UNIVERSITA' DEGLI STUDI DI MODENA E REGGIO EMILIA

FACOLTA' DI INGEGNERIA – SEDE DI MODENA

Corso di Diploma di Laurea in Ingegneria Informatica

Valutazione delle Tecnologie XML, Web Service per l'interoperabilità tra DBMS relazionali

Relatore:

Chiar.mo Prof.

Sonia Bergamaschi

Tesi di laurea di:

Yuri Debbi

Correlatori

Ing. Lorenzo Canali

Ing. Daniele Bergonzini

Parole Chiave

RDBMS Benchmark
SQL Server 2000
MySQL
XML
Web Service

Anno Accademico 2002/2003

INDICE

Ringraziamenti

Desidero ringraziare tutti i miei amici, la mia ragazza e soprattutto i miei genitori per il sostegno durante lo svolgimento di questa tesi.

Un ringraziamento speciale va a Sergio, il mio maestro di TAI CHI e massaggi SHIATSU che mi ha dato gli strumenti e l'appoggio necessari per potermi rilassare e concentrare anche nei momenti di maggiore difficoltà.

Inoltre desidero ringraziare per l'aiuto fornitomi durante lo svolgiemtno della tesi e per la disponibilità dimostrata: Giovanni Zuffolini, l'Ing. Daniele Bergonzini, l'Ing. Lorenzo Canali, l'ing. Maurizio Vincini e la Professoressa Sonia Bergamaschi.

1. Introduzione

1.1 Obbiettivi della Tesi

Lo scopo di questo studio è triplice: analizzare le differenze fra due modi estremamente differenti di concepire i DBMS relazionali (SQL Server 2000 e MySQL 4.0.12), l'utilizzo della tecnologia XML e XML Web Service per l'interoperabilità fra i vari DBMS relazionali e l'utilizzo e l'analisi della nuova tecnologia .NET soprattutto per quanto riguarda la sua interazione con strumenti esterni (RDBMS, XML Web Service).
Il punto di partenza sono applicazioni Webdatabase realizzate con la recente tecnologia ASP .NET che forniscono assistenza e servizi Web per reti civiche; ognuno di questi servizi si avvale di operazioni su database e manipolazione dei dati così ottenuti.

Organizzazione dei capitoli
Il secondo capitolo è una piccola introduzione all'ambiente di sviluppo .NET utilizzato per realizzare le applicazioni Web. In questo capitolo si vuole puntare l'attenzione sull'uso della tecnologia XML nell'implementazione stessa del Framework .NET e delle sue innovative caratteristiche orientate verso il Web.
Viene inoltre descritto il nuovo linguaggio C# fornito dalla Microsoft per lo sviluppo delle applicazioni come evoluzione naturale del C e del C++.

Il terzo capitolo si occupa dei DBMS relazionali. In questa sezione della tesi ci si occupa in primo luogo di capire a fondo le differenze di due scuole di pensiero completamente opposte per quanto riguarda lo sviluppo dei DBMS relazionali: SQL Server 2000 della Microsoft e MySQL 4.0.12 della MySQL AB.
Dopo aver analizzato le specifiche differenze fra le due piattaforme si passa alla valutazione dei differenti modi di utilizzo di SQL nell'interrogazione dei database dopo di ché vengono analizzati i metodi di connessione delle due piattaforme (con maggiore riguardo per il meno conosciuto MySQL) al Framework .NET.

Nel quarto capitolo ci si è occupati di XML definendone le specifiche e i modi di utilizzo, vengono inoltre ampliamente spiegati due dei principali strumenti che affiancano l'uso di XML nello sviluppo di applicazioni Web avanzate: XSD Schema (per il controllo dell'integrità dei dati e della formattazione degli stessi) e XPath (per l'interrogazione di un documento XML come albero di nodi).
Vengono poi valutate le diverse implementazioni nell'uso di XML, XSD Schema e XPath nel Framework .NET definendone così le prestazioni, i vantaggi e gli svantaggi.

Il quinto capitolo presenta una panoramica sul servizio XML Web Service analizzando gli strumenti necessari per creare ed utilizzare questa innovativa tecnologia: WSDL, SOAP, HTTP e UDDI.
Dopo aver fornito le basi per capire a pieno, sviluppare e pubblicare un XML Web Service, vengono indicati esempi specifici sia di realizzazione di questi servizi sia di utilizzo di servizi esterni pubblicati sul Web tramite UDDI.
Tutto questo sempre utilizzando gli innovativi strumenti forniti al Framework .NET che vede in queste tecnologie molte potenzialità per lo sviluppo futuro delle applicazioni Web.

Il sesto, settimo e ottavo capitolo forniscono una valutazione complessiva di queste problematiche affiancata da un glossario della terminologia tecnica utilizzata all'interno di questa tesi e la bibliografia che indica testi e siti di riferimento per questo studio.

Le appendici A, B e C contengogono tabelle e grafici che esaminano in dettaglio le problematiche descritte nei capitoli precedenti e immagini che hanno lo scopo di dare un idea anche visiva degli ambienti coi quali gli sviluppatori avranno a che fare durante la progettazione di applicazioni Webdatabase Avanzate.

1.2 Strumenti Utilizzati

Per realizzare questa ricerca sono stati utilizzati i seguenti strumenti:

Hardwere:

- Piattaforma come da requisiti Microsoft per progettare ed eseguire applicazioni Web .NET.

Softwere:

- Sistema operativo: Microsoft Windows XP Professional (SP 2) con IIS 5.1.
  Sono stati eseguiti, come richiesto dalle specifiche Microsoft per eseguire il Framework, tutti gli aggiornamenti disponibili sul sito della Microsoft.

- Microsoft Framework .NET 1.0 affiancato dai compilatori a riga di comando e dal Microsoft Data Access Components (MDAC) 2.7.

- I DBMS Relazionali utilizzati sono: Microsoft SQL Server 2000 e MySQL 4.0.12 della MySQL AB.

- Il linguaggio di Programmazione scelto per sviluppare le pagine ASP .NET e le applicazioni XML Web Service è il C#.

2. Panoramica di .NET

2.1 Intermediate Lenguage (IL) e Metadati

IL (Intermediate Language) è il formato del codice compilato della tecnologia .NET, tramite l'IL sia che si compili un file DLL ASP .NET scritto in COBOL o un file EXE Windows form scritto in C#, il risultato sarà sempre lo stesso linguaggio intermedio autodescrittivo. La semplice sintassi, basata su testo, dell'IL combinata coi metadati offre al runtime di .NET l'abilità di eseguire più operazioni con meno overhead e con un grande risparmio in termini di tempo.
Il runtime .NET offre allo sviluppatore opzioni di compilazione JIT (Just In Time) dell'applicazione in linguaggio macchina o la possibilità di precompilare ogni cosa nel codice macchina nativo in fase di installazione.
La tecnologia .NET precompila le applicazioni in linguaggio macchina nativo con l'aiuto dei metadati di un eseguibile. I Metadati sono documenti XML che descrivono la posizione e l'obbiettivo di ogni oggetto, proprietà, argomento, delegato e metodo nell'eseguibile di un applicazione .NET.
Il runtime usa i metadadti per eliminare l'overhead associato all'interpretazione dell'ignoto, pre convalidare l'accuratezza e l'obbiettivo di ogni funzione per evitare errori e ottimizzare la gestione della memoria.
I metadati non sono quindi altro che semplice codice XML che descrive i contenuti dell'eseguibile e fungono un po' da sommario per ogni applicazione .NET.
Si può accedere a queste informazioni anche utilizzando il disassembler ILDASM.EXE presente nel Framework .NET; questo strumento presenta i dettagli di alto livello dell'eseguibile in una gerarchia leggibile e organizzata che descrive i dettagli di ogni oggetto, metodo, argomento e proprietà nell'applicazione in questione.

