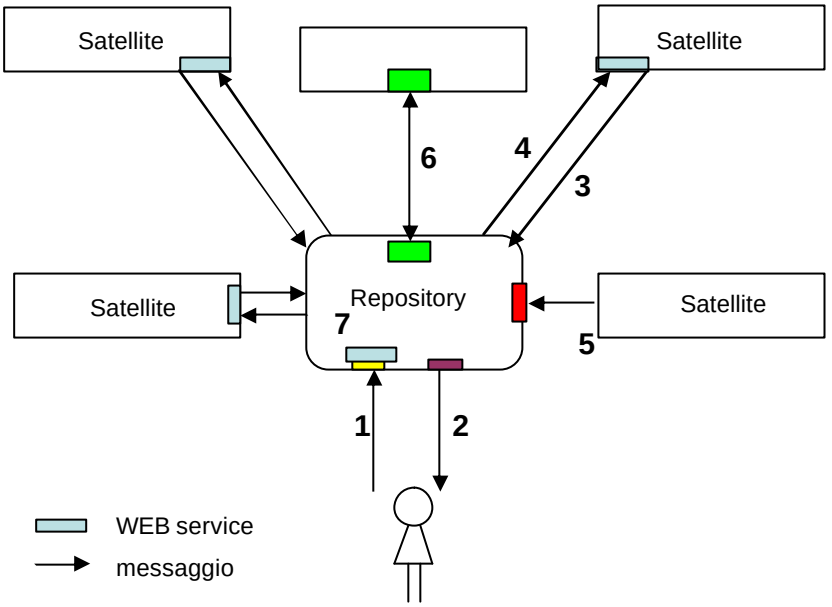


figura 3.2

Il colloquio tra il Repository ed ogni singolo Satellite avviene utilizzando dei protocolli ben definiti. Nel caso del prototipo realizzato sono stati utilizzati dei protocolli mutuati dal progetto SDMX. In pratica ciascun Satellite registra nel Repository la metainformazione dei dati che custodisce. L'utente attraverso l'interfaccia di navigazione della metainformazione, posizionata nel Repository, definisce quali dati sono di suo interesse. Il Repository indirizza la richiesta al Satellite in cui tali dati sono memorizzati. Il Satellite preleva i dati, li impacchetta secondo il protocollo di colloquio utilizzato e li invia al Repository. Quest'ultimo li formatta e li presenta all'utente.

I numeri evidenziati in figura presentano il seguente significato:

1. protocollo di ricerca dei dati attraverso la navigazione dei metadati strutturali;
2. protocollo che definisce il layout di presentazione dei dati in risposta;
3. protocollo di interrogazione del Satellite da parte del Repository;
4. protocollo di risposta del Satellite;
5. protocollo di registrazione del Satellite nel Repository;
6. protocollo di allineamento dei riferimenti temporali dei dati.
7. protocollo di autenticazione attraverso un sistema di SSO (Single Sign On)

Tale architettura viene adottata, in genere, in tutti quei casi in cui esistono, all'interno di un'organizzazione, sistemi complessi il cui rifacimento comporterebbe un onere eccessivo sia in termini economici che organizzativi.

In tale scenario lo svantaggio principale consiste nella complessità progettuale e realizzativa, mentre diversi sono i vantaggi che ne conseguirebbero:

a) utilizzo di funzionalità già presenti nei sistemi esistenti;
b) eliminazione di eventuali "colli di bottiglia" presenti, spesso, in architetture a forte vocazione centralizzata;
c) individuazione precisa delle responsabilità circa i flussi informativi coinvolti;
d) gestione e manutenzione della parte condivisa più semplificata rispetto all'architettura data exchange.

Nel caso dell'introduzione di tale architettura nel sistema informativo dell'Istat, si potrebbero ottenere dei vantaggi immediati così riassumibili:

a) raggiungimento di alcuni obiettivi già nel breve-medio periodo;
b) mantenimento dell'attuale infrastruttura organizzativa senza la necessità di creare nuove strutture;
c) mantenimento delle attuali banche dati per permette ad un'utenza più specialistica ricerche ed analisi più sofisticate;
d) utilizzo delle funzionalità di reporting già presenti nei sistemi Satelliti garantendo, come base di partenza, gli standard qualitativi già esistenti.

3.3 Progettazione di massima del prototipo

La progettazione del prototipo è stata basata sull'architettura data sharing. Ha previsto, quindi, la progettazione di un hub centralizzato con la sola funzione di essere l'intermediario tra le richieste degli utenti e le varie banche dati già esistenti. In pratica i dati non sono stati riorganizzati in un unico contenitore centrale ma verranno prelevati, a run-time, dai singoli contenitori di ciascuna banca dati.

Da un punto di vista tecnologico la progettazione del prototipo ha previsto l'uso di XML, attraverso le specifiche SDMX con particolare riguardo ai formati *query message* (in seguito sdmxQuery) e *compact data message* (in seguito sdmxCompact), e di WEB Services con protocollo SOAP.

Lo schema su cui si basa la progettazione del prototipo è quello proposto nella figura 3.2.

Come già anticipato, il prototipo non prevede tutte le funzionalità che un sistema completo dovrebbe possedere. Sono state studiate e progettate solo alcune delle funzionalità più importanti, i relativi strati software e le strutture di database d'interesse. Tali funzionalità, associate a moduli software ben definibili, vengono evidenziate, in seguito, per tipologia di localizzazione:

- nel Repository:
  - interfaccia utente di ricerca dei dati;
  - database e logica d'interrogazione;
  - client SOAP per colloquiare con i WEB services presenti nei Satelliti;
  - layout di presentazione dei dati all'utente;
- nei Satelliti:
  - WEB services per ricercare i dati richiesti dal Repository;
  - database di "mapping" tra WEB services e dati presenti nei Satelliti.

Non sono state affrontate le seguenti funzionalità:

- allineamento dei riferimenti temporali dei dati tra Repository e Satelliti;
- strato per la registrazione dei Satelliti presso il Repository;
- SSO (Single Sign-On) per l'accesso al Sistema.

L'intero prototipo è stato progettato per moduli software ben distinti (oggetti). Questo ha permesso, oltre ad una netta suddivisione dei compiti di programmazione tra varie persone, anche una facile gestione, manutenzione e riusabilità del codice. La tecnica utilizzata per scomporre l'intero sistema è stata quella definita top-down attraverso la quale, partendo da concetti generali, si riescono a riportare funzionalità "atomiche" in moduli elementari.

Il flusso di lavoro dell'intero sistema può essere così schematizzato:

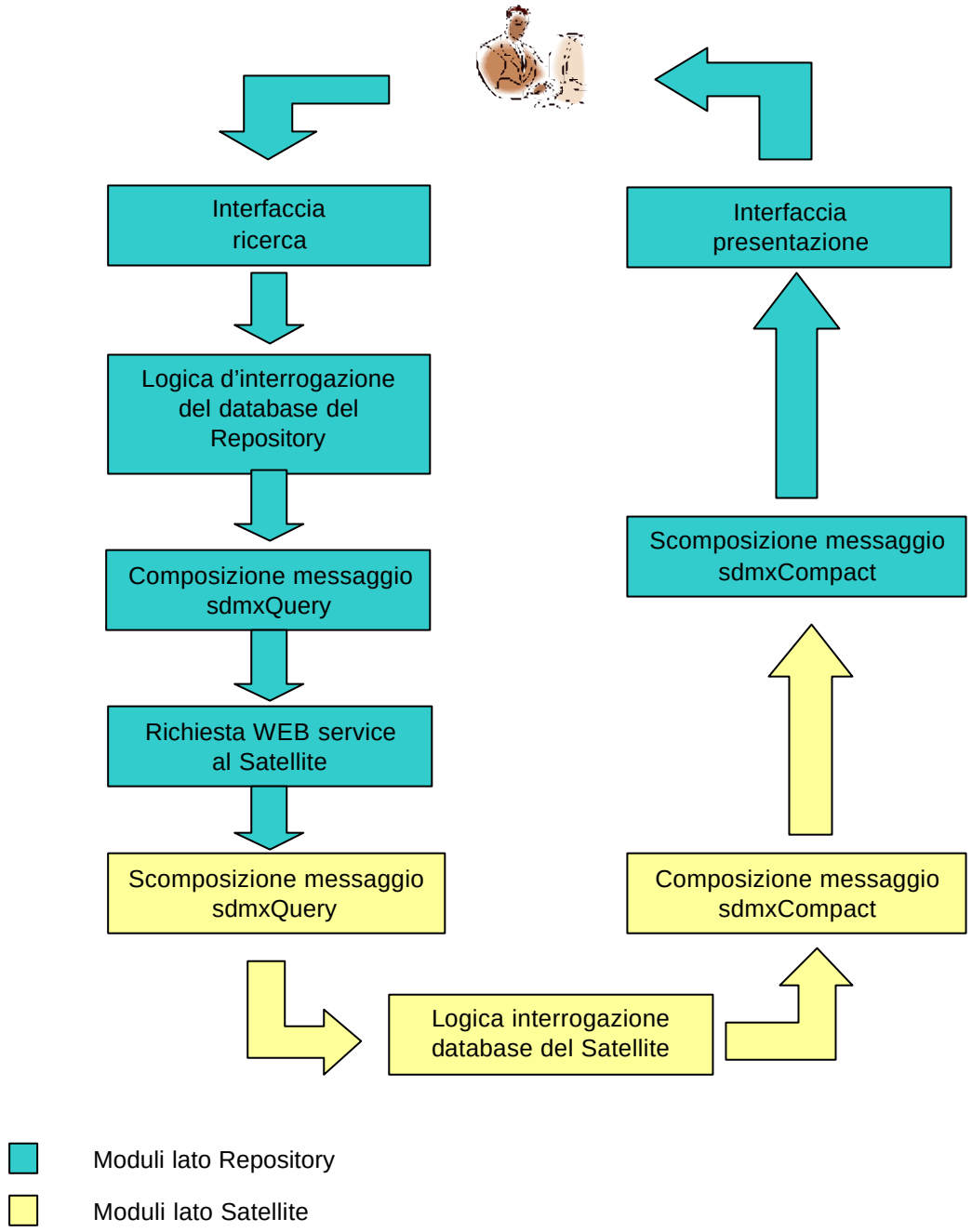
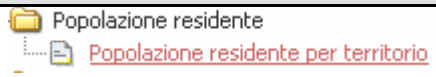
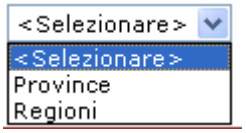
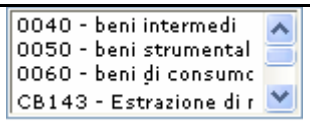
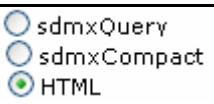

Interfaccia ricerca
Logica d'interrogazione del database del Repository
Composizione messaggio sdmxQuery
Richiesta WEB service al Satellite
Scomposizione messaggio sdmxQuery
Logica interrogazione database del Satellite
Composizione messaggio sdmxCompact
Scomposizione messaggio sdmxCompact
Interfaccia presentazione
Moduli lato Repository
Moduli lato Satellite

figura 3.3

3.3.1 Interfaccia utente di ricerca dei dati

Questo modulo, posizionato nel Repository, permette all'utente di interagire con il Sistema attraverso la navigazione dei metadati strutturali. In pratica l'utente segnala quali dati sono di suo interesse.

Gli oggetti d'interazione che sono stati presi in considerazione nel realizzare l'interfaccia utente sono i seguenti:

denominazione	immagine	descrizione
hyperlink		Attraverso il click del mouse permette di accedere ad un'altra pagina WEB
combo-box o drop-down-list		Permette di selezionare da una lista che si apre con un click del mouse, un solo elemento
list-box		Permette di selezionare uno o più elementi
option-box		Permette di effettuare una selezione tra tutte quelle proposte
button		Invia una richiesta al server

Tutti gli oggetti sopra elencati sono attivabili sia attraverso il mouse che attraverso la tastiera. Inoltre all'interno di ciascuna pagina tali oggetti sono stati predisposti in maniera tale da essere attivati seguendo una sequenza compatibile con un percorso ottimale di ricerca[16].

Prima di iniziare la progettazione, sono state definite le linee guida da seguire per ottimizzare l'interazione utente-sistema:

- ridurre il numero di interazioni (click attraverso il mouse);
- ridurre il numero di postback[17] verso il server;
- ridurre il numero di finestre che il sistema presenta all'utente durante le fasi di interazione;

[16] Attraverso il tasto "TAB" l'utente potrà passare da un oggetto ad un altro all'interno della stessa pagina. La sequenza di percorrenza rispecchia ciò che il sistema suggerisce come percorso "ottimo". Per esempio nel caso di una ricerca di dati territoriali la sequenza è la seguente: periodo di riferimento – disaggregazione territoriale – territori d'interesse – variabili d'interesse.

[17] Per postback si intende l'operazione che, al verificarsi di un evento, per esempio un click del mouse, avvia una procedura sul server. Tale operazione comporta un tempo d'attesa per l'utente, dato dalla somma del tempo impiegato dal messaggio, in formato HTTP che dal client viene inviato al server, dal tempo di elaborazione del server, dal tempo impiegato dal messaggio di risposta per arrivare dal server al client.

- aprire finestre di pop-up[18] in modalità padre-figlio nel caso di "interazioni discontinue", piuttosto che procedere con la stessa finestra. Tale eventualità avviene, come specificato meglio in seguito, tra il gruppo di interazioni circoscrivibili alla navigazione gerarchica attraverso l'albero delle cartelle (area, sottoarea, argomento) e il gruppo di interazioni che avvengono nella query-form[19]. In tal caso il passaggio dal primo al secondo gruppo avviene attraverso una discontinuità che si materializza nell'apertura di una nuova finestra di pop-up che lascia in background la finestra padre;
- definire un'area in cui visualizzare all'utente tutte le informazioni che il sistema ritiene necessario comunicare (per esempio i messaggi di errore o il completamento di una data operazione).

[18] La modalità pop-up avviene quando una finestra viene aperta sopra un'altra finestra lasciando generalmente visibili parti di quest'ultima. E' possibile invertire l'ordine di sovrapposizione delle due finestre semplicemente attraverso il click del mouse sulla finestra che si vuole passare in primo piano. Alcune volte è necessario inibire la possibilità di invertire l'ordine di sovrapposizione. Tale operazione non è facilmente ottenibile nei browser WEB, ed in ogni caso se realizzata non vale per tutti i tipi di browser.

[19] query-form: finestra che permette all'utente di specificare alcuni parametri necessari alla definizione della query che il Repository invierà al sistema Satellite interessato.

3.3.1.1 Dal sito WEB al Repository

L'interfaccia di ricerca è stata progettata per essere inserita in modo semplice nella logica di accesso "a portale" evidenziata nel nuovo sito WEB dell'Istat. In particolare l'utente che vuole ricercare dei dati viene guidato in una struttura gerarchica, facilmente percorribile, composta da aree (o domini), sottoaree (o sottodomini), argomenti (o categorie).

Il punto di partenza rimane la homepage del sito WEB. Per passare da quest'ultima alla pagina di navigazione, posizionata nel Repository, si hanno varie alternative:

- attraverso le voce di menù "banche dati" sul lato sinistro, oppure dalla voce "costruisci le tue tabelle personalizzate" sul lato destro. Tali voci permettono l'accesso alla pagina principale del Repository che visualizza l'albero[20] delle Aree al completo;

figura 3.3

[20] L'albero è rappresentato graficamente da libri, cartelle e fogli. I libri rappresentano le aree, le cartelle i settori e i fogli gli argomenti. Solo gli argomenti sono presentati in forma di hyperlink permettendo, quindi, il proseguimento della ricerca attraverso nuove pagine.

figura 3.4

- attraverso la scelta di una voce del menù "statistiche per argomento".

figura 3.5

Tale scelta porta ad una nuova pagina che si riferisce alla statistica selezionata.

figura 3.6

Nella parte di sinistra della pagina la voce “Banche dati” dà accesso alla parte dell’albero contenente come sottoarea la statistica selezionata.

figura 3.7

Le pagine presentate nelle figure 3.3, 3.5 e 3.6 fanno parte del sito WEB. Sarà necessario, quindi, predisporre gli opportuni hyperlink attraverso degli URL con passaggio di parametri in modalità “GET”[21]. Le figure 3.4 e 3.7 sono, invece, due rappresentazioni della stessa pagina (in seguito

[21] Il passaggio di parametri tra le varie pagine di un sito WEB o tra pagine di siti WEB differenti può avvenire attraverso due modalità, così come specificato dal protocollo HTTP: attraverso il metodo GET o attraverso il metodo POST.

denominata tree-form), posizionata sul Repository. La differente rappresentazione è dovuta alle diverse interazioni che l'utente effettua nelle pagine situate sul sito WEB.

La tree-form presenta sul lato sinistro un menù che riproduce, in forma verticalizzata, la parte centrale (quella presente sotto la testata "statistiche per argomento") della home-page del sito WEB dell'Istat. Tale menù permette di navigare all'interno delle aree e sottoaree senza dover tornare nelle pagine del sito WEB.

Nella parte destra della tree-form è stato posto un oggetto con struttura ad albero che permette di navigare attraverso le aree, sottoaree e argomenti. La scelta di un argomento da luogo all'apertura, in modalità pop-up, di una query-form.

Per uniformare le modalità di ricerche, è stato necessario limitare il numero di query-form a cui l'utente può accedere. A tal proposito sono state progettate due query-form: una orientata alla ricerca dei dati a dimensione temporale predominante (time series); una orientata alla ricerca dei dati a dimensione territoriale predominante (cross-sectional). Tali query-form presentano un numero di oggetti che potrebbero variare in funzione delle statistiche scelte. In ogni caso l'impostazione di base rimane identica.

Nell'ambito del prototipo lo sviluppo sia grafico che funzionale delle query-form è stato limitato all'essenziale; nella realizzazione di un sistema completo, si potrebbero prevedere ulteriori funzionalità quali:

- accesso ai metadati strutturali per ciascuna variabile;
- accesso ai metadati metodologici;
- personalizzazione del layout di presentazione;
- analisi sui dati attraverso calcolo di variazioni (congiunturali, tendenziali, cumulate, ecc.), visualizzazioni di grafici e mappe tematiche;
- scelta del formato di presentazione dei dati (CSV, Excel, Ascii, SDMX, ecc.).

Entrambe le query-form, progettate per il prototipo, presentano alcuni elementi in comune:

- possibilità di impostare, attraverso degli option-box, cosa ricevere in risposta:
  - sdmxQuery – indica al layout di presentazione di visualizzare uno stream in formato XML con grammatica SDMX che rappresenta il messaggio che il Repository invia al Satellite coinvolto per interrogarlo;
  - sdmxCompact – indica al layout di presentazione di visualizzare uno stream in formato XML con grammatica SDMX che rappresenta il messaggio, contenente i dati richiesti, che il Satellite coinvolto restituisce al Repository;
  - HTML - indica al layout di presentazione di trasformare lo stream in formato sdmxCompact in HTML e visualizzarlo;

- controllo, attraverso degli algoritmi realizzati in javascript[22], dell'avvenuta scelta dei parametri essenziali prima dell'avvio della ricerca attraverso il pulsante "Invia richiesta".

3.3.1.2 Query-form per i dati con dimensione dominante temporale

Tale query-form è stata progettata tenendo in considerazione la logica d'interrogazione presente già in Conistat: scelta delle variabili che si intendono visualizzare e scelta del periodo di riferimento (periodo iniziale, anno iniziale, periodo finale, anno finale).

figura 3.8

Le variabili che si riferiscono all'argomento statistico selezionato nell'albero di ricerca vengono presentate in una list-box, mentre in delle combo-box il sistema presenta le scelte possibili che si possono effettuare per l'intervallo temporale. I limiti dell'intervallo temporale che l'utente può scegliere, variano a secondo delle serie storiche che vengono proposte:

- il limite minimo dell'intervallo temporale viene calcolato considerando il minimo anno iniziale di tutte le serie storiche presenti nella listbox e su tale anno il minimo periodo iniziale;
- il limite massimo dell'intervallo temporale viene calcolato considerando il massimo anno finale di tutte le serie storiche presenti nella listbox e su tale anno il massimo periodo finale.

C'è da notare che il periodo iniziale e il periodo finale sono entrambi funzioni della periodicità (o frequenza) della serie storica presenti nella listbox. In particolare nel caso di:

[22] Javascript è un linguaggio di programmazione che viene interpretato direttamente dal browser. Attraverso tale linguaggio generalmente vengono eseguite operazioni di convalida sui campi presenti nella maschera.

- frequenza mensile, i periodi potranno essere scelti da un valore 1 ad un valore 12 in quanto rappresentano i mesi;
- frequenza trimestrale, i periodi potranno essere scelti da un valore 1 a un valore 4 in quanto rappresentano i trimestri;
- frequenza annuale, i periodi potranno assumere solo il valore 1 in quanto l'anno coincide con il periodo.

Sempre nell'ambito della scelta del periodo di riferimento, il sistema prima di avviare una ricerca controlla, attraverso una funzione realizzata in Javascript se l'utente ha selezionato correttamente il limite (anno + periodo) iniziale e quello finale: il limite iniziale dovrà risultare essere inferiore o uguale al limite finale. Nel caso ciò non avvenga, il sistema segnala all'utente la erronea scelta effettuata e non avvia la ricerca.

3.3.1.3 Query-form per i dati con dimensione dominante territoriale

La logica con la quale tale query-form è stata progettata è molto simile a quella seguita nella banca dati Demo e, in ogni caso, in molte banche dati che hanno come dimensione dominante il territorio:

- scelta del periodo di riferimento;
- scelta del livello di disaggregazione territoriale con cui ricercare i dati;
- scelta dei territori d'interesse;
- scelta delle variabili.

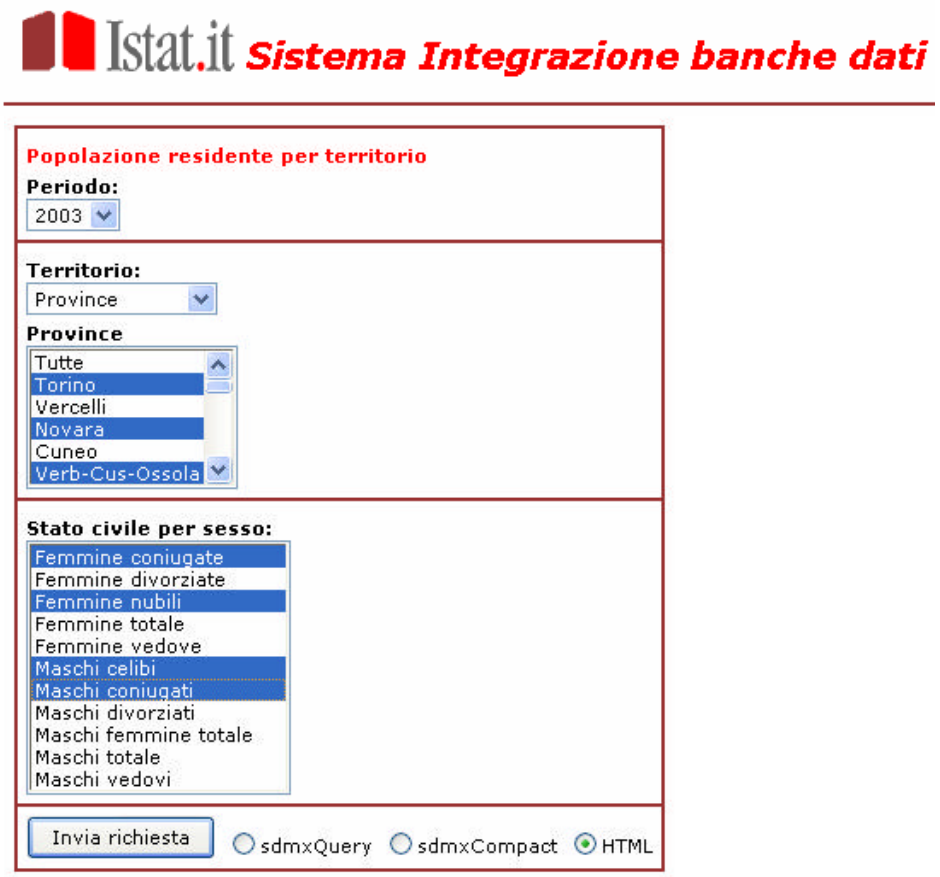


figura 3.8

La scelta del periodo di riferimento permette al sistema di proporre come territori e variabili valori congruenti. Per esempio scegliendo un determinato anno il sistema permette di effettuare la selezione territoriale tra le province esistenti in quell'anno. Scegliendo un altro anno le selezioni territoriali possibili potrebbero variare. Ciò è dovuto al fatto che nel tempo si possono formare nuove province o se ne possono accorpare altre.

3.3.2 Database del Repository e logica d'interrogazione

Nel Repository è stato previsto un database organizzato in più aree: una per memorizzare la struttura dell'albero di ricerca, una per memorizzare i metadati strutturali del Satellite Conistat, una per memorizzare i metadati strutturali del Satellite Demo.

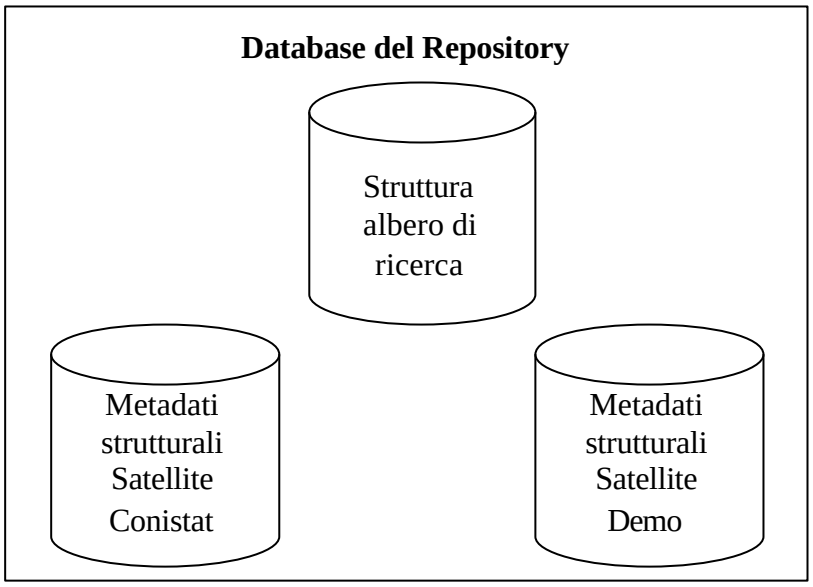


figura 3.9

Ad eccezione dell'area che si riferisce all'albero di ricerca che contiene un'unica tabella, le altre due aree prevedono una tabella principale, in cui vengono memorizzate le key family che si riferiscono a ciascun Satellite, e delle tabelle thesaurus in cui vengono memorizzate le code list.

Le tabelle thesaurus sono in relazione 1 a N con la tabella principale e permettono all'utente di ricercare i dati attraverso le dimensioni che rappresentano. In pratica una volta scelte le dimensioni di ricerca, le tabelle thesaurus corrispondenti, agiscono come tabelle di lookup[23] verso l'interfaccia utente.

Per quanto riguarda le tabelle principali delle aree che si riferiscono ai Satelliti, per ragioni di performance, bisogna considerare le loro dimensioni in termini di record contenuti. Un eccessivo numero di record potrebbe portare ad una riduzioni delle prestazioni del sistema. Dalle prove effettuate tale limite può attestarsi attorno ai 25.000 record. Nel caso la registrazione di un Satellite prevede il superamento di tale limite, è consigliabile suddividere tale tabelle in più tabelle organizzando i dati per argomenti omogenei. Questa operazione non comporta dei cambiamenti nell'architettura del Registry perché ciascuna delle tabelle ottenuta, dopo l'operazione di suddivisione, potrà essere vista come riferimento a un diverso Satellite virtuale.

[23] Le tabelle di lookup permettono di riempire gli oggetti, quali combo-box, list-box e option-box, dell'interfaccia utente di ricerca. Attraverso tali oggetti l'utente attua delle selezioni.

Volendo fare un esempio consideriamo il caso del Satellite Conistat. Attualmente la tabella di riferimento presenta circa 16.000 record. Ipotizziamo che tale numero cresca oltre il limite individuato. Allora si potrebbe pensare a più tabelle di riferimento: una per le statistiche che si riferiscono ai Prezzi, una per le statistiche che si riferisco alla Produzione industriale, una per le statistiche che si riferiscono al Commercio al dettaglio, eccetera.

3.2.2.1 Area del database in cui viene memorizzata la struttura dell'albero di ricerca

Quest'area del database risulta composta da una sola tabella, denominata TREE. La funzione della tabella TREE è quella di memorizzare tutti i nodi dell'albero (record) e le caratteristiche di ciascuno di esso (campi). Tale tabella presenta ciò che in letteratura viene definita "relazione circolare". In pratica la primary-key della tabella è in relazione 1 a N con le foreign-key della stessa tabella. Nel nostro caso inoltre esiste la particolarità che non viene memorizzato il nodo radice, quindi esistono dei record, cioè quelli che memorizzano i nodi che discendono direttamente dalla radice, che sono "orfani": le foreign-key di questi nodi presentano tutti il valore 0 e non risultano relazionati con nessun valore della primary-key.

La tabella a "relazione circolare" presenta una certa difficoltà nel momento in cui deve essere utilizzata da un applicativo. Il ricorso ad essa è stato necessario perché l'albero di ricerca non è definito a priori: nel tempo può variare in maniera casuale con l'inserimento di nuovi nodi, rami e foglie.