2.2 C# (SHARP) e Tipi di Dati Orientati al Web 

C# (C SHARP) è stato introdotto dalla Microsoft per ovviare ai problemi legati ai lunghi cicli di sviluppo e alla complessità dei linguaggi C e C++.
C# è, come del resto lo sono C e C++, un linguaggio orientato all'uso degli oggetti che consente agli sviluppatori di software di creare rapidamente una vasta gamma di applicazioni per la nuova tecnologia .NET. Utilizzando i semplici costrutti del linguaggio C# è possibile perfino convertire automaticamente classi e costrutti in XML Web Service che possono essere richiamati da qualunque linguaggio in esecuzione su qualsiasi sistema operativo.
Oltre alle sue nuove potenzialità è importante ricordare che C# è progettato per non sacrificare niente della potenza e del controllo che hanno rappresentato, per oltre vent'anni, le caratteristiche principali di C e C++; grazie a questa eredità, C# offre un livello di somiglianza molto elevato con C e C++, fornendo una facile piattaforma di partenza per programmatori che si avvicinano a questo strumento.

3. RDBMS

(Relational DataBase Management System)

3.1 Panoramica su SQL Server 2000

SQL Server 7.0 si è dimostrato un DBMS Web e di commercio elettronico con elevati livelli di scalabilità e ha fornito alle aziende e agli sviluppatori di software alti livelli di disponibilità; SQL Server 2000 è un ulteriore passo avanti verso le nuove tecnologie Web.
I punti di forza di questo DBMS sono essenzialmente tre: interfaccia assolutamente user friendly, elevata sicurezza e utilizzo ad alto livello delle nuove tecnologie Web (XML).

SQL Server offre un interfaccia a tutte le funzioni del database intuitiva e sicura che guida anche gli utenti meno esperti nell'uso sicuro del database, numerosi strumenti quali: query analyzer, copia guidata di un database, servizi di trasformazione dati e funzionalità multilingue; rendono la progettazione e lo sviluppo tramite questo strumento assolutamente agevole sia per qualsiasi addetto del settore che per utenti meno esperti.

Oltre alle funzioni T-SQL incorporate nel prodotto, SQL Server 2000 rende possibile per l'utente creare proprie funzioni SQL tramite il comando CREATE FUNCTION; inoltre questa nuova versione include altre funzionalità di programmazione migliorate come: vincoli di integrità referenziali a catena, Trigger INSTEAD OF e AFTER (oltre ai comuni Trigger già implementati in SQL Server 7.0), indici per le colonne calcolate, nuovi tipi di dati (quali ad esempio i BIGINT) e regole di confronto a livello di colonna.

Inoltre SQL Server 2000 offre un tipo di supporto di XML integrato che risulta flessibile, ad alte prestazioni e semplice da utilizzare sia per gli sviluppatori Web, sia per i programmatori di database.
Le funzionalità XML disponibili in SQL Server 2000 facilitano la programmazione di database relazionali da parte di sviluppatori Web, in quanto consentono l'utilizzo di tecnologie quali XPath, le query URL e gli XML updategram; in modo analogo gli sviluppatori di database non sono costretti a imparare un linguaggio orientato agli oggetti o a comprendere tutti gli aspetti di XML.
Possono infatti adattare lo scenario di base garantendo l'accesso XML a un database relazionale esistente con la clausola FOR XML che restituisce dati in formato XML da un istruzione SELECT e la parola chiave T-SQL OPENXML.

Tramite SQL Server 2000 è quindi possibile interagire completamente con i dati XML, importando database in formato XML per poi lavorarci tramite SQL o lavorare in SQL per poi restituire dati in XML; questo rende SQL Server molto flessibile e facilita molto il lavoro di chi si occupa di database orientati ad applicazioni Web.

3.2 Panoramica su MySQL

MySQL è il più conosciuto e popolare Open Source SQL DBMS ed è sviluppato e distribuito dalla MySQL AB una compagnia formata dai fondatori di MySQL e dai principali sviluppatori originariamente avente sede in Svezia. Attualmente la compagnia è divenuta un organizzazione virtuale che raggruppa persone da dozzine di paesi che ogni giorno dialogano sulla rete con lo scopo di sviluppare e promuovere MySQL.
MySQL è un DBMS relazionale che supporta SQL, fornisce un database management system importante che segue lo sviluppatore nell'accesso e nella gestione dei dati; inoltre, cosa più importane, è Open Source.

Essere Open Source significa che è possibile per chiunque usarlo e modificarlo; chiunque può scaricare il software MySQL dalla rete e utilizzarlo senza dover pagare niente. Chiunque si senta in grado di farlo può studiare il sorgente di MySQL e modificarlo a suo piacimento per renderlo più performante e più ottimizzato per le proprie esigenze.
MySQL è completamente scritto e sviluppato tramite linguaggio C e C++ e verificato su vari compilatori in molteplici piattaforme; essendo progettato principalmente per sistemi Unix e Linux supporta molto bene CPU multiple, ma la sua caratteristica più importante è la velocità. Qualsiasi strumento sviluppato all'interno di MySQL è prima di tutto ottimizzato per essere il più reattivo e veloce possibile.

MySQL è quindi una valida soluzione per la gestione di un database, soprattutto per sviluppatori Web che necessitano di applicazioni veloci, pur essendo dotato di un ottima documentazione le caratteristiche implicite di MySQL (massima interoperabilità, massima standardizzazione e massima libertà di sviluppo) fanno sì che MySQL non sia intuitivo nel suo utilizzo e richieda un buona preparazione teorica per chi intende servirsene per la progettazione e lo sviluppo di applicazioni Web.

Fino alla versione 3.23.56 MySQL era comunque molto limitato per quanto riguarda le funzionalità e la gestione sicura delle transizioni che erano lasciate completamente a un controllo esterno (quasi sempre eseguito dalle applicazioni che utilizzavano il database), ma con l'uscita dell'attuale versione, MySQL 4.0.12 (2003) , è stato fatto un grande passo avanti. Sono state implementate tabelle (BDB e InnoDB) in grado di garantire transazioni sicure, il lock non è più a livello di tabella, ma diventa a livello di riga e finalmente anche MySQL (se pur per alcuni tipi di tabelle soltanto) può dirsi ACID Compliant.

Il Server MySQL fornisce tutto il necessario per lavorare su un database, ma se si vuole un ambiente di sviluppo più user friendly è necessario utilizzare un Client MySQL che fornisca un query analyzer e gli altri strumenti per velocizzare lo sviluppo.
Sono comunque disponibili alcuni Client sia gratuiti che proprietari che forniscono tutto il necessario per rendere agevole e più intuitivo il lavoro con MySQL; garantendo anche ottimi servizi di importazione ed esportazione dati, compresi dati XML estremamente utili nelle applicazioni Web.

3.3 Differenze generali fra SQL Server 2000 e MySQL 4.0.12

Sono dunque molti gli aspetti che è opportuno valutare quando una ditta, che si occupa di applicazioni Webdatabase (o comunque di applicazioni che richiedono l'uso di database), deve decidere quale Server scegliere e se sviluppare o meno applicazioni che supportino più di un DBMS. E' necessario verificare le differenze nei tipi di dati gestiti, nelle istruzioni SQL92 supportate e nelle eventuali istruzioni proprietarie, l'uso di driver pubblici o privati, l'uso delle funzioni proprietarie, i livelli di affidabilità, velocità e scalabilità, le interfacce e i limiti delle due soluzioni.

A questo scopo sono state riportate, nell'appendice A, una serie di tabelle e grafici che confrontano in modo sintetico e completo le due piattaforme SQL Server 2000 e MySQL 4.0.12 sviluppate secondo ideologie completamente opposte e concorrenti sul mercato; di seguito indicheremo quindi solo le caratteristiche più importanti che contraddistinguono questi due DBMS così diversi e rimandiamo all'appendice A, chi volesse informazioni più dettagliate.

La prima cosa da considerare è che SQL Server 2000 è un RDBMS commerciale che viene attualmente distribuito dalla Microsoft la quale fornisce una licenza Server (da installare sul Server che ospiterà fisicamente i database) a un costo di 830 Euro e un certo numero di licenze Client (da installare sulle postazioni che intendono lavorare sui database dislocati sul Server) che costano ognuna 182 Euro.

MySQL 4.0.12 è invece Open Source e offe vari tipi di licenza a seconda delle esigenze della ditta: GPL, LGPL o Commerciale.
MySQL 4.0.12 può essere scaricato direttamente dal sito ufficiale della MySQL AB (vedi bibliografia) e i prezzi per la ditta dipendono dall'uso che se ne intende fare.

- Completamente gratuito per la ricerca che non abbia scopo di lucro.