Nel caso l'albero di ricerca fosse stato definito a priori, sarebbero bastate più tabelle, una per ciascun livello di profondità dell'albero, relazionate in maniera tale da formare una struttura gerarchica.

La tabella TREE presenta la seguente struttura:

TREE		
Nome campo	**Tipo di dati**	**Descrizione**
Id	Testo	primary-key
Parent	Testo	foreign-key
Html	Testo	Titolo della voce che viene visualizzata nel nodo
Url	Testo	Indirizzo a cui si accede attraverso questa voce
Foglia	Boolean	Indica se il nodo è una foglia
Icon	Testo	Link all'immagine del nodo
Expanded	Boolean	Indica se il nodo deve apparire espanso al primo caricamento

I campi Id e Parent contengono valori che rappresentano un codice "parlante". In pratica, ad eccezione dei record che rappresentano i nodi discendenti direttamente dalla radice (contengono sempre un solo carattere rappresentato da uno zero), tutti gli altri record contengono un numero di caratteri pari. Tali caratteri presentano il seguente significato:

- ogni due caratteri consecutivi rappresentano un livello di profondità dell'albero. Per esempio se un nodo presenta una primary-key di n caratteri significa che oltre al nodo radice è preceduto da altri n/2-1 nodi;
- se un nodo da luogo ad un ulteriore livello di profondità rappresentato da più nodi, tali nodi assumono gli ultimi due caratteri in ordine crescente. Per esempio se il nodo la cui primari-key ha valore 07 dà luogo ad un ulteriore livello composto da due nodi, gli stessi assumeranno come primary-key i valori rispettivamente 0701 e 0702.

Segue uno schema ad albero che mette in evidenza i caratteri utilizzate nelle chiavi:

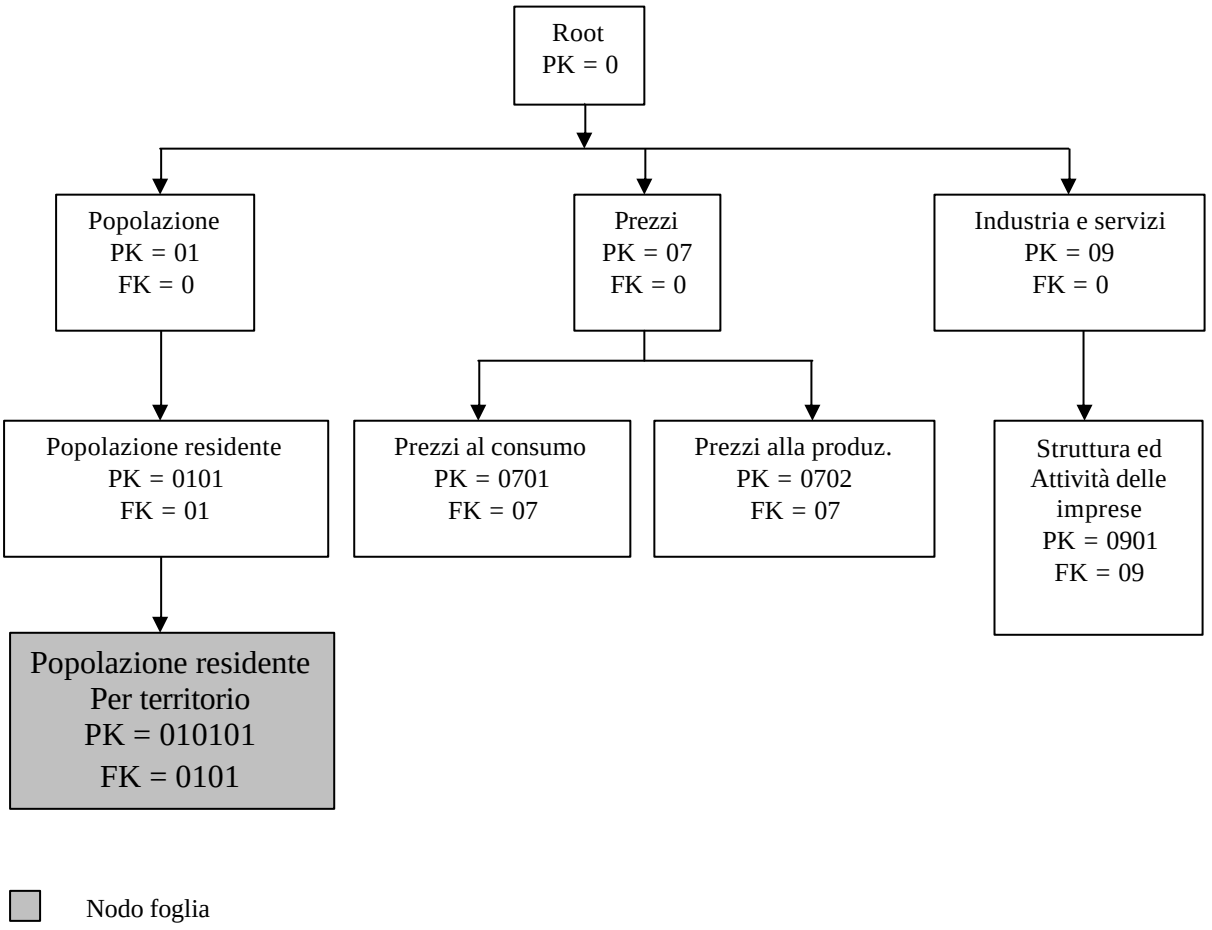


figura 3.10

3.2.2.2 L'area del database in cui è stato registrato il Satellite Conistat

La struttura del database in cui viene registrato il Satellite Conistat, è rappresentata da una tabella principale, metadatiConistat, e un insieme di tabelle di lookup. La tabella metadatiConistat presenta la seguente struttura:

metadatiConistat		
Nome campo	**Tipo di dati**	**Descrizione**
m_domominio	Testo	Dominio Statistico di appartenenza
m_s__dominio	Testo	Sottodominio Statistico di appartenenza
m_s_s_dominio	Testo	Chiave Primaria
tipo	Testo	Grezzo/Destaggionalizzato/Corretto
vs	Testo	Valore assoluto/relativo
fk_classificazioni	Testo	Codice Ateco/Deco/Sec/Altro
freq	Testo	Periodicità
u_mis	Testo	Unità di misura
anno_in	Testo	Anno iniziale
periodo_in	Testo	Periodo iniziale
anno_fin	Testo	Anno finale
periodo_fin	Testo	Periodo finale

La figura seguente illustra lo schema completo della parte del database che si riferisce al Satellite Conistat:

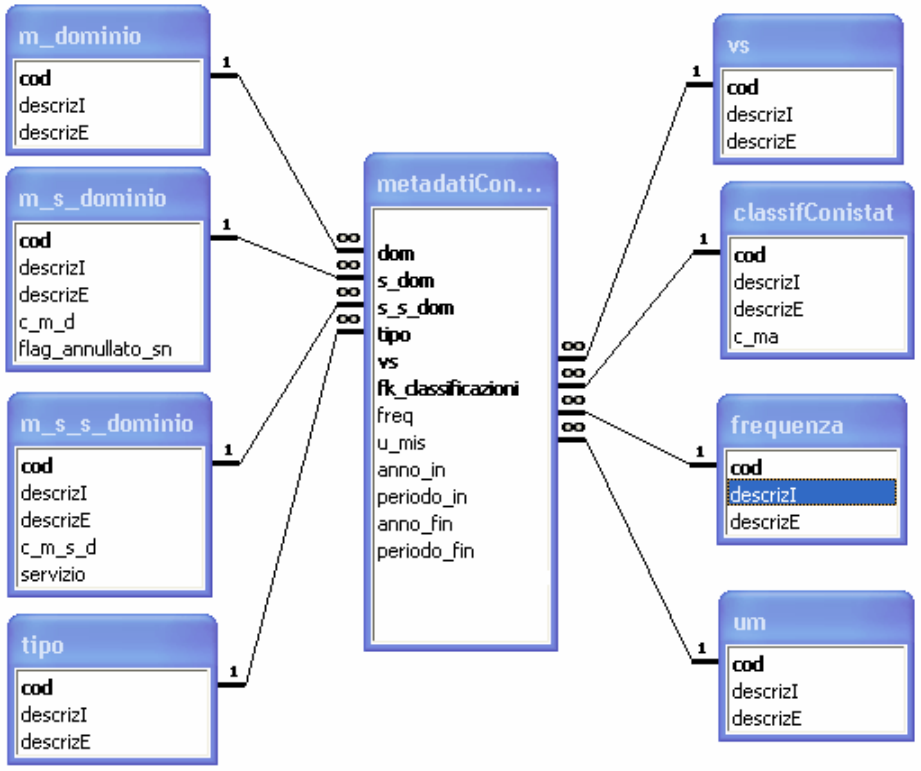


figura 3.11

3.2.2.3 La logica d'interrogazione del Satellite Conistat

La logica d'interrogazione del Satellite Conistat da luogo alla realizzazione di un messaggio sdmxQuery inviato come parametro nel momento in cui viene interrogato il WEB service messo a disposizione dal Satellite remoto.

E' stato scelto di far ricercare i dati secondo le dimensioni "argomento" e "classificazione" per cui la logica d'interrogazione può essere così schematizzata:

1. l'utente attraverso l'albero di ricerca si muove lungo la dimensione "argomento" ed effettua la selezione d'interesse. La dimensione "argomento" è referenziata nella tabella metadatiConistat attraverso il campo s_s_dom, mentre la code list corrispondente è memorizzata nella tabella m_s_s_dominio;
2. il sistema filtra i record della tabella metadatiConistat secondo la selezione effettuata nel punto precedente. Tale operazione di filtraggio viene realizzata contemporaneamente ad una operazione di join tra la tabella metadatiConistat e la tabella classifConistat (metadatiConistat.fk_classificazioni=classifConistat.cod);
3. il risultato della query viene proposto in una listbox a scelta multipla in cui l'utente potrà selezionare le serie storiche d'interesse;
4. l'utente seleziona l'intervallo temporale d'interesse.

A questo punto il sistema ha tutte le informazioni necessarie per creare il messaggio sdmxQuery e chiamare il WEB service remoto.

Segue un esempio di messaggio sdmxQuery in cui per ragioni di semplicità è stato omesso il contenuto dell'Header:

```
<?xml  version="1.0" encoding="UTF-8"?>
<QueryMessage>
   <Header>
      . . . . . .
      . . . . . .
   </Header>
   <Query>
      <DataFrom>metadatiConistat</DataFrom>
      <DataWhere>
         <And>
            <DataSet>11</DataSet>
            <Dimension Name="class">CA</Dimension>
            <Dimension Name="class">CB</Dimension>
            <Dimension Name="class">DA</Dimension>
            <Time>
               <StartTime>2005-2</StartTime>
               <EndTime>2005-4</EndTime>
            </Time>
         </And>
      </DataWhere>
   </Query>
</QueryMessage>
```

3.2.2.4 La struttura del database in cui è stato registrato il Satellite Demo

La struttura del database in cui viene registrato il Satellite Demo, è rappresentata da una tabella principale, metadatiDEMO, e un insieme di tabelle di lookup. La tabella metadatiDEMO presenta la seguente struttura:

metadatiDEMO		
Nome campo	**Tipo di dati**	**Descrizione**
pk_metadati	contatore	
fk_classificazioni	Testo	Variabili di classificazione
fk_argomento	Testo	Argomenti statistici
fk_sesso	Testo	Maschi/femmine
fk_disaterr	Testo	Nazionale/ripartizionale/regionale/provinciale/comunale
fk_statocivile	Testo	Nubile/celibe
fk_periodo	Testo	Anno di riferimento

Il campo fk_classificazione, è stato introdotto per referenziare una ulteriore key family che rappresenta il prodotto cartesiano delle key family memorizzate nel campo fk_sesso e quelle memorizzate nel campo fk_statocivile. Tale forzatura è stata fatta solo per mantenere la ricerca molto simile a quella presente in Demo.

La figura seguente illustra lo schema completo della parte del database che si riferisce al Satellite Demo:

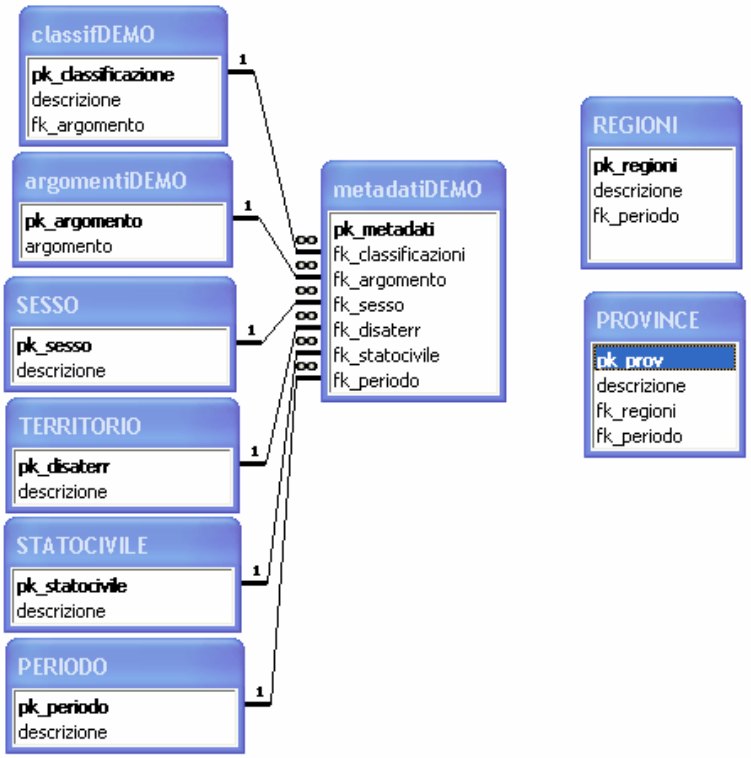


figura 3.12

Le tabelle di lookup che vengono utilizzate nell'interfaccia utente sono:

- la tabella PERIODO riempie una combo-box che permette all'utente di scegliere il periodo di riferimento;
- la tabella TERRITORIO riempie una combo-box che permette all'utente di scegliere un valore di disaggregazione territoriale (Regioni/Province);
- le tabelle REGIONI e PROVINCE riempiono alternativamente, a secondo della scelta fatta a livello di disaggregazione territoriale e di periodo, una list-box a scelta multipla in maniera tale che l'utente possa scegliere i territori d'interesse;
- la tabella classifDEMO riempie una list-box a scelta multipla che permette all'utente di selezionare le variabili d'interesse.

Le altre tabelle, SESSO e STATOCIVILE, allo stato attuale non hanno una particolare utilizzazione, ma in futuro potrebbero permettere di fare delle ricerche attraverso le dimensioni che rappresentano.

3.2.2.5 La logica d'interrogazione del Satellite Demo

La logica d'interrogazione del Satellite Demo dà luogo alla realizzazione di un messaggio sdmxQuery che viene inviato come parametro nel momento in cui viene interrogato il WEB service messo a disposizione dal Satellite remoto.