- Completamente gratuito se non viene modificato in alcun modo il sorgente; MySQL 4.0.12 può quindi essere utilizzato da una ditta che lavora nel campo del Webdatabase e venduto come parte integrante delle applicazioni Web (o usato come server dalla ditta per fornire un servizio di DBMS ai clienti) senza alcun costo aggiuntivo ne per la ditta ne per i clienti.

- Se invece la ditta che utilizza MySQL modifica il codice per adattare MySQL alle proprie esigenze o addirittura lo rende Embedded, cioè integrato con il proprio codice applicativo, allora deve per forza acquistare una licenza commerciale per ogni installazione (sul Server) di MySQL a un costo che varia da 440 Euro (meno di 10 licenze) per la versione PRO che contiene le tabelle InnoDB Transaction safe, a 220 Euro per la versione Classic che non prevede tabelle transaction safe.

- Può essere inoltre acquistato un contratto annuale di assistenza al costo di circa 1700 Euro.

Considerato quindi la differenza di costi che esiste fra le due piattaforme è necessario esaminare le principali differenze nelle prestazioni di questi due RDBMS.

Per quanto riguarda i tipi di dati entrambe le soluzioni prevedono la quasi totalità dei tipi di dato espressi nello standard SQL92, ma mentre MySQL prevede un tipo di dato BLOB, SQL Server 2000 esprime questa tipologia associando tipi di dato TEXT e tipi IMMAGE; SQL Server 2000 prevede inoltre tipi di dati MONEY specifici per la gestione delle valute.

Entrambe le tecnologie prevedono gli operatori relazionali standard SQL92 e MySQL aggiunge alla lista anche l'operatore Null Safe Equal (<=>) che permette uguaglianze fra un qualsiasi valore e il valore Null.

Sia SQL Server 2000 che MySQL implementano le funzioni aggregate (AVG, COUNT, MAX, MIN e SUM) e sempre entrambi supportano tutte le clausole HAVING e ORDER BY; esistono differenze però nell'implementazione dei vincoli di tabella da parte dei due DBMS: MySQL prevede infatti il vincolo ON DELETE/UPDATE SET NULL e ON UPDATE CASCADE che non sono supportati da SQL Server 2000, mentre entrambi non supportano l'istruzione ON DELETE/UPDATE SET DEFAULT.

E' inoltre importante ricordare che MySQL non supporta le espressioni di controllo CHECK sulle tabelle è prevede un discorso particolare per le chiavi primarie e le chiavi esterne.
MySQL, infatti, prevede un unica chiave primaria costituita da una unica colonna, nel caso si vogliano implementare chiavi primarie su più attributi si devono creare esplicitamente tramite un indice, che contenga tutte le chiavi, definito PRIMARY

Inoltre MySQL supporta da questa versione anche le FOREIGN KEY, per utilizzarle la procedura è molto più complessa di quella di SQL Server 2000.
Per prima cosa si possono usare le chiavi esterne solo su i tipi di tabella BDB e InnoDB, dopo di ché ogni chiave deve essere impostata esplicitamente dall'utente ed è molto difficile modificare le tabelle così ottenute dopo la creazione. MySQL vuole che le chiavi primarie e le loro chiavi importate siano identiche e non controlla chiavi con valori NULL.
La procedura di utilizzo delle chiavi esterne è quindi la seguente:

- Creare le tabelle Padre in cui si specifica la chiave primaria (che sarà ereditata).
- Creare le tabelle Figli con i campi ereditati identici a quelli del padre.
- Definire le chiavi primarie.
- Creare un indice per ogni chiave esterna del Padre e poi aggiungere le FOREIGN KEY.

Per quanto riguarda le istruzioni di CREATE e DROP entrambe le piattaforme non supportano la creazione di domini come prevista dallo standard SQL92 ed inoltre MySQL non supporta ne viste ne schemi rendendo più arduo il suo utilizzo da parte degli sviluppatori; MySQL introduce però costrutti quali: CREATE TABLE FROM SELECT e CREATE TABLE IF NOT EXIST che rendono quantomeno più agevole il lavoro dei programmatori MySQL che non dispongono della sintassi più completa di SQL Server 2000.

Mentre nella modifica delle tabelle (ALTER TABLE) MySQL è più flessibile di SQL Server 2000, per quanto riguarda le istruzioni di SELECT MySQL ha il grande svantaggio di non prevedere SELECT innestate; questo aspetto estremamente importante sarà comunque esaminato nello specifico nei capitoli successivi.

Pur prevedendo alcune possibilità in più nei raggruppamenti (GROUP BY), MySQL non prevede le istruzioni: ANY, ALL, NOT, SOME e NOT EXIST nelle funzioni WHERE e come vedremo in seguito questo complica notevolmente le interrogazioni ai database.

Mentre entrambe le soluzioni dispongono di tutte le funzioni standard SQL92 per INSERT, DELETE e UPDATE; alcune differenze emergono nell'uso del JOIN: MySQL prevede il NATURAL JOIN mentre SQL Server 2000 non lo implementa, rendendo però disponibile agli sviluppatori un potente FULL OUTER JOIN.

Per quanto riguarda le funzioni SQL92 e le funzioni proprietarie, MySQL 4.0.12 e SQL Server 2000 dispongono, seppur con qualche differenza, delle stesse capacità espressive mettendo a disposizione dei programmatori un elevato numero di funzioni per la manipolazione e la gestione dei dati.

Ricordiamo inoltre un piccolo particolare che se trascurato potrebbe dare molti fastidi ai programmatori: MySQL gestisce i dati Boolean come TINYINT(1) quindi i confronti con questi tipi di dati sono eseguiti tramite "1" e "0" e non tramite "TRUE" e "FALSE" come in SQL Server 2000.

Se per quanto riguarda MySQL si prendono in considerazione le tabelle Transaction Safe BDB e InnoDB, allora entrambe le piattaforme RDBMS supportano i quattro fondamentali livelli di isolamento (READ-COMMITTED, READ-UNCOMMITTED, REPEATABLE-READ e SERIALIZABLE) ed entrambi sono ACID Compliant; MySQL però non prevede l'uso di Trigger e Stored Procedure.

Per qunto riguarda la velocità dei due DBMS si è voluto testare le due piattaforme all'interno di una pagina ASP .NET che esegue una connessione al DBMS ed esegue un istruzione SQL.

La forza di MySQL è sicuramente l'inseriemnto dati, MySQL risuta infatti essere fino al 57% più veloce di SQL Server nell'inseriemnto dei record in una tabella; nella modifica dei record MySQL è più veloce dell'88% rispetto a SQL Server fino alla soglia dei 25000 Record modificati, dopo di chè i ruoli dei due DBMS si scambiano e SQL Server diventa fino al 37% più veloce (il divario si assottiglia sempre più aumentando la mole di dati da modificare).

Nella selezione da una singola tabella i due DBMS hanno prestazioni quasi identiche fino alla soglia dei 25000 record selezionati; a questo punto MySQL subisce un drastico rallentamento che lo rende fino al 86% più lento rispetto a SQL Server.

Se si eseguono selezioni su un Join di più tabelle si ripete il caso precedente ma questa volta se ci si spinge fino alla seconda solia di 150000 record selezionati anche SQL Server subisce un brusco rallentamento e superato questo punto i due DBMS ritornano a prestazioni molto simili (MySQL risulta più lento dell'8%).

Questo confronto tra le rispettive velocità dei due DBMS è approfondito in dettaglio nell'appendice A.

In fine è necessario ricordare che se pur entrambi i DBMS sopportano le più comuni interfacce a programmi (C, C++, C#, ODBC, JDBC, XML, PHP, PERL e altri) essendo SQL Server 2000 una tecnologia Microsoft non supporta le piattaforme Unix o Linux.

Tipi di Tabelle Gestite da MySQL

E' necessario chiarire maggiormente la definizione dei tipi di tabelle disponibili per il programmatore che progetta e utilizza un database realizzato tramite MySQL 4.0.12.
Quando ci si appresta all'analisi di un databse con MySQL si ha infatti la possibilità di scegliere, per ogni tabella utilizzata, il tipo adatto alle nostre esigenze; fondamentalmente le tabelle MySQL possono essere divise in due grandi famiglie: Transaction Safe (TST) e Non Transaction Safe (NTST).

Le tabelle TST definite dai tipi BDB e InnoDB offrono una grande sicurezza: in caso di crash del sistema è infatti sempre possibile recuperare i prori dati utilizzando i backup e il transaction log, tramite queste tabelle è possibile combinare molte istruzioni all'interno di strutture complesse che potranno essere eseguite tramite un comando di COMMIT e sarà sempre possibile un ROLLBACK in caso si vogliano ignorare i cambiamenti fatti.
In questo modo se un operazione dovesse essere abortita per qualche ragione sarebbe possibile risalire alla situazione stabile precedente.

Le tabelle NTST definite dai tipi ISAM, MyISAM, MERGE e HEAP non dispongono di nessuna delle precedenti strutture di controllo, ma non avendo nessun overhead per le transazioni sono estremamente più veloci e occupano meno spazio sul disco fisso; inoltre l'impegno della memoria nelle operazioni di modifica e inserimento è molto minore rispetto alle tabelle TST.

La scelta del tipo di tabella è quindi lasciata allo sviluppatore (se il tipo non viene indicato MySQL definisce la tabella come MyISAM) che dovrà decidere in base alle proprie esigenze tenendo conto delle particolari caratteristiche di ogni tipo di tabella.

Tabelle ISAM:

- Strutturate tramite B-Tree.
- Gli indici sono tenuti in un file .ISM.
- I dati sono archiviati in un file .ISD.
- Possono essere controllate o riparate tramite il tool ISAMCHK.
- Utilizzando Chiavi compresse di lunghezza fissa.
- Prevedono lunghezze sia fisse che dinamiche dei record.
- Prevedono al massimo 16 Chiavi con 16 part/key.
- La lunghezza massima di una chiave è di 256 caratteri.
- I Dati sono archiviati in formato macchina.
- Non è possibile creare tabelle maggiori di 4 GB.
- Tabelle dinamiche risultano maggiormente frammentate.

Tabelle MyISAM:

- Tabelle di Default per MySQL, definiscono un tipo migliorato delle tabelle ISAM.
- Gli indici sono tenuti in un file .MYI.
- I dati sono archiviati in un file .MYD.
- Possono essere controllate o riparate tramite il tool ISAMCHK.
- Possono essere ulteriormente compresse tramite il tool MYISAMPACK.
- Un Flag è compreso nelle tabelle per definire se è avvenuta una corretta chiusura.
- Implementano un concurrent insert che utilizza tutti gli eventuali blocchi liberi per un inserimento.
- I Dati sono archiviati in modo indipendente dal sistema operativo.
- Tutte le chiavi numeriche hanno il byte maggiore come primo byte aumentando così la loro compressione.
- Possono essere indicizzati anche campi TEXT o BLOB.
- Valori Null sono permessi come indici di colonna.
- Le chiavi possono essere lunghe al massimo 500 bytes.

Tabelle MERGE:

- Sono una collezione di tabelle MyISAM che possono essere utilizzate come una sola tabella.
- Sono utilizzati per gestire più semplicemente dati contenuti in diverse parti del disco fisso o in dischi differenti (anche con tecnologie RAID).
- Rendono più veloci le ricerche su più tabelle che vengono trattate come una tabella sola.
- Permettono riparazioni più semplici.
- Permettono il mapping di più file in uno solo.
- Rendono più veloci operazioni che richiederebbero un uso pesante dei JOIN.
- Aggirano il problema del limite per la grandezza dei file del sistema operativo.

Tabelle HEAP:

- Caricano tutto l'indice in memoria rendendo queste tabelle molto veloci.
- In caso di rottura del sistema tutti i dati vengono irrimediabilmente persi.
- Hanno una lunghezza fissa dei record.
- Non supportano tipi di dato TEXT o BLOB.
- Non dispongono di autoincrementi.
- Non prevedono l'uso di Null in un indice.
- Si può avere una chiave non unica.
- Sono accessibili a tutti i Client.

Tabelle BDB (BerkeleyDB):

- Per una completa analisi di queste tabelle è possibile visitare il sito http://www.sleepycat.com.
- Prevedono la gestione sicura delle Transazioni.
- Prevedono l'uso dei comandi COMMIT e ROLLBACK.
- Prevedono l'uso di FOREIGN KEY.
- Si basano su una tecnologia XML.

Tabelle InnoDB:

- Per una completa analisi di queste tabelle è possibile visitare il sito http://www.innodb.com.
- ACID Compliant.
- Prevedono la gestione sicura delle Transazioni.
- Prevedono l'uso dei comandi COMMIT e ROLLBACK.
- Prevedono un transaction log per il recupero dei dati in caso di rottura del sistema.
- Prevedono un Lock a livello di riga.
- Prevedono l'uso di FOREIGN KEY.
- Ottimizzate per grandi quantità di dati, è possibile archiviare tabelle fino a 1 TB.

3.4 Differenze nell'uso di SQL tra SQL Server 2000 e MySQL 4.0.12

A livello di progettazione e sviluppo di applicazioni che si basano su RDBMS, che necessitano quindi di una fonte di dati accessibile tramite il linguaggio SQL, la differenza veramente importante fra SQL Server 2000 e MySQL 4.0.12 è che quest'ultimo non supporta i seguenti costrutti SQL92:

- SELECT innestate.
- IN e NOT IN con parametro una SELECT.
- EXIST e NOT EXIST.

Questa mancanza rende l'interrogazione del database molto meno intuitiva, più prolissa e molto più difficile per un progettista che deve quindi avere ottime conoscenze di algebra relazionale per implementare (tramite SELECT semplici, tabelle temporanee e i JOIN messi a disposizione da MySQL), costrutti di interrogazioni ormai di uso comune sia a livello accademico che negli ambienti lavorativi.
Verranno quindi proposti esempi di interrogazioni SQL per dimostrare le differenze nell'uso di SQL Server 2000 e MySQL.

Esempio 1
Interrogazioni scritte tramite SQL Server 2000
Tramite questa interrogazione viene espressa l'operazione algebrica di differenza:
(table1 –table2).

```
SELECT *
FROM table1
WHERE id NOT IN (SELECT id FROM table2);
```

```
SELECT *
FROM table1
WHERE NOT EXISTS (
                    SELECT id FROM table2
                      WHERE table1.id=table2.id);
```

Stessa interrogazione per MySQL 4.0.12

```
SELECT table1.*
FROM table1
      LEFT JOIN table2
      ON table1.id=table2.id
      WHERE table2.id IS NULL;
```

Esempio 2
Interrogazione scritte tramite SQL Server 2000
Tramite questa interrogazione viene espresso un multi-join a 4 vie.

```
SELECT      Tab1.ID, Tab1.servizi, Tab2.ID, Tab2.utilizzo, Tab1.descrizione, Tab1.link,
            Tab1.colore, Tab3.utenti, Tab3.gruppi
FROM        Tab1,Tab2,Tab3
WHERE       Tab2.ID=Tab3.utilizzo AND Tab3.servizi=Tab1.ID
AND         Tab1.valoresistema <> 1
AND         Tab3.multiportal=IDPortaleTab3
AND         ( Tab3.utenti=IDTab3
            OR Tab3.gruppi IN
                        (SELECT     gruppi
                         FROM       Tab4
                         WHERE      utenti=ID AND multiportal=IDPortale)
             )
AND         (Tab3.utenti=IDTab3 OR Tab3.gruppi = 3)
ORDER BY    Tab3.multiportal,Tab3.utenti DESC,Tab3.servizi;
```

Stessa interrogazione per MySQL 4.0.12
Creo una tabella temporanea (HEAP) adita a contenere i risultati della select che era innestata.

```
CREATE TABLE tabtemp (ID tinyint(3)) TYPE=HEAP
```

Eseguo la select innestata e metto gli eventuali risultati nella tabella temporanea.

```
INSERT INTO tabtemp  SELECT  DISTINCT Tab3.gruppi
                             FROM Tab3
                             LEFT JOIN Tab4 ON Tab3.gruppi=Tab4.gruppi
                             WHERE Tab4.gruppi IS NOT NULL
                             AND utenti=ID AND multiportal=IDPortale
```