E' stato scelto di far ricercare i dati secondo le dimensioni "argomento", "territorio" e "classificazione", per cui la logica d'interrogazione può essere così schematizzata:

1. l'utente attraverso l'albero di ricerca si muove lungo la dimensione "argomento" ed effettua la selezione d'interesse. La dimensione "argomento" è referenziata nella tabella metadatiDEMO attraverso il campo fk_argomento, mentre la code list corrispondente è memorizzata nella tabella argomentiDEMO;
2. l'utente seleziona il periodo d'interesse;
3. il sistema, utilizzando la selezione effettuata al punto 1., filtra i livelli di disaggregazione territoriale;
4. il sistema, utilizzando la selezione effettuata al punto 1. e al punto 2., filtra i territori (regioni o province) disponibili;
5. il sistema filtra le variabili disponibili per la categoria scelta nel punto1. e le presenta in una list-box a scelta multipla.

A questo punto il sistema ha tutte le informazioni necessarie per creare il messaggio sdmxQuery e chiamare il WEB service remoto.

Segue un esempio di messaggio sdmxQuery in cui per ragioni di semplicità è stato omesso il contenuto dell'Header:

```
<?xml  version="1.0" encoding="UTF-8"?>
<QueryMessage>
   <Header>
      . . . . . .
      . . . . . .
   </Header>
   <Query>
      <DataFrom>
         <DataSet>metadatiDEMO</DataSet>
      </DataFrom>
      <DataWhere>
         <And>
            <Dimension Name="class">fdiv</Dimension>
            <Dimension Name="class">fnub</Dimension>
            <Dimension Name="disaterr">3</Dimension>
            <Territory>
              <Dimension Name="terr">213</Dimension>
              <Dimension Name="terr">217</Dimension>
              <Dimension Name="terr">218</Dimension>
           </Territory>
           <Time>
              <StartTime>2003</StartTime>
              <EndTime>2003</EndTime>
           </Time>
           <DataSet>PRT</DataSet>
         </And>
      </DataWhere>
   </Query>
</QueryMessage>
```

3.3.3 Il documento sdmxQuery e il client SOAP

Alla base della tecnologia WEB Services c'è la possibilità che un sistema (client) richieda un servizio ad un sistema remoto (server), specificando anche come tale servizio deve essere realizzato. Volendo fare un esempio nell'ambiente classico di programmazione, è come se un programma principale chiamasse una procedura passandole dei parametri. In genere il client SOAP invoca un WEB service passando una o più variabili.

Nel caso in esame, la variabile che viene passata è di tipo testo e contiene una stringa rappresentante l'sdmxQuery.

Questa parte è stata suddivisa in:

- un oggetto per creare una stringa contenente un documento sdmxQuery;
- una funzione di richiamo del server SOAP;

L'oggetto che crea la stringa è stato realizzato come oggetto "stand-alone" e quindi utilizzabile anche al di fuori dell'applicativo WEB. Tale oggetto espone un certe numero di metodi e proprietà[24] che permettono di inserire tutti gli elementi e attributi richiesti dal formato sdmxQuery.

Per realizzare l'oggetto, dopo aver esaminato tutte le classi messe a disposizione da ASP.NET, è stato deciso di utilizzare `XmlTextWriter`, e di mantenere lo stream XML che si ottiene in

[24] Le proprietà ed i metodi sono descritti nel paragrafo "Realizzazione del prototipo".

memoria senza scrivere un file su disco. Questo oltre ad aumentare le performance, non necessità una funzione di pulizia del file system.

La funzione di richiamo del server SOAP è stata parametrizzata in maniera tale poter impostare sia l'indirizzo WEB che l'identificativo[25] per collegarsi al server SOAP. La risposta, sotto forma di stringa in formato sdmxCompact, del server SOAP viene passata al modulo "layout di presentazione".

Tutta la complessità del colloquio tra client e server SOAP viene mascherato da uno "stub[26]" costruito direttamente dall'ambiente di sviluppo.

3.3.4 L'interfaccia di presentazione dei dati all'utente

L'interfaccia di presentazione è un modulo software che permette agli utenti, che accedono al Registry, di ricevere in una forma facilmente interpretabile, le informazioni che hanno richiesto. Per la realizzazione del prototipo la progettazione di tale interfaccia è stata limitata all'essenziale:

- presentazione dei dati in forma tabellare con i periodi in fiancata e le variabili in testata nel caso time-series; con il territorio in fiancata e le variabili in testata nel caso cross-sectional.;
- presentazione della decodifica codice-titoli delle variabili in forma tabellare.

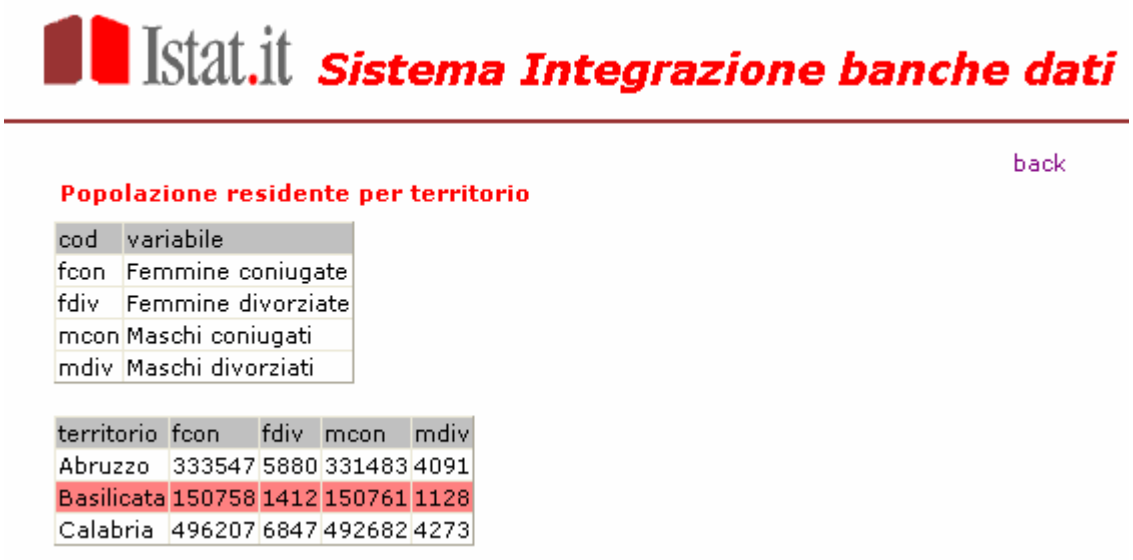


figura 3.13

Questo modulo software riceve come input la risposta che ciascun satellite invia dal proprio WEB service. Tale risposta è una stringa che rappresenta un documento XML organizzato secondo le regole grammaticali di un "SDMX compact data message".

[25] I WEB services sono stati protetti attraverso una userid e una password.

[26] Lo stub è il termine tecnico per indicare, in questo caso, un oggetto che implementa il protocollo SOAP. Il programmatore non ha necessità di conoscere come il protocollo deve essere implementato in quanto l'ambiente permette di creare un oggetto le cui le proprietà e metodi mascherano la complessità del protocollo SOAP.

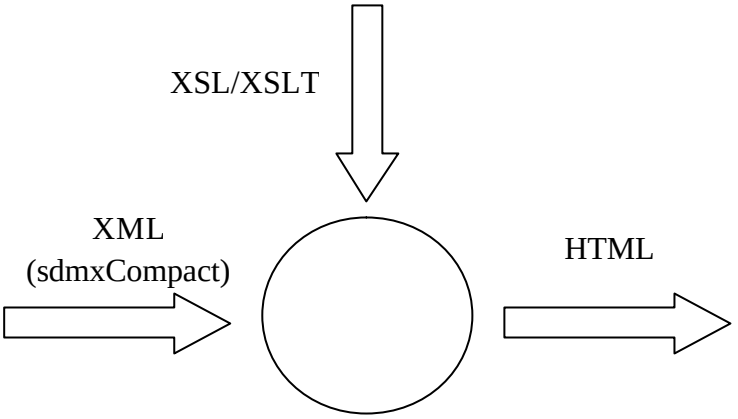


figura 3.14

Il documento XML in ingresso, viene trasformato, attraverso le tecnologie XSL/XSLT, in un documento HTML. Tale trasformazione viene realizzata direttamente sul server in maniera tale da garantire anche quei WEB browser che non posseggono un user agent capace di attuare la trasformazione.

Da un punto di vista tecnico, vengono utilizzati alcuni oggetti nativi di ASP.NET capace di gestire facilmente sia documenti XML che trasformazioni XSL/XSLT:

- la stringa di input viene caricata in un oggetto `XPathDocument` che permette di mantenere in memoria il documento XML senza doverlo salvare su disco migliorando, in tal caso, le performance del sistema;
- attraverso un oggetto `XslTransform` viene caricato da disco il foglio di stile opportuno;
- utilizzando il metodo `Transform` dell'oggetto `XslTransform` si realizza la trasformazione e, il risultato, sotto forma di uno stream HTML, viene inviato direttamente al browser.

3.3.5 I Satelliti

I Satelliti colloquiano con il Repository attraverso due WEB services. Ciascuno di essi ha le seguenti funzionalità:

1. accettano richieste con il passaggio di un parametro di tipo stringa;
2. interpretano la stringa in formato sdmxQuery scomponendola in elementi tali da poter produrre delle query SQL;
3. interrogano, nel caso è stato previsto, una o più tabelle di mapping[27];
4. interrogano il database proprietario[28], estraendo i dati richiesti;
5. compongono una stringa in formato sdmxCompact;
6. compongono una risposta SOAP e la inviano al client richiedente.

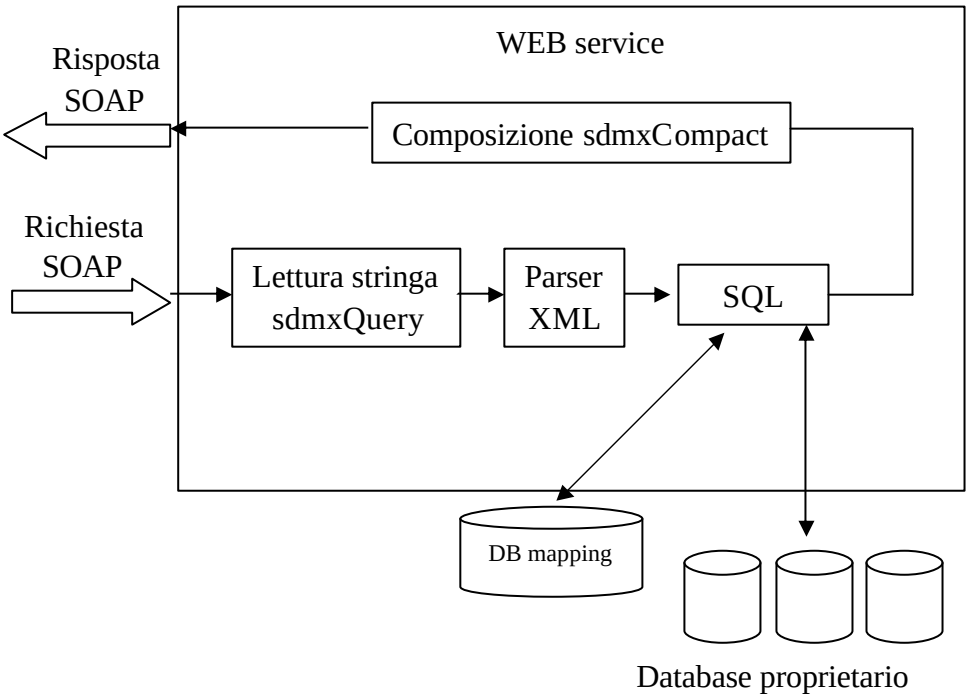


figura 3.13

Nel caso in cui si verifichi un errore, i WEB services sono in grado di inviare al client un messaggio con le specifiche di tale errore.

I WEB services sono stati protetti con lo scopo di renderli accessibili solo al repository. La protezione avviene sfruttando le utilità messe a disposizione da Windows e IIS (Internet

[27] Le tabelle di mapping, possono essere inserite direttamente nel database proprietario e permettono di creare i giusti legami tra la metainformazione presente sul Repository e quella presente sul Satellite.

[28] Per database proprietario si intende il database già presente sul Satellite prima dell'integrazione.

Information Server) definendo una userid e una password che il client deve fornire per poter accedere alle funzionalità messe a disposizione dai WEB services.

3.3.5.1 Il Satellite Conistat

L'interfacciamento del Satellite Conistat al Repository prevede un WEB service che permette di attuare il colloquio. Non sono state previste tabelle di mapping in quanto viene utilizzata direttamente la tabella "metadati" presente nel database proprietario. Ciascun record di tale tabella individua una specifica serie storica. In particolare il campo "n_dato" di ciascun record indica il nome della tabella in cui si trovano i dati per quella serie storica.

L'interpretazione della stringa sdmxQuery avviene attraverso un interprete (parser) XML realizzato attraverso un apposito oggetto. Tale oggetto permette di recuperare le seguenti informazioni:

Tag della stringa sdmxQuery	Informazioni utili nella realizzazione dell'SQL
`<Dimension Name="class" />`	specifica i codici delle serie storiche

`<DataFrom>`
`  <DataSet>metadati</DataSet>`
`</DataFrom>`
	specifica il nome della tabella in cui cercare la metainformazione [29]

`<Time>`
`  <StartTime>2003</StartTime>`
`  <EndTime>2003</EndTime>`
`</Time>`
	specifica l'intervallo temporale di riferimento dei dati da estrarre

`<DataSet>11</DataSet>`	specifica l'argomento statistico

Dalle informazioni ricavate il sistema è in grado di conoscere quali serie storiche sono state scelte dall'utente e quale è l'intervallo temporale di riferimento. Il sistema interroga, quindi, la tabella "metadati" per sapere quali sono le tabelle dei dati interessate.

A questo punto il sistema ha tutte le informazioni necessarie per estrarre i dati richiesti, formattarli secondo il formato sdmxCompact e inviarli al Repository come messaggio di risposta SOAP.

3.3.5.2 Il Satellite Demo

L'interfacciamento del Satellite Demo al Repository prevede, oltre al WEB service che permette di attuare il colloquio, una tabella di mapping rappresentata, in questo caso, dalla tabella "sdmx_metadati" all'interno del database proprietario. In particolare la tabella "sdmx_metadati" è simile alla tabella metadatiDEMO (vedi figura 3.12).