Eseguo la select identica aggiungendo la tabella temporanea e controllo se il contenuto della tabella temporanea è nullo o meno.

```
SELECT      Tab1.ID,Tab1.servizi,Tab2.ID,Tab2.utilizzo,Tab1.descrizione,Tab1.link,
            Tab1.colore,Tab3.utenti,Tab3.gruppi
FROM        Tab1,Tab2,Tab3,tabtemp
WHERE       Tab2.ID=Tab3.utilizzo AND Tab3.servizi=Tab1.ID
AND         Tab1.valoresistema <> 1
AND         Tab3.multiportal=IDPortaleTab3
AND         (Tab3.utenti=IDTab3 OR tabtemp.id IS NOT NULL)
AND         (Tab3.utenti=IDTab3 OR Tab3.gruppi = 3)
ORDER BY    Tab3.multiportal,Tab3.utenti DESC,Tab3.servizi;
```

Cancello la tabella temporanea

```
DROP TABLE tabtemp
```

Esempio 3
Interrogazioni scritte tramite SQL Server 2000
Questa interrogazione esprime l'operazione di divisione: seleziona tutte le istanze di Tab1 che contengono tutte le istanze di Tab3 dove il campo CC è uguale al campo CC della tabella Tab2 e il campo CD di Tab2 è uguale a D1.
Questa interrogazione può essere riferita a tre tabelle: studenti, professori e loro corsi ed esami sostenuti.
Selezionare tutti gli studenti che hanno sostenuto tutti gli esami relativi ai corsi del docente D1.

```
SELECT *
FROM Tab1
WHERE NOT EXISTS
                (       SELECT *
                        FROM Tab2
                        WHERE CD="D1"
                        AND NOT EXISTS
                                        (       SELECT *
                                                FROM Tab3
                                                 WHERE Tab3.Matr=Tab1.Matr
                                                AND Tab3.CC=Tab2.CC))
```

Relativa interrogazione per MySQL 4.0.12
Creo una tabella temporanea e la riempio con i dati presi da una SELECT che implementa il costrutto EXIST.

```
CREATE TABLE temptab ( )TYPE=HEAP
SELECT Tab1.matr
FROM Tab1,Tab2
LEFT JOIN Tab3 ON Tab3.matr=Tab1.matr
                AND Tab3.CC=Tab2.CC
                WHERE Tab2.CD='D1'
                AND Tab3.matr IS NULL
                AND Tab3.CC IS NULL;
```

Eseguo una select facendo la differenza fra gli studenti che non hanno dato tutti gli esami di un professore e il totale egli studenti ottenendo il risultato voluto.

```
SELECT *
FROM Tab1
LEFT JOIN temptab ON Tab1.MATR=temptab.matr
WHERE temptab.matr IS NULL;
```

Elimino la tabella temporanea.

```
DROP TABLE temptab;
```

3.5 Differenze di interfaccia e uso con .NET tra SQL Server 2000 e MySQL 4.0.12

3.5.1 Corrispondenza dati Fra MySQL e .NET

Quando ci si appresta ad utilizzare MySQL Server 4.0.12 integrandolo in un applicazione Webdatabase sviluppata tramite il Framework .NET la prima cosa da controllare è la consistenza dei dati scambiati fra le due tecnologie.
Il Framework è standardizzato per utilizzare come database Server SQL Server 2000 quindi dispone di oggetti costruiti in modo da supportare i dati secondo la logica SQL Server 2000, questo comporta qualche difficoltà quando si vuole sostituire la piattaforma che si occupa dei dati.

Avendo scelto ODBC (Open Database Connectivity) come driver di connessione fra il Framework .NET e MySQL si deve controllare cosa succede ai dati quando vengono creati tramite il Framework e passati tramite ODBC al database implementato su MySQL.

Il Driver ODBC prevede un suo set di tipi di dato aspettandosi una precisa sintassi e dei precisi range, ma quando deve immettere i dati nelle tabelle MySQL lo fa tramite dei CAST adattando quindi il dato alla tabella precedentemente costruita secondo le specifiche di chi ha costruito e sviluppato il database tramite un Client MySQL.

Questo procedimento fa si che si possa "ingannare" il driver ODBC inserendo tipi di dati più "grandi", che includano cioè nelle loro regole sintattiche e nei loro range quelli dei dati realmente voluti; una volta inseriti nelle tabelle i dati saranno convertiti dal CAST e diverranno quindi conformi ai valori voluti.

Se ad esempio voglio inserire un valore decimal, per prima cosa si deve creare una colonna in una tabella MySQL dichiarandola di tipo decimal, dopo di che si immetterà tramite il driver ODBC un dato di tipo real (dello stesso formato del decimal ma più capiente).

Quando tramite una Funzione del Framework il valore viene inserito nella tabella il dato subirà un CAST che lo trasformerà in un dato di tipo decimal.

La seguente tabella descrive la corrispondenza dei dati tra Framework .NET, ODBC e MySQL.

TIPI DI .NET	TIPI DI ODBC	TIPI DI MySQL	NOTE
BIGINT	BigInt	BIGINT	
BINARY	Binary	CHAR	(Va con Char e opzione Binary su MySQL)
BIT	Bit	TINYINT(1)	
CHAR	Char	CHAR(n)	
DATE	Date	DATE	
DATETIME	DateTime	DATETIME	
DECIMAL	Decimal	DECIMAL	(Va fatto con Real)
DOUBLE	Double	DOUBLE	
IMMAGE	Immage	NON ESISTE	
INT	Int	INT	
NCHAR	NChar	CHAR	
NTEXT	NText	TEXT	
NUMERIC	Numeric	DECIMAL	(Va fatto con Real)
NVARCHAR	NVarChar	VARCHAR	
REAL	Real	DOUBLE	
SMALLDATETYPE	SmallDateTime	DATETIME	
SMALLINT	SmallInt	SMALLINT	
TEXT	Text	TEXT	
TIME	Time	TIME	(Va con DateTime)
TIMESTAMP	Timestamp	TIMESTAMP	
TINYINT	TinyInt	TINYINT	(Va con Int o con Unisigned)
UNIQUEIDENTIFIER	UniqueIdentifier	NON ESISTE	
VARBINARY	VarBinary	VARCHAR	(Va con VarChar e opzione Binary su MySQL)
VARCHAR	VarChar	VARCHAR	

Esempio di dati immessi in un Database MySQL tramite il metodo del Framework .NET: command.Parameters

Questo esempio illustra come è possibile passare alcuni parametri impostati da una funzione .NET a una tabella che si trova su un database impostato tramite MySQL utilizzando come driver di trasporto ODBC.

Queste procedure di trasferimento dati saranno meglio specificate nei capitoli successivi, ora si vuole solo dare un idea di base sulle operazioni riguardanti il trasferimento dati tra applicazioni sviluppate tramite il Framework .NET e database MySQL.

```
// Creo la stringa di inserimento
string command_testo = "insert into MiaTab (datainserimento, richiesta"
                       +",nomeCognome, azienda, email, telefono,
                       indirizzo,località, provincia, testo,numerico"
                       +") values ( ";
                       decimal prova=Convert.ToDecimal("1,1");

                       command_testo +="?"; //data di inserimento
                       command.Parameters.Add(new OdbcParameter("@datainserimento",
                        OdbcType.Date, 4, "datainserimento")).Value = DateTime.Now;
                       command.Parameters.Add(new OdbcParameter("@numeric",
                        OdbcType.Double));

                       command.Parameters["@numeric"].Value = -23.435;
                       command.Parameters["@numeric"].Precision = 2;
                       command.Parameters["@numeric"].Scale = 3;

                       ……
                       ……

                       command_testo += ",?";
                       command_testo += ");";
```

3.5.2 Connessione al Framework .NET di MySQL

3.5.2.1 Connessione Tramite ODBC

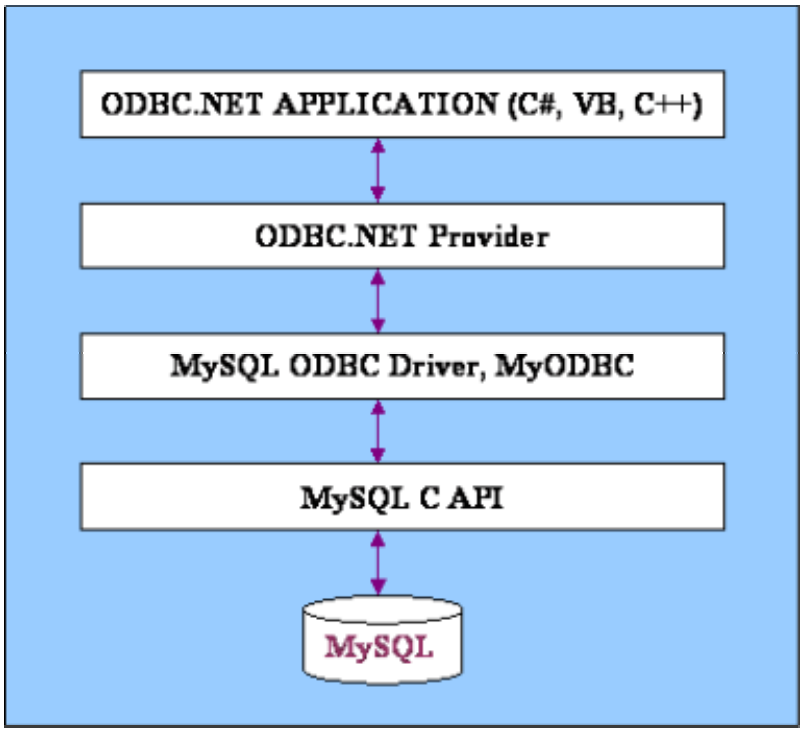
Essendo stati sviluppati come componenti della tecnologia .NET, SQL Server 2000 e il Framework .NET sono interamente compatibili tramite qualsivoglia soluzione (ODBC, OLEDB, Provider Nativi) per quanto riguarda MySQL questa interoperabilità con il Framework non è così immediata e comporta alcuni, limiti e alcuni accorgimenti.

MySQL supporta totalmente ODBC (Open Database Connectivity), come del resto il Framework .NET; partiremo quindi da questa piattaforma comune per sperimentare l'interoperabilità fra queste due tecnologie.

E' necessario quindi installare i seguenti componenti sulla piattaforma che si intende utilizzare per sviluppare applicazioni .NET che facciano riferimento a un database MySQL:

- Si deve installare .NET Framework SDK 1.0
- Si deve installare Microsoft Data Access Components (MDAC) 2.7
- Si deve installare ODBC .NET Provider
- Questo provider ha alcuni bachi che provocano degli errori quando si utilizzano stringhe vuote in determinate operazioni; è necessario quindi installare anche la patch per questi problemi che andrà applicata al file ODBC32.DLL e può essere recuperata dal sito: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q319243.
- Installare MySQL Server 4.0.12
- Installare il Driver ODBC per MySQL: Driver-MyODBC 3.51

Una volta installati tutti i componenti i driver ODBC faranno da “ponte” per il trasporto dei dati fra le applicazioni sviluppate tramite le classi e gli oggetti del Framework e il database MySQL.



Capitolo 3, Figura 1: Architettura Odbc – MyOdbc

Per usare ODBC .NET Data Provider, si deve importare il Namespace “Microsoft.Data.Odbc” nell'applicazione.

```
using Microsoft.Data.Odbc;
```

Si devono quindi dichiarare gli oggetti che si intende usare in modo che seguano la sintassi prevista da ODBC.

```
protected OdbcConnection cn;
...
...
protected OdbcCommand command;
```

Sarà quindi necessario creare una stringa di connessione che possa utilizzare il driver ODBC per connettersi al server MySQL; questa stringa dovrà quindi indicare: il driver utilizzato, l’indirizzo del server sul quale è fisicamente dislocato il database MySQL, il nome del Database a cui ci si vuole connettere, lo user name dell’utente che vuole connettersi e la sua password.

Una volta creata la stringa di connessione, la si utilizzerà per aprire e chiudere la connessione ogni volta che sarà necessario.

```
StringaDiConnessione = new OdbcConnection ("DRIVER={MySQL ODBC 3.51
                        driver};SERVER=localhost;DATABASE=MioDB;UID=MioUtente;
                        PASSWORD=MiaPassword;");

    cn.Open();
    command = new OdbcCommand("",StringaDiConnessione);
    ……
    ……
    cn.Close();
```

Rispetto alla connessione tramite OLDB standard per SQL Server 2000, ODBC offre allo sviluppatore le stesse classi tranne le corrispettive di : OleDBSchemaGuid e OleDBLiteral.
Il Namespace "Microsoft.Data.Odbc" fornisce funzioni per connettersi a una fonte di dati, eseguire comandi e acquisire i risultati. I valori ottenuti potranno essere processati direttamente o essere inseriti in un DataSet ADO .NET per essere poi processati una volta disconnesso il database.

Una volta all'interno del DataSet i dati potranno essere visualizzati, combinati con altri dati provenienti da varie fonti o trasferiti ad altre funzioni o applicazioni remote.

Classi	Descrizione
OdbcCommand	Rappresenta un'istruzione SQL o una stored procedure da eseguire in relazione a un'origine dati.
OdbcCommandBuilder	Fornisce un metodo per la generazione automatica di comandi di tabella singola per risolvere le modifiche apportate a un oggetto DataSet con il database associato. La classe non può essere ereditata.
OdbcConnection	Rappresenta una connessione aperta a un'origine dati.
OdbcDataAdapter	Rappresenta un set di comandi di dati e una connessione a un database utilizzati per riempire l'oggetto DataSet e aggiornare l'origine dati.
OdbcDataReader	Fornisce un modo per leggere un flusso di righe di dati forward-only da un'origine dati. La classe non può essere ereditata.
OdbcError	Raccoglie informazioni importanti relative a un avviso o a un errore restituito dall'origine dati. La classe non può essere ereditata.
OdbcErrorCollection	Raccoglie tutti gli errori generati dal provider di dati ODBC .NET. La classe non può essere ereditata.
OdbcException	Eccezione generata quando il provider sottostante restituisce un avviso o un errore per un'origine dati ODBC. La classe non può essere ereditata.
OdbcInfoMessageEventArgs	Fornisce i dati per l'evento InfoMessage. La classe non può essere ereditata.

Classi	Descrizione
OdbcParameter	Rappresenta un parametro di un oggetto OleDbCommand, ed eventualmente il relativo mapping a una colonna di DataSet. La classe non può essere ereditata.
OdbcParameterCollection	Raccoglie tutti i parametri rilevanti di un oggetto OleDbCommand e i rispettivi mapping alle colonne dell'oggetto DataSet.
OdbcPermission	Fornisce a un provider di dati OLE DB .NET la funzionalità per assicurare che un utente disponga di un livello di protezione adeguato per l'accesso a un'origine dati ODBC.
OdbcPermissionAttribute	Associa un'azione di protezione a un attributo di protezione personalizzato.
OdbcRowUpdatedEventArgs	Fornisce i dati per l'evento RowUpdated.
OdbcRowUpdatingEventArgs	Fornisce i dati per l'evento RowUpdating.
OdbcTransaction	Rappresenta una transazione SQL da effettuare in corrispondenza di un'origine dati. La classe non può essere ereditata.

Delegati	Descrizione
OdbcInfoMessageEventHandler	Rappresenta il metodo che gestirà l'evento InfoMessage di un oggetto OdbcConnection.
OdbcRowUpdatedEventHandler	Rappresenta il metodo che gestirà l'evento RowUpdated di un oggetto OdbcDataAdapter.
OdbcRowUpdatingEventHandler	Rappresenta il metodo che gestirà l'evento RowUpdating di un oggetto OdbcDataAdapter.

Enumerazioni	Descrizione
OdbcType	Specifica il tipo di dati di un campo, una proprietà o un oggetto OdbcParameter.

Una volta scritto il codice che descrive l'applicazione .NET, per compilare adeguatamente i file di progetto (.cs) si deve aggiungere nel file batch (.bat) che si occupa della compilazione una istruzione che contenga il riferimento ai driver ODBC.

```
@echo off
csc /debug+ /debug:full /out:..\..\bin\MiaDLL.dll /t:library /r:System.dll
/r:System.Data.dll  /r:System.Web.dll /r:Microsoft.Data.Odbc.dll
 FileDiConfigurazione.cs
File1.cs database_File1.cs
```

Dopo aver editato un file batch come sopra, si dovrà: aprire un promt comandi, entrare nella directory dove si trovano i file da compilare ed eseguire il file per la compilazione. Si otterrà quindi il file eseguibile che rappresenta l'applicazione perfettamente integrato coi driver ODBC e quindi con MySQL.