Il database proprietario in realtà è rappresentato da più database, in ognuno dei quali sono stati memorizzati i dati che si riferisce ad una argomento statistico differente. Esiste così il database per l'argomento "Popolazione residente", il database per l'argomento "Bilancio demografico", eccetera. All'interno di ciascun database i dati sono memorizzati in tabelle organizzate per anno. Per esempio

[29] Tale tag si è reso necessario perché la tabella della metainformazione potrebbe essere scomposta in più tabelle come specificato nel paragrafo 3.3.2

nel database che si riferisce all'argomento "Popolazione residente" esistono due tabelle "Prov2003" e "Prov2004" che si riferiscono ai dati provinciali per gli anni 2003 e 2004.
Segue la struttura di una tabella dati:

	Var 1	Var 2	Var 3			Var n
Territorio 1						
Territorio 2						
Territorio n						

figura 3.14

In pratica le tabelle in cui sono stati memorizzati i dati hanno un primo campo che memorizza il codice del territorio (provincia o regione) e gli altri campi che memorizzano le variabili (per esempio "Maschi celibi", "Femmine nubili", eccetera). Ciascun record rappresenta l' occorrenza delle variabili nello specifico territorio.
L'interpretazione della stringa sdmxQuery attraverso il parser XML, permette di recuperare le seguenti informazioni:

Tag della stringa sdmxQuery	Informazioni utili nella realizzazione dell'SQL
`<Dimension Name="class" />`	specifica i codici delle variabili

`<DataFrom>`
`  <DataSet>sdmx_metadati</DataSet>`
`</DataFrom>`
	specifica il nome della tabella in cui cercare la metainformazione[30]

`<Time>`
`  <StartTime>2003</StartTime>`
`  <EndTime>2003</EndTime>`
`</Time>`
	specifica il periodo di riferimento dei dati da estrarre

`<Dimension Name="disaterr">`
`  3`
`</Dimension>`
	specifica il livello di disaggregazione territoriale

`<Territory>`
`  <Dimension Name="terr">217</Dimension>`
`</Territory>`
	specifica i territori

`<DataSet>PRT</DataSet>`	specifica l'argomento statistico

Dalle informazioni ricavate il sistema è in grado di eseguire le opportune query SQL per recuperare i dati richiesti dall'utente.

[30] Tale tag si è reso necessario perché la tabella della metainformazione potrebbe essere scomposta in più tabelle come specificato nel paragrafo 3.3.2

3.4 Realizzazione del prototipo

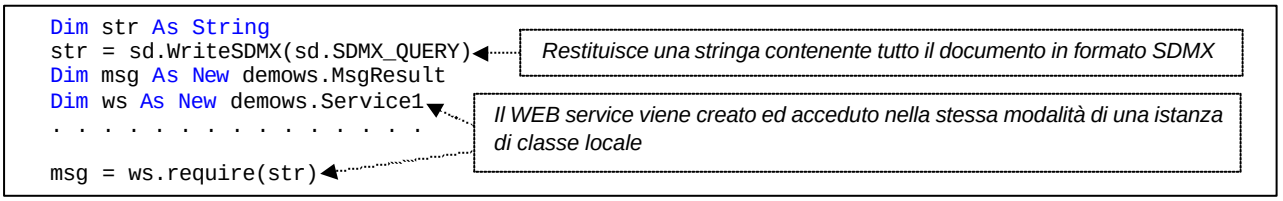
La necessità di realizzare il prototipo in breve tempo ha comportato una scelta tecnologica che tenesse conto dei seguenti fattori:

- utilizzo di un ambiente di sviluppo attraverso uno strumento IDE (Interface Development Environment) avanzato;
- conoscenze pregresse delle persone coinvolte nello sviluppo;
- funzionalità avanzate nella gestione del formato XML e della tecnologia WEB services.

La scelta è ricaduta su Microsoft ASP.NET che presenta alcune caratteristiche interessanti che si possono così riassumere:

- funzionalità avanzate di sviluppo in ambiente WEB attraverso l'uso di oggetti da utilizzare nell'interfaccia utente;
- gestione delle connessioni verso i database ottimizzate attraverso l'uso dell'oggetto dataset che lavorando in modalità disconnessa, garantisce la scalabilità del sistema realizzato;
- utilizzo del debug in ambiente WEB;
- gestione nativa di XML e WEB services.

In particolare tale ambiente di sviluppo maschera la complessità del protocolla SOAP in quanto, direttamente dall'IDE, è possibile leggere in automatico il WSDL di un WEB service ed ottenere le opportune routine software per accedervi. In pratica il programmatore utilizza un WEB service come se richiamasse una qualunque funzione passando anche dei parametri. Di seguito viene proposto un frammento di codice utilizzato nel modulo software posizionato sul Registry che avvia la richiesta al WEB service posizionato su un Satellite:



L'infrastruttura dei servizi WEB XML generati da .NET è totalmente conforme a standard quali SOAP, XML e WSDL. Pertanto tutti i client, indipendentemente dalla piattaforma, sono in grado di interoperare con i servizi WEB realizzati in tecnologia .NET.

In seguito vengono elencate le classi offerte dalla tecnologia .NET ed utilizzate per la realizzazione del prototipo:

Classe	Descrizione
`XmlTextWriter`	Fornisce la possibilità di creare un documento XML nodo per nodo in maniera serializzata scrivendo semplicemente i tag e il relativo contenuto verso uno stream di output.
`XmlTextReader`	Fornisce un accesso forward-only, read-only a flussi di tipo XML. Utilizza un puntatore che avanza ad ogni utilizzo dei metodi di lettura e delle proprietà che contengono il valore del nodo corrente. `XmlTextReader` è conforme allo standard W3C Extensible Markup Language (XML) 1.0.
`MemoryStream`	Crea flussi che utilizzano la memoria, anziché un disco o una connessione di rete, come archivio di backup
`Collection`	Rappresenta un modo pratico di fare riferimento a un gruppo correlato di elementi come se si trattasse di un unico oggetto. L'unica correlazione necessaria fra gli elementi, o membri, di un insieme, è la loro presenza nell'insieme. Non è necessario che condividano lo stesso tipo di dati. Per la creazione di un insieme è possibile seguire la stessa procedura utilizzata per la creazione di altri oggetti, Ad esempio: `Dim X As New Collection` Una volta creato un insieme, è possibile aggiungervi membri con il metodo `Add` oppure rimuoverli con il metodo `Remove`. Il metodo `Item` consente la restituzione di specifici membri dall'insieme, mentre l'istruzione `For Each...Next` consente l'iterazione in tutto l'insieme.
`XpathNodeIterator`	Fornisce un iteratore su un insieme di nodi selezionati.
`XPathNavigator`	Legge i dati di un qualsiasi documento XML utilizzando un modello a cursore. Fornisce un accesso ai dati casuale e di sola lettura. Il nodo corrente corrisponde al nodo sul quale è posizionato lo strumento di selezione. Lo strumento di selezione viene spostato in avanti utilizzando uno dei metodi di spostamento disponibili, mentre le proprietà riflettono il valore del nodo corrente.

In seguito vengono proposte alcune parti di codice che utilizzano le classi sopra esposte:

- accesso ad un nodo <*DataSelect*> di un documento di tipo sdmxQuery attraverso le classi `XpathNodeIterator – XpathNavigator;`

```
StreamXML = New IO.StringReader(str)
rdr = New XmlTextReader(StreamXML)
Dim myXPathDocument As XPathDocument = New XPathDocument(rdr)
Dim myXn As XPathNavigator = myXPathDocument.CreateNavigator()

'DataSelect
Dim Nav As XPathNodeIterator = myXn.Select("descendant::DataSelect")
```

- scrittura in uno stream XML attraverso la classe `xmlTextWriter`.

codice ASP.NET	stream XML
writer.WriteStartElement("Receiver")	<Receiver ID="12345">
writer.WriteAttributeString("ID", Me.ReceiverID) writer.WriteStartElement("Name") writer.WriteAttributeString("xml", "lang", "", Me.Lang) writer.WriteString(Me.ReceiverName) writer.WriteEndElement()	<Name xml:lang="en"> European Central Bank </Name>
writer.WriteStartElement("Contact")	<Contact>
writer.WriteStartElement("Name") writer.WriteAttributeString("xml", "lang", "", Me.Lang) writer.WriteString(Me.DepartmentReceiver) writer.WriteEndElement()	<Name xml:lang="en"> Statistic Division </Name>
writer.WriteStartElement("Department") writer.WriteAttributeString("xml", "lang", "", Me.Lang) writer.WriteString(Me.ContactReceiver) writer.WriteEndElement()	<Department xml:lang="en"> B.S.Featherstone </Department>
writer.WriteElementString("Telephone", Me.ContactTelReceiver)	<Telephone>+0039 </Telephone>
writer.WriteEndElement()	</Contact>
writer.WriteEndElement()	</Receiver>

Le classi per la gestione dei documenti XML rispettano i seguenti standard:

- XML 1.0 - http://www.w3.org/TR/1998/REC-xml-19980210 - incluso il supporto per DTD.
- XML Namespaces - http://www.w3.org/TR/REC-xml-names/ - sia stream level che DOM.
- XSD Schemas - http://www.w3.org/2001/XMLSchema
- XPath expressions - http://www.w3.org/TR/xpath
- XSLT transformations - http://www.w3.org/TR/xslt
- DOM Level 1 Core - http://www.w3.org/TR/REC-DOM-Level-1/
- DOM Level 2 Core - http://www.w3.org/TR/DOM-Level-2/

3.4.1 Classi generalizzate

Per la realizzazione del prototipo sono state create delle classi. Segue uno schema di un loro utilizzo per la produzione di un documento SDMX e una tabella in cui vengono spiegate le funzionalità.

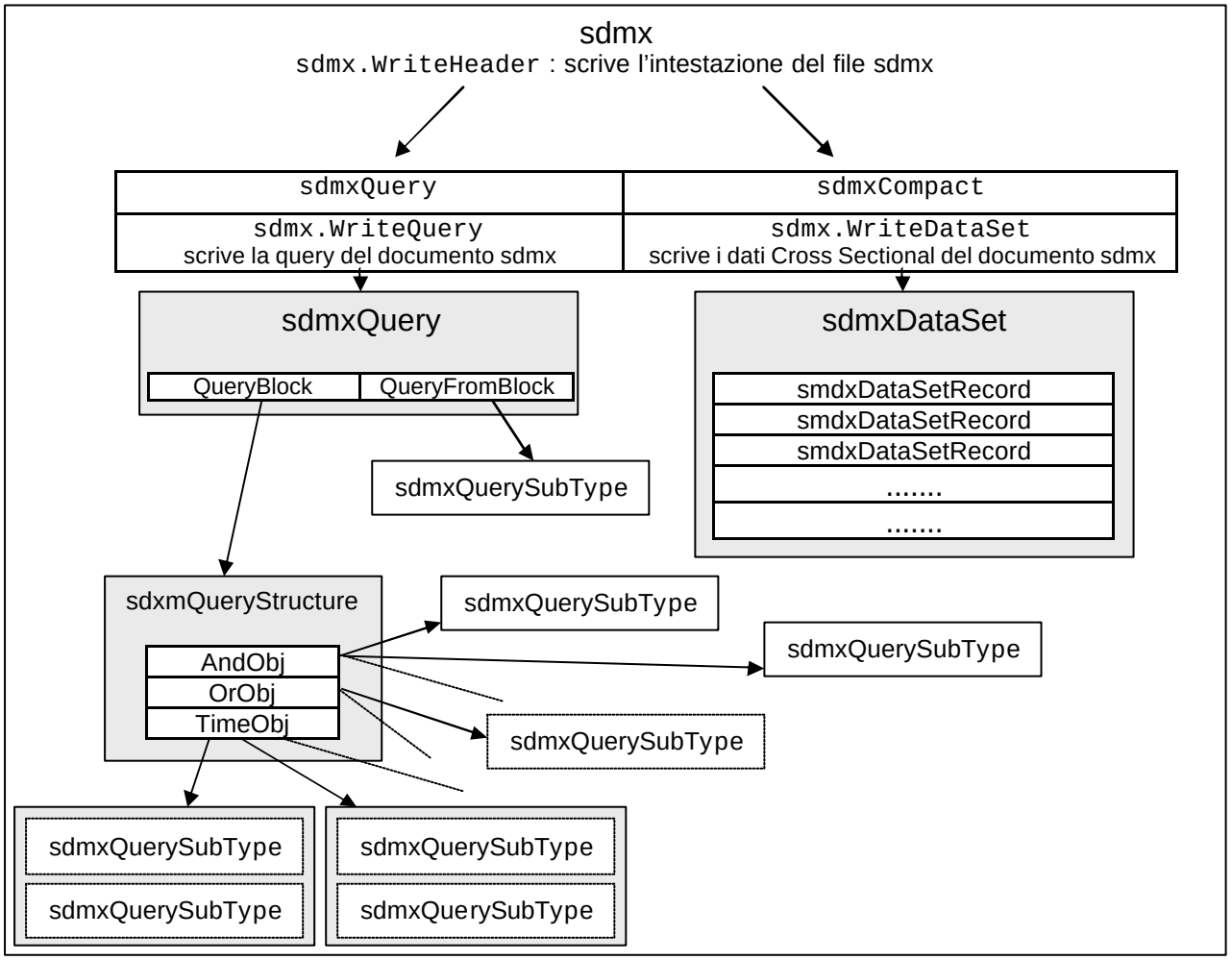


figura 3.15

Classe	Descrizione
`sdmx`	E' la classe principale della libreria. Permette la creazione del documento SDMX composto da una intestazione (header) e da un corpo (body). Il body varia in funzione del tipo di documento SDMX che si vuole produrre: sdmxQuery oppure sdmxCompact. L'istanza della classe avviene attraverso il costruttore *new* con il compito di inizializzare le variabili contenente le informazioni necessarie per la creazione dell'intestazione del documento SDMX e gli oggetti forniti da ASP.NET per scrivere nello stream XML (`XMLTextWriter`). Tale stream viene memorizzato in un oggetto `MemoryStream`. Per mettere assieme l'header e il body, viene utilizzato il metodo `sdmx.WriteSDMX(ByVal Tpe As Integer)` che a sua volta richiama i seguenti metodi: • `sdmx.WriteHeader(ByVal Tpe As Integer)` con il compito di scrivere nel `XMLTextWriter` gli attributi dell'intestazione; • `sdmx.WriteQuery(ByVal Tpe As Integer)` con il compito di scrivere nel body nel caso si stia creando un sdmxQuery; • `sdmx.WriteDataSet(ByVal Tpe As Integer)` con il compito di scrivere nel body nel caso si stia creando un sdmxCompact. Inoltre, nel caso dell'sdmxQuery, viene richiamato il metodo `WriteQueryChild(ByVal List As Collection)` che permette un'analisi sequenziale ed eventualmente ricorsiva dei nodi figli della struttura dati interessata.

`sdmxQuery`	Questa classe permette di creare un documento SDMX di tipo Query.
	Essa contiene le definizioni delle costanti che definiscono le varie tipologie di query che lo standard SDMX prevede.
	Utilizza due oggetti di tipo ***`sdmxQuerySubType`*** per comporre i blocchi di tipo *and/or* e *DataFrom*:
	`Public QueryBlock As sdmxQuerySubType`
	`Public QueryFromBlock As sdmxQuerySubType`

`sdmxQuerySubType`	La classe `sdmxQuerySubType` ha il compito di memorizzare ordinatamente i nodi che compongono i blocchi *and/or* e *DataFrom*. Supporta le seguenti tipologie di blocco: • DATASET • DIMENSION • AND • OR • TIME (TIME_START, TIME_END) • TERRITORY
`sdmxQueryParser`	Questa classe permette il recupero della query e la sua conversione dal formato SDMX alla struttura `sdmxQueryStructure` prossima al linguaggio SQL. Tale trasformazione utilizza il metodo `Translate(ByVal str As String) As sdmxQueryStructure`

<table>
<tr><td>sdmxQueryStructure</td><td>La sdmxQueryStructure è rappresentata dallo schema riportato in appendice B.
La conversione in SQL avviene associando gli elementi sdmxQueryStructure ai campi del database presente nel satellite.
Di seguito viene schematizzata la conversione da sdmxQueryStructure in SQL:
SELECT
<contenuto delle DIMENSION del blocco AND con denominazione “class”>
FROM
<nome tabella estratto da query sulla tabella metadati il cui nome viene contenuto nel DATASET del blocco DATAFROM>
WHERE
<contenuto delle DIMENSION del blocco TERRITORY>
AND
<contenuto delle DIMENSION del blocco OR>
AND
<contenuto delle DIMENSION del blocco AND con denominazione diversa da “class”>
AND
<campo del database con valore temporale>
BETWEEN
<contenuto delle DIMENSION del blocco TIME - STARTTIME>
AND
<contenuto delle DIMENSION del blocco TIME - ENDTIME></td></tr>
<tr><td>sdmxHashTable</td><td>Questa classe permette la creazione di collezioni di oggetti contenenti una relazione nome-valore.
A differenza dell’oggetto HashTable di ASP.NET non ordina gli oggetti per nome.
L’implementazione di tale classe è stata fondamentale per non perdere l’ordine con cui l’utente seleziona le dimensioni messe a disposizione nel Registry.</td></tr>
<tr><td>sdmxDataSet</td><td>La classe rappresenta la struttura dati necessaria a memorizzare le informazioni base tipiche del documento sdmxCompact.</td></tr>
<tr><td>sdmxDataSetRecord</td><td>Questa classe permette di accodare in sdmxHashtable i dati necessari e formare un documento sdmxCompact.
Nel seguente esempio viene messo in evidenza di utilizzo delle classi per la creazione del DataSet Cross Sectional e la corrispondente struttura gerarchica.
<table>
<tr><th>sorgente ASP.NET</th><th>stream XML</th></tr>
<tr><td><pre>SDataSet = New sdmxDataSet
SDataSet.Collection = "B"
SDataSet.TimeFormat = "P1Y"
SDataSet.Variable = “fcon”

s = New sdmxDataSetRecord
s.Name = "Obj"
s.AddElement("PROV",”1”)
s.AddElement("VALUE",”579590”)

SDataSet.AddData(s)</pre></td><td><pre><DataSet>
<Section COLLECTION="B"
TIME_FORMAT="P1Y"
VARIABLE="fcon">

<Obj PROV="1"
VALUE="579590"/>

  </Section>
</DataSet></pre></td></tr>
</table></td></tr>
<tr><td>sdmxGroup</td><td>Rappresenta le informazioni in caso di gruppo nei documenti sdmxCompact</td></tr>
</table>

3.4.2 Funzionamento

Segue, ora, un esempio di interazione utente-sistema con lo scopo di analizzare il colloquio fra i moduli del prototipo.

Si prenderà in esame il caso di dati di tipo Cross Sectional, più specificatamente i dati demografici per territorio.

Negli schemi presentati, per dare maggiore comprensione al testo, verranno presentati i nomi delle classi senza istanziarle.

3.4.2.1 Utente-Registry

Al momento dell'accesso, tramite il sito WEB istituzionale, l'utente accede ad una struttura ad albero attraverso la quale seleziona l'argomento statistico d'interesse. Il prototipo prevede al momento due tipologie di dati : Cross Sectional e Time series.

Vengono proposte le dimensioni che l'utente può selezionare:

dati demografici per territorio:

- periodo
- territorio
- sesso x stato civile (prodotto cartesiano fra gli insiemi delle due classificazioni)

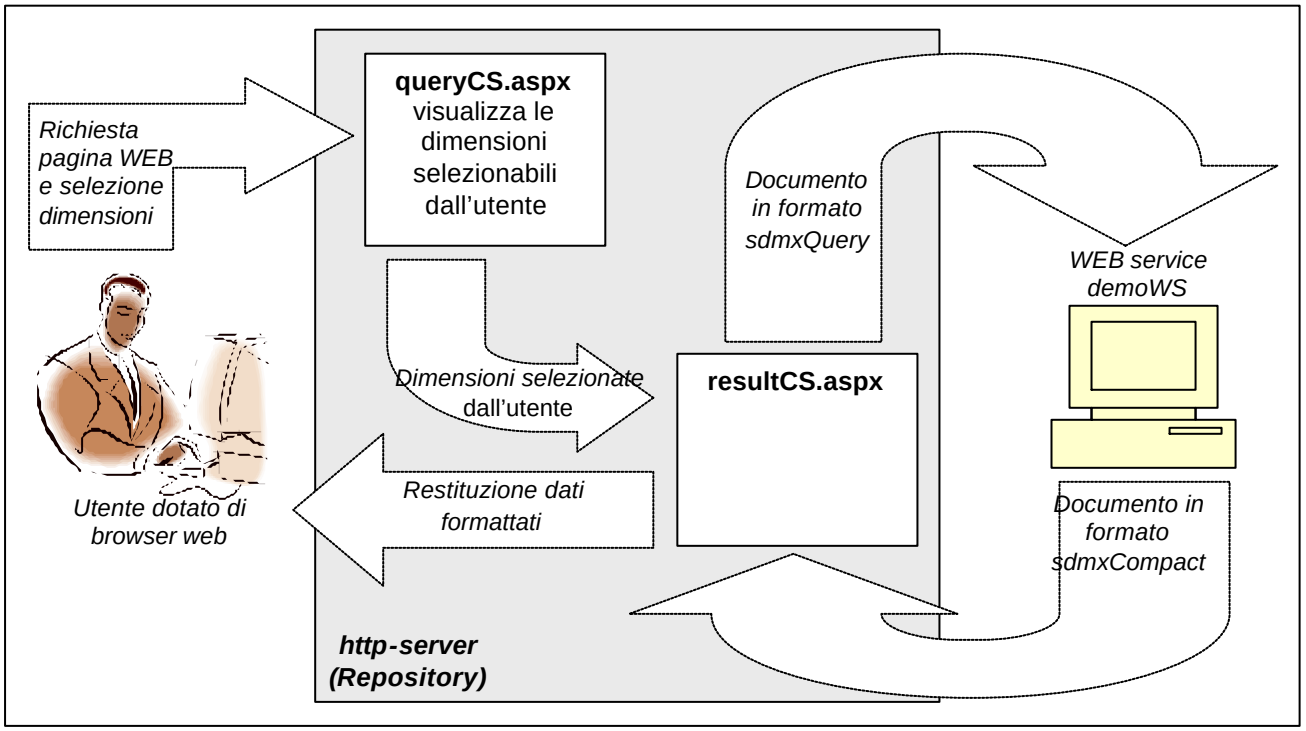


figura 3.16

La comunicazione fra due pagine WEB in genere avviene attraverso i metodi POST/GET del protocollo HTTP. Tale processo richiede però il passaggio attraverso un browser. Invece,

ASP.NET mette a disposizione il metodo `Server.Transfer` che permette di passare l'elaborazione direttamente tra pagine posizionate sullo stesso server, trasferendo anche dei parametri attraverso l'oggetto `Context`.

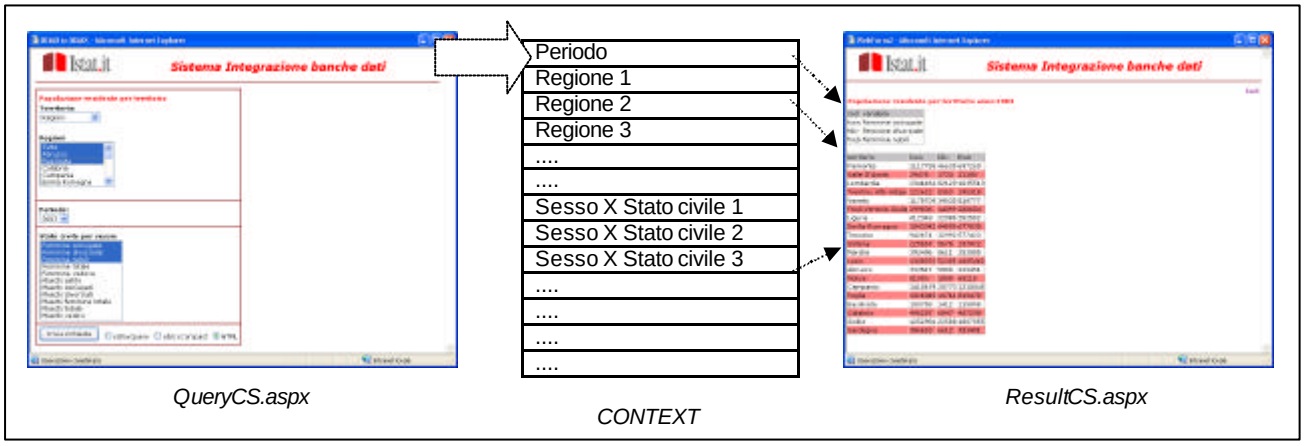


figura 3.17

Recuperati i parametri dal `Context` la pagina `ResultCS.aspx` attiva un'istanzia della classe `sdmx` per creare la struttura dati necessaria a comporre ed inviare la query formato SDMX.

Con il metodo

```
string = sdmx.WriteSDMX(sdmx.SDMX_QUERY)
```

viene restituita la stringa contenente il documento sdmxQuery che sarà passata al WEB service.

Il seguente schema è la rappresentazione secondo le strutture dati del prototipo del documento XML in formato SDMX presentato nel paragrafo 3.2.2.5.

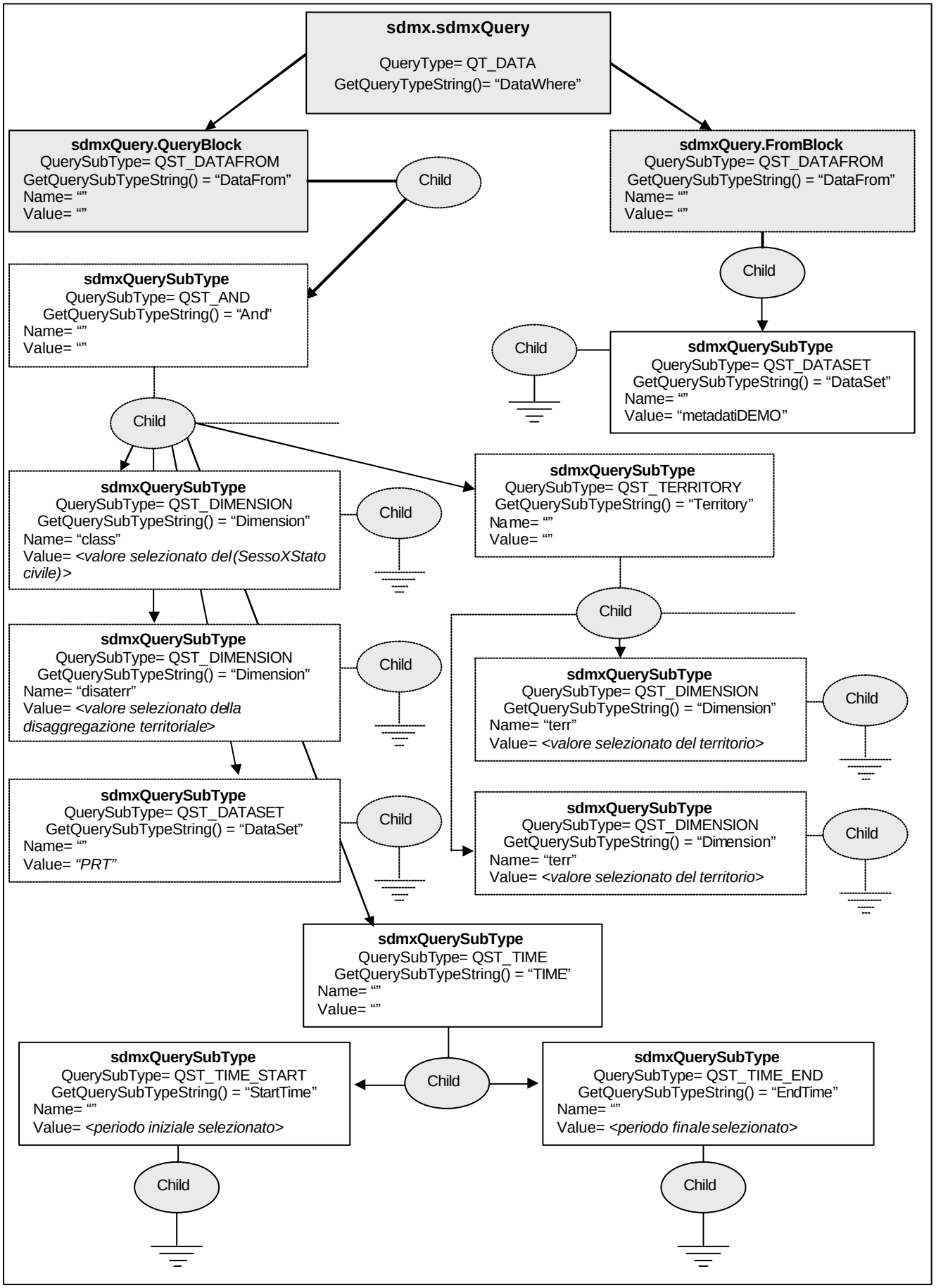

sdmx.sdmxQuery
QueryType= QT_DATA
GetQueryTypeString()= "DataWhere"
sdmxQuery.QueryBlock
QuerySubType= QST_DATAFROM
GetQuerySubTypeString() = "DataFrom"
Name= ""
Value= ""
Child
sdmxQuery.FromBlock
QuerySubType= QST_DATAFROM
GetQuerySubTypeString() = "DataFrom"
Name= ""
Value= ""
Child
sdmxQuerySubType
QuerySubType= QST_AND
GetQuerySubTypeString() = "And"
Name= ""
Value= ""
Child
Child
sdmxQuerySubType
QuerySubType= QST_DATASET
GetQuerySubTypeString() = "DataSet"
Name= ""
Value= "metadatiDEMO"
Child
sdmxQuerySubType
QuerySubType= QST_DIMENSION
GetQuerySubTypeString() = "Dimension"
Name= "class"
Value= <valore selezionato del (SessoXStato civile)>
Child
sdmxQuerySubType
QuerySubType= QST_TERRITORY
GetQuerySubTypeString() = "Territory"
Name= ""
Value= ""
Child
sdmxQuerySubType
QuerySubType= QST_DIMENSION
GetQuerySubTypeString() = "Dimension"
Name= "disaterr"
Value= <valore selezionato della disaggregazione territoriale>
Child
sdmxQuerySubType
QuerySubType= QST_DIMENSION
GetQuerySubTypeString() = "Dimension"
Name= "terr"
Value= <valore selezionato del territorio>
Child
sdmxQuerySubType
QuerySubType= QST_DATASET
GetQuerySubTypeString() = "DataSet"
Name= ""
Value= "PRT"
Child
sdmxQuerySubType
QuerySubType= QST_DIMENSION
GetQuerySubTypeString() = "Dimension"
Name= "terr"
Value= <valore selezionato del territorio>
Child
sdmxQuerySubType
QuerySubType= QST_TIME
GetQuerySubTypeString() = "TIME"
Name= ""
Value= ""
sdmxQuerySubType
QuerySubType= QST_TIME_START
GetQuerySubTypeString() = "StartTime"
Name= ""
Value= <periodo iniziale selezionato>
Child
Child
sdmxQuerySubType
QuerySubType= QST_TIME_END
GetQuerySubTypeString() = "EndTime"
Name= ""
Value= <periodo finale selezionato>
Child

figura 3.18

3.4.2.2 Registry-Satellite

Come precedentemente accennato ASP.NET rende trasparente la creazione del formato SOAP; il WEB service viene istanziato come un normale oggetto locale al quale viene passata la stringa contenente la query.

```
MsgResult= ws.require(string)
```