A questo punto per visualizzare in locale la pagina ASP .NET in modo che venga interpretata dal compilatore JIT correttamente, si deve creare una directory virtuale tramite l'IIS (Internet Information Service) dopo di che lanciare la pagina ASP .NET dal localhost.

Una volta descritte le modalità da seguire nella programmazione verranno ora sperimentate le istruzioni più comuni che si utilizzano per interagire con un database in modo da verificare un corretto funzionamento del server MySQL; in questo modo verranno anche fornite informazioni sulla sintassi delle interrogazioni SQL che variano a volte da quelle per SQL Server 2000, e verranno anche esposti esempi di interrogazioni al database MySQL tramite codice C# scritto grazie al Framework .NET.

SPERIMENTAZIONE DELL'ISTRUZIONE INSERT CON ODBC PER .NET

Tipica espressione di INSERT in MySQL:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
        VALUES ((expression | DEFAULT),...),(...),...

        or
          INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
         SELECT ...

        or
          INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name
        SET col_name=(expression | DEFAULT), ...
```

Tipico esempio:
INSERT INTO MiaTabella (ID,Nome,Cognome) VALUES(2,'Mario','Rossi');

Esempio di INSERT in un file .cs:

```
//  CREO LA STRINGA DI INSERIMENTO SEGUENDO LA SINTASSI SQL
string command_testo = "insert into "+conf.TABELLA_CONTATTI+"
                        (datainserimento, richiesta"
                        +",nomeCognome, azienda, email, telefono, indirizzo,località,
                        provincia, testo" values ( ";

command_testo +="?"; //data di inserimento
command.Parameters.Add(new OdbcParameter("@datainserimento", OdbcType.Date, 4,
                                "datainserimento")).Value = DateTime.Now;

// ESEGUO I CONTROLLI SUI DATI INSERITI
if (richiesta!="" && richiesta!=null )
        {
        command_testo +=",'"+richiesta+"'"; // IDSezione
        }
else
        {
        command_testo +=",NULL"; // IDSezione
        }
if (nomeCognome!="" && nomeCognome!=null )
        {
        command_testo +=",'"+nomeCognome+"'"; // IDSezione
        }
……
……
// COSTRUTTO DI CONTROLLO DELLA TRANSAZIONE SE RIESCE FAI,
// ALTRIMENTI FAI..
try
        {
        //INSERISCO USANDO LA STRINGA DI INSERIMENTO
        command.CommandText=command_testo;
        command.ExecuteNonQuery();
        myTrans.Commit();
        return true;
        }
catch (Exception ex)
        {
        // ESEGUO IL ROLLBACK SE LA TRANSAZIONE E' FALLITA
        myTrans.Rollback();
        throw ex;
        return false;
        }
finally
        {
        // CHIUDO LA CONNESSIONE AL DB
        cn.Close();}}
```

SPERIMENTAZIONE DELL'ISTRUZIONE DELETE CON ODBC .NET

Tipica espressione di DELETE in MySQL:

```
DELETE [LOW_PRIORITY | QUICK] FROM table_name
        [WHERE where_definition]
        [ORDER BY ...]
        [LIMIT rows]
        or
        DELETE [LOW_PRIORITY | QUICK] table_name[.*] [,table_name[.*] ...]
         FROM table-references
        [WHERE where_definition]
        or
        DELETE [LOW_PRIORITY | QUICK]
        FROM table_name[.*], [table_name[.*] ...]
         USING table-references
        [WHERE where_definition]
```

Tipico esempio:
DELETE FROM MiaTabella WHERE id=1

Esempio di DELETE in un file .cs:

```
 // ELIMINA SINGOLO RECORD
public bool eminina_conatto(String ID)
{// APRO LA CONNESSIONE AL DB TRAMITE LA STRINGA DI CONNESSIONE
cn.Open();
command = new OdbcCommand("",cn);
myTrans = cn.BeginTransaction();
command.Transaction = myTrans;
try
        {/
        / CANCELLO UN RECORD
        command.CommandText="DELETE  from "+conf.TABELLA_CONTATTI+"
        where ID="+ID;
        command.ExecuteNonQuery();
        myTrans.Commit();
        return true;}
catch (Exception ex)
        {
        // ESEGUO IL ROLLBACK SE LA TRANSAZIONE E' FALLITA
        myTrans.Rollback();
        throw ex;
        return false;}
finally
        {// CHIUDO LA CONNESSIONE AL DB
        cn.Close();}}// FINE eminina_conatto
```

SPERIMENTAZIONE DELL'ISTRUZIONE SELECT (semplice) CON Odbc.NET

Tipica espressione di SELECT in MySQL:

```
SELECT [STRAIGHT_JOIN]
[SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
[SQL_CACHE | SQL_NO_CACHE]
[SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
      [DISTINCT | DISTINCTROW | ALL]
        select_expression,...
        [INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
    [FROM table_references
      [WHERE where_definition]
    [GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ...
    [HAVING where_definition]
    [ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC] ,...]
    [LIMIT [offset,] rows]
    [PROCEDURE procedure_name]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

Tipico esempio:
SELECT * FROM MiaTabella;

Esempio di SELECT in un file .cs:

```
// RECUPERA RECORD
public DataTable recupera_dati()
{
DataTable dt = new DataTable();
// ESEGUO LA SELECT
command_select = new OdbcDataAdapter("select * from "+conf.TABELLA_CONTATTI ,
cn);

try
        {
        command_select.Fill (dt);
        }
catch (Exception ex)
        {
        throw ex;
        }
return dt;
}// FINE DI recupera_dati
```

SPERIMENTAZIONE DELL'ISTRUZIONE UPDATE CON ODBC .NET

Tipica espressione di UPDATE in MySQL:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
SET col_name1=expr1 [, col_name2=expr2, ...]
 [WHERE where_definition]
 [LIMIT #]
```

Tipico esempio:
UPDATE MiaTabella SET Nome='Giovanni' WHERE Nome='Mario';

Esempio di UPDATE in un file .cs:

```
 // UPDATE SINGOLO RECORD
public bool update_conatto(String ID)
{
// APRO LA CONNESSIONE AL DB TRAMITE LA STRINGA DI CONNESSIONE
cn.Open();
command = new OdbcCommand("",cn);
myTrans = cn.BeginTransaction();
command.Transaction = myTrans;

try
        {
        // ESEGUO L'UPDATE
        command.CommandText="UPDATE "+conf.TABELLA_CONTATTI+" SET
        nomeCognome='Gustavo' where ID="+ID;
        command.ExecuteNonQuery();
        myTrans.Commit();
        return true;
        }
catch (Exception ex)
        {
        // ESEGUO IL ROLLBACK SE LA TRANSAZIONE E' FALLITA
        myTrans.Rollback();
        throw ex;
        return false;
        }
finally
        {
        // CHIUDO LA CONNESSIONE AL DB
        cn.Close();
        }
}// FINE DI update_conatto
```

SPERIMENTAZIONE DELL'ISTRUZIONE CREATE TABLE CON ODBC .NET

Tipica espressione di CREATE TABLE in MySQL:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
        tbl_name [(create_definition,...)]
        [table_options] [select_statement]

create_definition:
         col_name type [NOT NULL | NULL] [DEFAULT default_value]
         [AUTO_INCREMENT]
         [PRIMARY KEY] [reference_definition]
        or
         PRIMARY KEY (index_col_name,...
         or
         KEY [index_name] (index_col_name,...)
         or
         INDEX [index_name] (index_col_name,...)
         or
         UNIQUE [INDEX] [index_name] (index_col_name,...)
         or
         FULLTEXT [INDEX] [index_name] (index_col_name,...)
         or
         [CONSTRAINT symbol]
         FOREIGN KEY [index_name] (index_col_name,...)
        [reference_definition]
       or    CHECK (expr)

type:    TINYINT[(length)] [UNSIGNED] [ZEROFILL]
         SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
         MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
         INT[(length)] [UNSIGNED] [ZEROFILL]
         INTEGER[(length)] [UNSIGNED] [ZEROFILL]
         BIGINT[(length)] [UNSIGNED] [ZEROFILL]
         REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
         DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
         FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
         DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
         NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
         CHAR(length) [BINARY]
         VARCHAR(length) [BINARY]
         DATE
         TIME
         TIMESTAMP
         DATETIME
         TINYBLOB
         BLOB
         MEDIUMBLOB
```

```
        LONGBLOB
        TINYTEXT
        TEXT
        MEDIUMTEXT
        LONGTEXT
        ENUM(value1,value2,value3,...)
        SET(value1,value2,value3,...)
        index_col_name:
        col_name [(length)]

reference_definition:

      REFERENCES tbl_name [(index_col_name,...)]
    [MATCH FULL | MATCH PARTIAL]
    [ON DELETE reference_option]
    [ON UPDATE reference_option]

reference_option:

      RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

table_options:

      TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE |
              MRG_MYISAM | MYISAM }
        AUTO_INCREMENT = #
        AVG_ROW_LENGTH = #
        CHECKSUM = {0 | 1}
        COMMENT = "string"
or      MAX_ROWS = #
        MIN_ROWS = #
        PACK_KEYS = {0 | 1 | DEFAULT}
        PASSWORD = "string"
        DELAY_KEY_WRITE = {0 | 1}
        ROW_FORMAT= { default | dynamic | fixed | compressed }
        RAID_TYPE= {1 | STRIPED | RAID0 } RAID_CHUNKS=#
                          RAID_CHUNKSIZE=#
        UNION = (table_name,[table_name...])
        INSERT_METHOD= {NO | FIRST | LAST }
        DATA DIRECTORY="absolute path to directory"
        INDEX DIRECTORY="absolute path to directory"

select_statement:

       [IGNORE | REPLACE] SELECT ...  (Some legal select statement)
```