Per considerare eventuali codici e messaggi di errore il risultato viene restituito attraverso un'istanza della classe `MsgResult`.

```
Public Class MsgResult
    <XmlElementAttribute(ElementName:="ReturnCode")> Public ReturnCode As String
    <XmlElementAttribute(ElementName:="ErrMsg")> Public ErrMsg As String
    <XmlElementAttribute(ElementName:="Msg")> Public Msg As String
End Class
```

Il metodo `require` presente nel WEB service ha il compito di estrapolare il codice SQL dalla struttura SDMX, recuperare i dati dal proprio database, formattarli secondo lo standard Cross Sectional di SDMX v1.0 e ritornarli come stringa alla pagina `ResultCs.aspx`

3.4.2.3 Satellite

Ricevuta la stringa dal Registry, il WEB service demoWS attraverso l'utilizzo del metodo `sdmxQueryParser.Translate(string)` trasforma lo stream XML nella struttura dati `sdmxQueryStructure.` Tale struttura dati è rappresentata nella figura seguente:

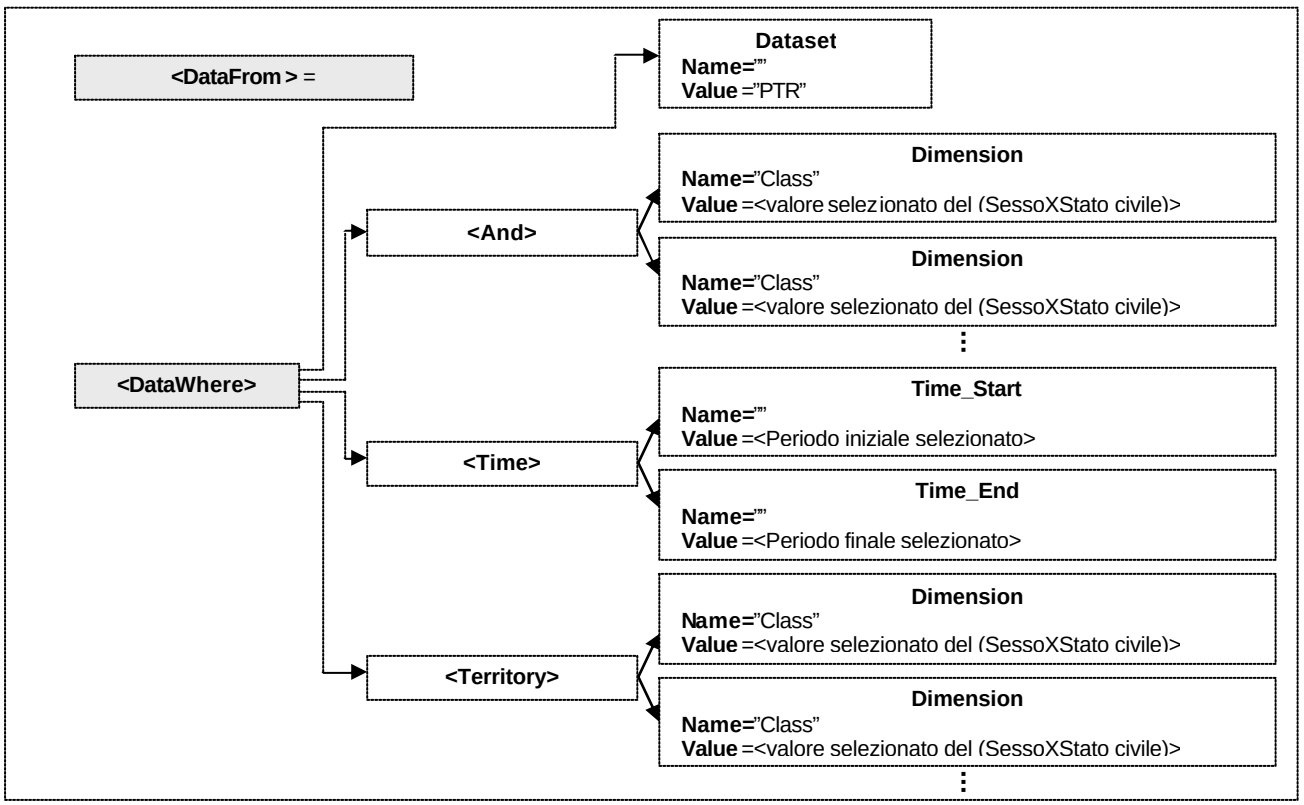


figura 3.19

Una parte specifica del WEB service è dedicata alla conversione della `sdmxQueryStructure` in linguaggio SQL.

Nel WEB service *demoWS* questo compito è effettuato dal componente `SQLConverter` e dalla funzione:

```
Public Function getSQL(ByVal Str As sdmxQueryStructure) As String
```

che effettuando la lettura della `sdmxQueryStructure` associa agli elementi della struttura dati le chiamate SQL, come illustrato precedentemente nel paragrafo riservato all'analisi di questa classe.

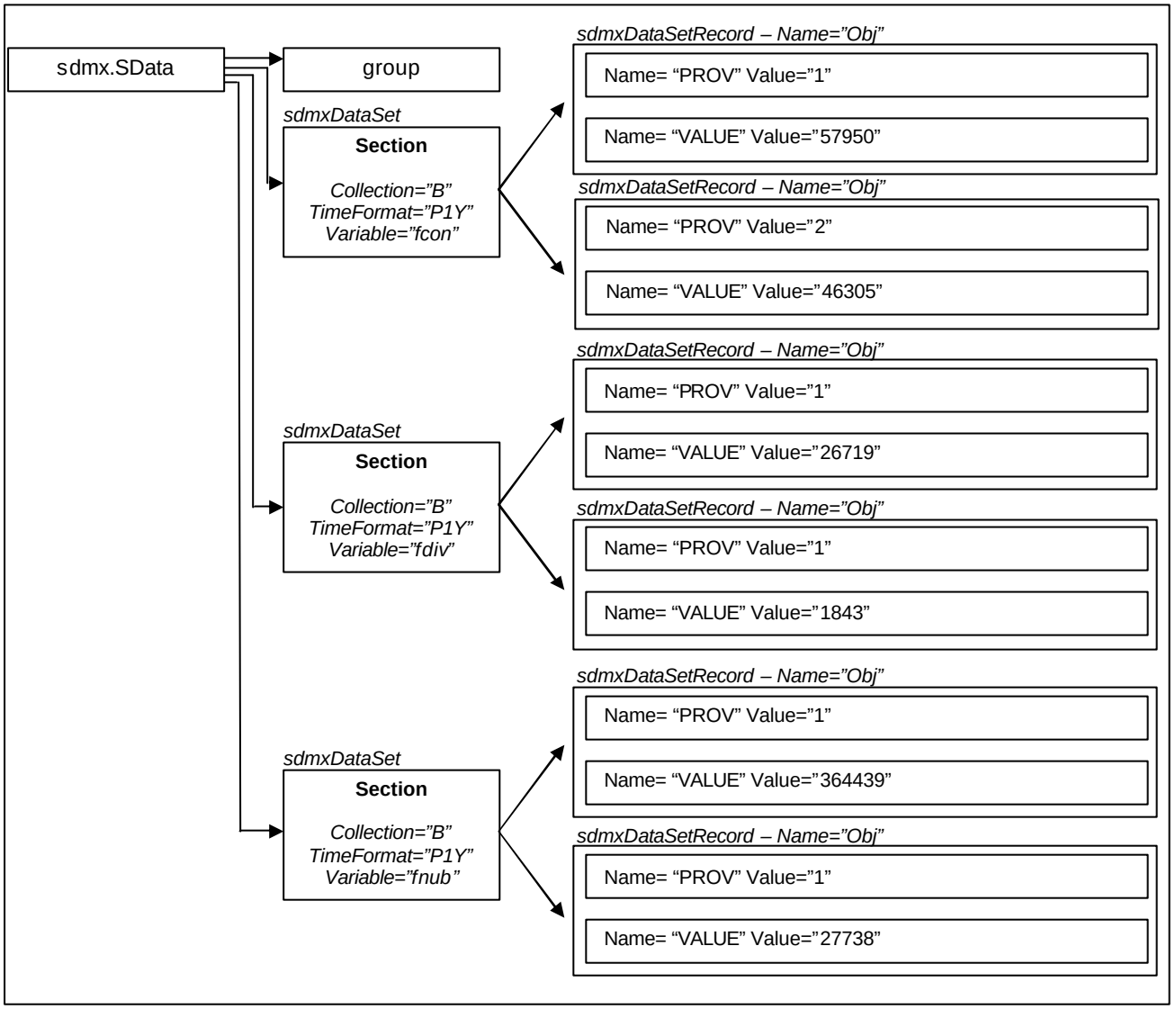
Ottenuta la query si procede all'estrazione dei dati e alla loro formattazione come documento Cross Sectional.

Nella seguente tabella si visualizza la corrispondenza dalle variabili selezionate alla stringa SQL.

Variabili selezionate	SQL
Territorio: Torino (1), Vercellli (2)	SELECT prov,fcon,fdiv,fnub FROM Prov2003 WHERE prov=2 OR prov=1
sesso X stato civile Sesso: Femmine Stato civile: Coniugate, Divorziate, Nubili	
Periodo: 2003	

I dati estratti attraverso la query SQL, vengono passati alle classi `sdmxDataSet` e `sdmxDataSetRecord` che provvedono alla creazione del documento sdmxCompact che viene restituito alla pagina `ResultCS.aspx` sul Registry.

Segue uno schema di composizione del documento sdmxCompact e il documento stesso.




```
<DataSet>
      <Group TIME="2003" UNIT="" UNIT_MULT="1" DECIMALS="2" AVAILABILITY="A" FREQ="A" VARIABLE="">
         <Section COLLECTION="B" TIME_FORMAT="P1Y" VARIABLE="fcon">
            <Obj PROV="1" VALUE="579590" />
            <Obj PROV="2" VALUE="46305" />
         </Section>
         <Section COLLECTION="B" TIME_FORMAT="P1Y" VARIABLE="fdiv">
            <Obj PROV="1" VALUE="26719" />
            <Obj PROV="2" VALUE="1843" />
         </Section>
         <Section COLLECTION="B" TIME_FORMAT="P1Y" VARIABLE="fnub">
            <Obj PROV="1" VALUE="364439" />
            <Obj PROV="2" VALUE="27738" />
         </Section>
      </Group>
   </DataSet>
```

3.4.2.4 Satellite-Registry

Ottenuto il risultato, richiesto al WEB services posizionato sul satellite, sotto forma di stringa (`MsgResult`), si controlla la sua correttezza valutando `MsgResult.ReturnCode` quindi si formattano i dati in formato HTML per la visualizzazione.

Nel caso in esempio (Cross Sectional – dati demografici per territorio) si utilizza per la visualizzazione un `DataGrid`.

Tale oggetto rende possibile la creazione di tabelle HTML attraverso una semplice gestione `cell[riga , colonna]` nel codice ASP.NET.

Una soluzione alternativa è stata proposta nel caso di dati time series in cui per la presentazione dei dati viene utilizzata una trasformazione XSL/XSLT direttamente dal documento sdmxCompact che il Satellite restituisce al Registry.

APPENDICE A

W3C XML 1.0 recommendation	http://www.w3.org/TR/REC-xml
Standard Generalized Markup Language (SGML)	http://xml.coverpages.org/sgml.html
XML FAQ (Frequently-Asked Questions)	http://www.ucc.ie:8080/cocoon/xmlfaq
XBRL (e**X**tensible **B**usiness **R**eporting **L**anguage)	http://www.xbrl.org/
FpML (industry-standard protocol for complex financial products)	http://www.fpml.org/
Namespaces in XML Recommendation	http://www.w3.org/TR/REC-xml-names/
SDMX Statistical Data and Metadata Exchange	http://www.sdmx.org
Applying XML and WEB services standards	http://www.xml.org
Resource site for XML technology and Web Services	http://www.webservices.org
XML Journal	http://www.sys-con.com/xml
Microsoft Corporation	http://www.microsoft.com
Microsoft WEB services developer center	http://msdn.microsoft.com/webservices/
Microsoft XML developer center	http://msdn.microsoft.com/xml/default.aspx
Java Technology and XML	http://java.sun.com/xml/index.jsp
Java Technology and WEB services	http://java.sun.com/webservices/index.jsp

Bibliografia

XML: Extensible Markup Language (Paperback) di Elliotte Rusty Harold

Learning XML, SecondaEdizione di Erik T. Ray

Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services -- di Thomas Erl

Programming .NET Web Services di Alex Ferrara, Matthew MacDonald

Implementor's Guide for SDMX Format Standards Autori vari

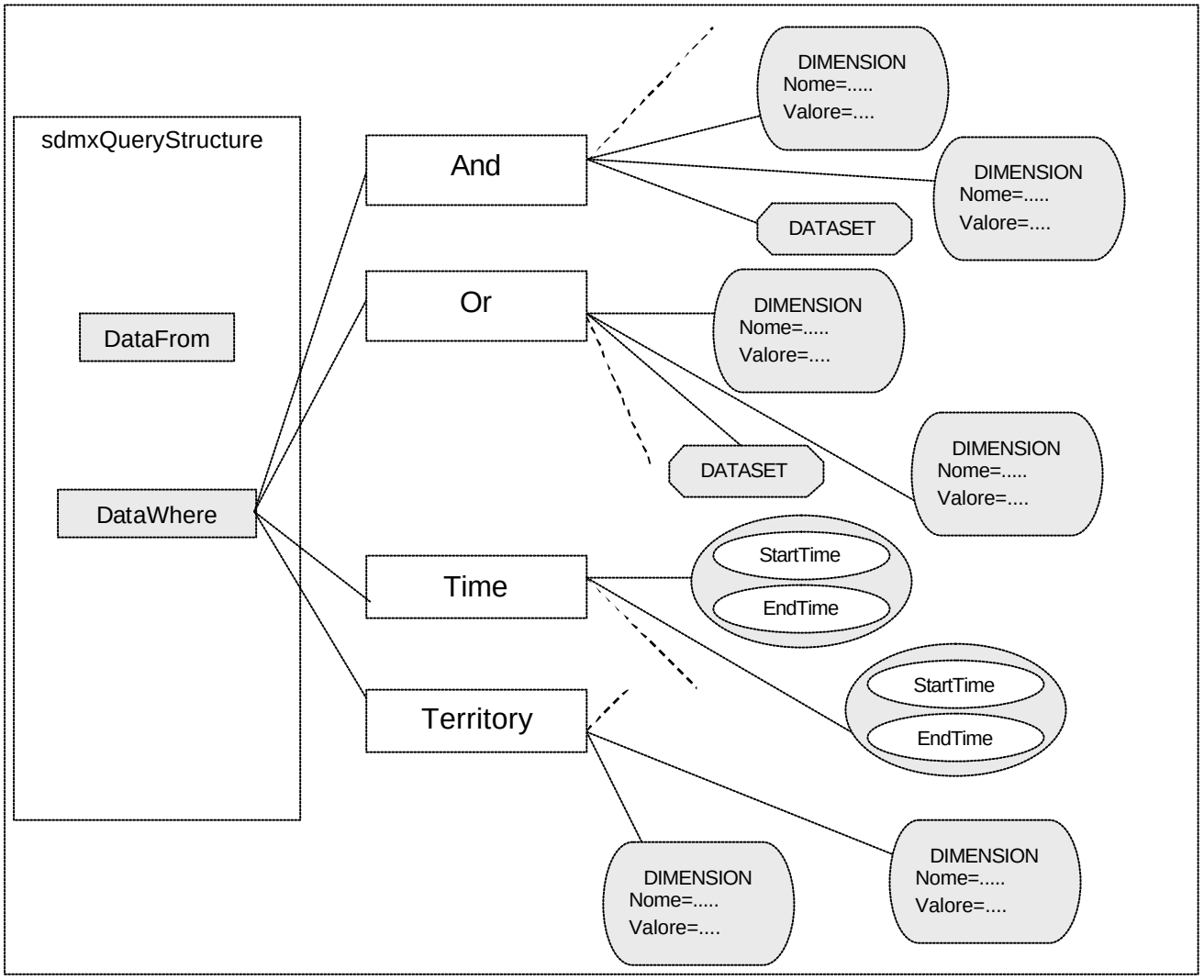
SDMX Information Model: UML Conceptual Design Autori vari

NET XML e Web Services. Full Contact di Pialorsi Paolo; Russo Marco

Creare XML Web Services. Con CD-ROM di Short Scott

APPENDICE B

Schematizzazione della struttura dati contenuta nella classe `sdmxQueryStructure`



APPENDICE C

Riepilogo dei campi e metodi delle classi utilizzate nel prototipo

Classe sdmx

Costanti pubbliche	
`Public Const SDMX_QUERY = 0`	Definisce la tipologia del formato nel caso di QUERY
`Public Const SDMX_CROSS_SECTIONAL = 1`	Definisce la tipologia del formato nel caso di CROSS SECTIONAL
Costanti private	
`Private Const intestazione_xml = "<?xml version=""1.0"" encoding=""UTF-8"" ?>`	Contiene l'intestazione standard XML con la specifica di *encoding*
`Private Const cross_sectional = "CrossSectionalData xmlns=""http://www.SDMX.org/resources/ ........`	Contiene l'header complete del file in caso di SDMX di tipo Cross Sectional
Campi pubblici	

`Public ID As String`
`Public Test As Boolean`
`Public Truncated As Boolean`
`Public Lang As String`
`Public TransName As String`
`Public Prepared As String`
`Public SenderID As String`
`Public SenderName As String`
`Public ContactSender As String`
`Public ContactTelSender As String`
`Public ReceiverID As String`
`Public ReceiverName As String`
`Public ContactReceiver As String`
`Public DepartmentReceiver As String`
`Public ContactTelReceiver As String`
`Public DataSetID As String`
`Public DataSetAction As String`
`Public DataSetExtractedData As String`
`Public ReportBegin As String`
`Public ReportEnd As String`
	Rappresentano le variabili contenenti le specifiche del *Sender* ed il *Receiver* dell'intestazione informativa del formato SDMX.

`Public SData As Collection`	Identifica la struttura gerarchica contenente il formato SDMX Compact
`Public SQuery As sdmxQuery`	Identifica la struttura gerarchica contenente la Query SDMX
Campi private	
`Private utf As New UTF8Encoding`	Rappresenta una codifica UTF-8 di caratteri Unicode
`Private ms As New MemoryStream`	`MemoryStream` sul quale viene "scritto" il file SDMX in formato XML per essere poi passato al WEB services
`Private writer As New XmlTextWriter(ms, Encoding.UTF8)`	Oggetto dedicato alla scrittura sul `MemoryStream` in formato XML

Metodi pubblici	
`Public Sub New()`	Costruttore, inizializza il `MemoryStream` e consequenzialmente il `XMLTextWriter`.
`Public Function WriteSDMX(ByVal Type As Integer) As String`	Metodo pubblico per la creazione del documento in formato SDMX restituito come stringa. In `Type` si indica la tipologia di file da creare (il prototipo supporta Query, CrossSectional e Time Series)
Metodi private	
`Private Sub WriteHeader(ByVal Type As Integer)`	Scrive l'header del formato SDMX nel `XMLTextWriter`, comprese le informazioni di intestazione (Sender, Receiver)
`Private Sub WriteDataSet()`	Scrive nel `XMLTextWriter` la struttura nel caso di file di tipo sdmxCompact
`Private Sub WriteQueryChild(ByVal List As Collection)`	Scompone e scrive nel `XMLTextWriter` gli eventuali nodi con figli della Query
`Private Sub WriteQuery()`	Scrive nel `XMLTextWriter` i nodi principali della Query. Nel caso il nodo abbia figli ne passa il contenuto a `WriteQueryChild`.

Classe sdmxQuery

Costanti pubbliche	

`Public Const QT_DATA = 1`
`Public Const QT_KEYFAMILY = 2`
`Public Const QT_CODELIST = 3`
`Public Const QT_CONCEPT = 4`
`Public Const QT_AGENCY = 5`
`Public Const QT_DATAFROM = 6`
	Indicano il tipo di query che si può effettuare in un documento SDMX secondo lo standard v 1.0. (il prototipo supporta con completezza la tipologia DATA)

Campi pubblici	
`Public QueryType As Integer`	Indica il tipo della query
`Public QueryBlock As sdmxQuerySubType`	Rappresenta la struttura gerarchica dei nodi XML contenente i dati relativi alla parte della query con operatori logici
`Public QueryFromBlock As sdmxQuerySubType`	Rappresenta la struttura della query relativa al nodo <DataFrom> proposto nella studio del prototipo

Metodi pubblici	
`Public Sub New()`	Costruttore, inizializza le istanze di classe dei campi pubblici
`Public Function GetQueryTypeString() As String`	Ritorna la stringa standard per la scrittura del nome del nodo relativo alla tipologia di query prescelta

Classe sdmxQuerySubType

Costanti pubbliche	

`Public Const QST_DATASET = 0`
`Public Const QST_KEYFAMILY = 1`
`Public Const QST_DIMENSION = 2`
`Public Const QST_ATTRIBUTE = 3`
`Public Const QST_CODELIST = 4`
`Public Const QST_TIME = 5`
`Public Const QST_CATEGORY = 6`
`Public Const QST_CONCEPT = 7`
`Public Const QST_AGENCY = 8`
`Public Const QST_OR = 9`
`Public Const QST_AND = 10`
`Public Const QST_TIME_START = 11`
`Public Const QST_TIME_END = 12`
`Public Const QST_TERRITORY = 13`
	Indicano il tipo di elemento che può comporre le sezioni della query SDMX secondo lo standard v 1.0. Lo studio del prototipo ha portato all'immissione del tag <Territory> per identificare il territorio nell tipologia di dati Cross Sectional.

Campi pubblici	
`Public QuerySubType As Integer`	Indica il tipo del nodo figlio della sezione <DataWhere> (nel caso di query DATA, l'unica supportata dal prototipo)
`Public Name As String`	Nome dell'attributo del nodo
`Public Value As Object`	Valore dell'attributo del nodo
`Public Child As Collection`	Classe contenente eventuali figli del nodo.

Metodi pubblici	
`Public Sub New()`	Costruttore, inizializza le istanze di classe ed i valori dei campi pubblici in base al numero di parametri passati Ritorna la stringa standard per la scrittura del nome del nodo relativo alla tipologia di query prescelta
`Public Sub New(ByVal tp As Integer, ByVal vl As Object)`	
`Public Sub New(ByVal tp As Integer, ByVal Nm As String, ByVal vl As Object)`	
`Public Function GetQuerySubTypeString() As String`	Ritorna la stringa standard per la scrittura del nome nel nodo filgio prescelto

Classe sdmxQueryParser

Campi privati	
`Private reader As Xml.XmlTextReader`	Identifica l' `XmlTextReader` che ha il copito di leggere i dati dal file XML contenente la query in formato SDMX
`Private FileName As String`	`MemoryStream` sul quale viene "scritto" il file SDMX in formato XML per essere poi passato al webservices

Metodi pubblici	
`Public Sub SetFileName(ByVal path As String)`	Costruttore, inizializza il `MemoryStream` e consequenzialmente il `XMLTextWriter`.
`Public Function GetFileName() As String`	Metodo pubblico per la creazione del documento in formato SDMX restituito come stringa. In *Type* si indica la tipologia di file da creare
Metodi privati	
`Private Function getInside(ByVal Nav As XPathNavigator, ByVal name As String, ByVal Attribute As Boolean) As sdmxHashTable`	Scrive l'header del formato SDMX nel `XMLTextWriter`, comprese le informazioni di intestazione (Sender, Receiver)
`Public Function Translate(ByVal str As String) As sdmxQueryStructure`	Scrive nel `XMLTextWriter` la struttura nel caso di file di tipo Cross Sectional

Classe sdmxQueryStructure

Sottoclassi pubbliche	
`Public Class sdmxSubQueryStructure`	Sottoclasse necessaria alla realizzazione della struttura gerarchica suddivisa per tipologia di operazione
Campi pubblici	
`Public AndObj As sdmxHashTable`	Rappresentano le strutture della query supportate dal prototipo
`Public OrObj As sdmxHashTable`	
`Public TimeObj As Hashtable`	
`Public Territory As sdmxHashTable`	
Metodi pubblici	
`Public Sub New()`	Costruttore, inizializza le istanze delle tre `sdmxHashTable`

Campi privati	
`Public FromObj As Collection`	Rappresenta la serie dei nodi `<DataFrom>` della query SDXM
`Public WhereObj As sdmxSubQueryStructure`	Indica la struttura contenuta nel nodo `<DataWhere>` della query SDMX

Metodi pubblici	
`Public Sub New()`	Costruttore, inizializza le istanze dei campi

Classe sdmxHashTable

Sottoclassi pubbliche	
`Public Class Hitem`	Classe necessaria ad identificare la relazione *nome-valore*
Campi pubblici	
`Public key As String`	Rappresenta il nome nella relazione *nome-valore*
`Public value As String`	Rappresenta il valore nella relazione *nome-valore*

Campi private	
`Private cl As Collection`	Contiene la collezione di istanze di *Hitem*

Campi pubblici	
`Public Sub New()`	Costruttore, inizializza l'istanza del campo privato di tipo `Collection`: *cl*
`Public Sub Add(ByVal Key As String, ByVal Value As String)`	Aggiunge un oggetto con nome *Key* e valore di tipo stringa *Value*
`Public Function getEnumerator() As IEnumerator`	Restituisce un `IEnumerator` per la visita completa, attraverso tale oggetto, alla colleazione
`Public Function getCount() As Integer`	Restituisce il numero di elementi presenti nella collezione
`Public Function getItem(ByVal index As Integer) As Hitem`	Restituisce l'`Hitem` contenuto nell' i-esima posizione della collezione

`Public Function getItem(ByVal key As String) As Hitem`	Restituisce l'*Hitem* con nome uguale alla stringa *Key*

Classe sdmxDataSet

Campi pubblici	
`Public Collection As String`	Informazioni base della tipologia di dato Cross Sectional secondo lo standard SDMX v1.0
`Public TimeFormat As String`	
`Public Variable As String`	
`Private Datas As Collection`	Contiene la collezione dei record di tipo Cross Sectional

Metodi pubblici	
`Public Sub New()`	Costuttore: inizializza le istanze dei campi pubblici
`Public Sub AddData(ByVal value As sdmxDataSetRecord)`	Aggiunge un record alla collezione di dati di tipo Cross Sectional
`Public Function GetData(ByVal index As Integer) As sdmxDataSetRecord`	Restituisce l' i-esimo record della collezione di dati
`Public Function GetCount() As Integer`	Restituisce il numero totale di record contenuti nella collezione

Classe sdmxDataSetRecord

Campi pubblici	
`Public Name As String`	Definisce il nome del nodo
`Private Records As Hashtable`	Contiene la collezione di elementi di tipo *nome-valore* componenti il dato Cross Sectional

Metodi pubblici	
`Public Sub New()`	Costruttore, inizializza le istanze dei campi pubblici
`Public Sub AddElement(ByVal key As String, ByVal value As String)`	Aggiunge un elemento al record richiedendone il nome (*key*) ed il valore (*value*)
`Public Function GetElementByKey(ByVal key As String) As String`	Retituisce l'elemento del record in base al nome *Key*
`Public Function GetElementByIndex(ByVal index As Integer) As String`	Restituisce l'i-esimo elemento del record
`Public Function GetDataEnumerator() As IDictionaryEnumerator`	Restituisce un `IDictionaryEnumerator` per la visita totale della collezione degli elementi del record
`Public Function GetCount() As Integer`	Restituisce il numero totale di elementi *nome-valore* componenti il record

Classe sdmxGroup

Campi pubblici	
`Public Unit As String`	Variabili delle informazioni relative ai gruppi come indicato nello standard v1.0 del protocollo SDMX
`Public Unit_Mult As String`	
`Public Decimals As String`	
`Public Availability As String`	
`Public Freq As String`	
`Public Terr As String`	
`Public Time As String`	
`Public Variable As String`	