Tipico esempio:

```
CREATE TABLE MiaTabella (
                `ID` tinyint(3) NOT NULL default '0',
                `Nome` varchar(100) NOT NULL default '',
                `Cognome` varchar(100) NOT NULL default '',
                PRIMARY KEY  (`ID`,`Nome`),
                KEY `ID` (`ID`),
                KEY `NomeF` (`Nome`),
                FOREIGN KEY (`ID`)
                REFERENCES MiaTab2 (`IDAgente`)
                ON DELETE CASCADE,
                FOREIGN KEY (`Nome`)
                REFERENCES MiaTab3 (`Nome`) ON DELETE CASCADE
                        ) TYPE=InnoDB
```

Esempio di CREATE TABLE (con InnoDB e foreign Key) in un file .cs:

```
// CREAO UNA TABELLA
public bool crea_tabella()
{
// APRO LA CONNESSIONE AL DB TRAMITE LA STRINGA DI CONNESSIONE
cn = new OdbcConnection
                ("DRIVER={MySQL ODBC 3.51 Driver};
                SERVER="+conf.NOMESERVER+";DATABASE=dbinnodb;
                UID=utente;PASSWORD=ciao;");
cn.Open();
command = new OdbcCommand("",cn1);
myTrans = cn1.BeginTransaction();
command.Transaction = myTrans;

try
        {
        // CREO LA TABELLA USANDO IL TIPO TRANSACTION SAFE InnoDB
        command.CommandText = "CREATE TABLE  tabellacreata ("
                        +"IDProva tinyint(3) NOT NULL default '0',"
                        +"NomeFProva varchar(100) NOT NULL,"
                        +"NomeVProva varchar(100) NOT NULL,"
                        +"PRIMARY KEY  (IDProva,NomeFProva),"
                        +"KEY IDProva (IDProva),"
                        +"KEY NomeFProva (NomeFProva),"
                        +"FOREIGN KEY (IDProva) REFERENCES id (IDAgente)
                        ON DELETE CASCADE,"
                        +"FOREIGN KEY (NomeFProva) REFERENCES alias (Nome)
                        ON DELETE CASCADE"
                        +") TYPE=InnoDB";
```

```
        command.ExecuteNonQuery();
        myTrans.Commit();
        return true;
        }
catch (Exception ex)
        {
        // ESEGUO IL ROLLBACK SE LA TRANSAZIONE E' FALLITA
        myTrans.Rollback();
        throw ex;
        return false;
        }
finally
        {
        // CHIUDO LA CONNESSIONE AL DB
        cn.Close();
        }
}// FINE DI crea tabella
```

SPERIMENTAZIONE DEL DROP TABLE CON ODBC .NET

Tipica espressione di DROP TABLE in MySQL:

DROP TABLE [IF EXISTS] tbl_name [, tbl_name,...] [RESTRICT | CASCADE]

Tipico esempio:
DROP TABLE IF EXISTS MiaTablella;

Esempio di DROP TABLE in un file .cs:

```
// ELIMINA TABELLA
public bool elimina_tabella()
{
// APRO LA CONNESSIONE AL DB TRAMITE LA STRINGA DI CONNESSIONE
cn = new OdbcConnection
                        ("DRIVER={MySQL ODBC 3.51 Driver};
                        SERVER="+conf.NOMESERVER+";DATABASE=dbinnodb;
                        UID=utente;PASSWORD=ciao;");
cn.Open();
command = new OdbcCommand("",cn1);
myTrans = cn1.BeginTransaction();
command.Transaction = myTrans;
try
        {
        // ELIMINO LA TABELLA NEL DB
        command.CommandText = "DROP TABLE  tabellacreata;";
        command.ExecuteNonQuery();
        myTrans.Commit();
        return true;
        }
catch (Exception ex)
        {
        // ESEGUO IL ROLLBACK SE LA TRANSAZIONE E' FALLITA
        myTrans.Rollback();
        throw ex;
        return false;
        }
finally
        {
        // CHIUDO LA CONNESSIONE AL DB
        cn.Close();
        }
}// FINE DI elimino tabella
```

SPERIMENTAZIONE DELL'ISTRUZIONE ALTER TABLE CON ODBC .NET

Tipica espressione di ALTER TABLE in MySQL:

ALTER [IGNORE] TABLE tbl_name alter_spec [, alter_spec ...]

alter_specification:
ADD [COLUMN]
create_definition [FIRST | AFTER column_name]
or ADD [COLUMN] (create_definition, create_definition,...)
or ADD INDEX [index_name] (index_col_name,...)
or ADD PRIMARY KEY (index_col_name,...)
or ADD UNIQUE [index_name] (index_col_name,...)
or ADD FULLTEXT [index_name] (index_col_name,...)
or ADD [CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name,...) [reference_definition]

ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}

or CHANGE [COLUMN] old_col_name create_definition [FIRST | AFTER column_name]
or MODIFY [COLUMN] create_definition [FIRST | AFTER column_name]
or DROP [COLUMN] col_name
or DROP PRIMARY KEY
or DROP INDEX index_name
or DISABLE KEYS
or ENABLE KEYS
or RENAME [TO] new_tbl_name
or ORDER BY col
or table_options

Tipico esempio:
UPDATE MiaTabella SET Nome='Laura' WHERE Nome='Giovanni';

Esempio di ALTER TABLE in un file .cs:

```
// ALTERA TABELLA
public bool altera_tabella()
{
// APRO LA CONNESSIONE AL DB TRAMITE LA STRINGA DI CONNESSIONE
cn = new OdbcConnection
        ("DRIVER={MySQL ODBC 3.51 Driver};
        SERVER="+conf.NOMESERVER+";DATABASE=contatti;
        UID=utente;PASSWORD=ciao;");
cn.Open();
command = new OdbcCommand("",cn1);
myTrans = cn1.BeginTransaction();
command.Transaction = myTrans;
```

```
try
        {
        // ESEGUO L'ALTER TABLE SUL DB
        command.CommandText = "ALTER TABLE t_contatti ADD UNIQUE
nomeCognome (nomeCognome);";
        command.ExecuteNonQuery();
        myTrans.Commit();
        return true;
        }
catch (Exception ex)
        {
        // ESEGUO IL ROLLBACK SE LA TRANSAZIONE E' FALLITA
        myTrans.Rollback();
        throw ex;
        return false;
        }
finally
        {
        // CHIUDO LA CONNESSIONE AL DB
        cn.Close();
        }
}// FINE DI altero tabella
```

Abbiamo quindi verificato che con minime differenze di sintassi (rappresentate soprattutto dai vari tipi di tabelle implementati da MySQL) è possibile trasferire un applicazione sviluppata tramite framework .NET da una piattaforma SQL Server a una MySQL tramite il driver ODBC.

3.5.2.2 Connessione di MySQL tramite Native Provider

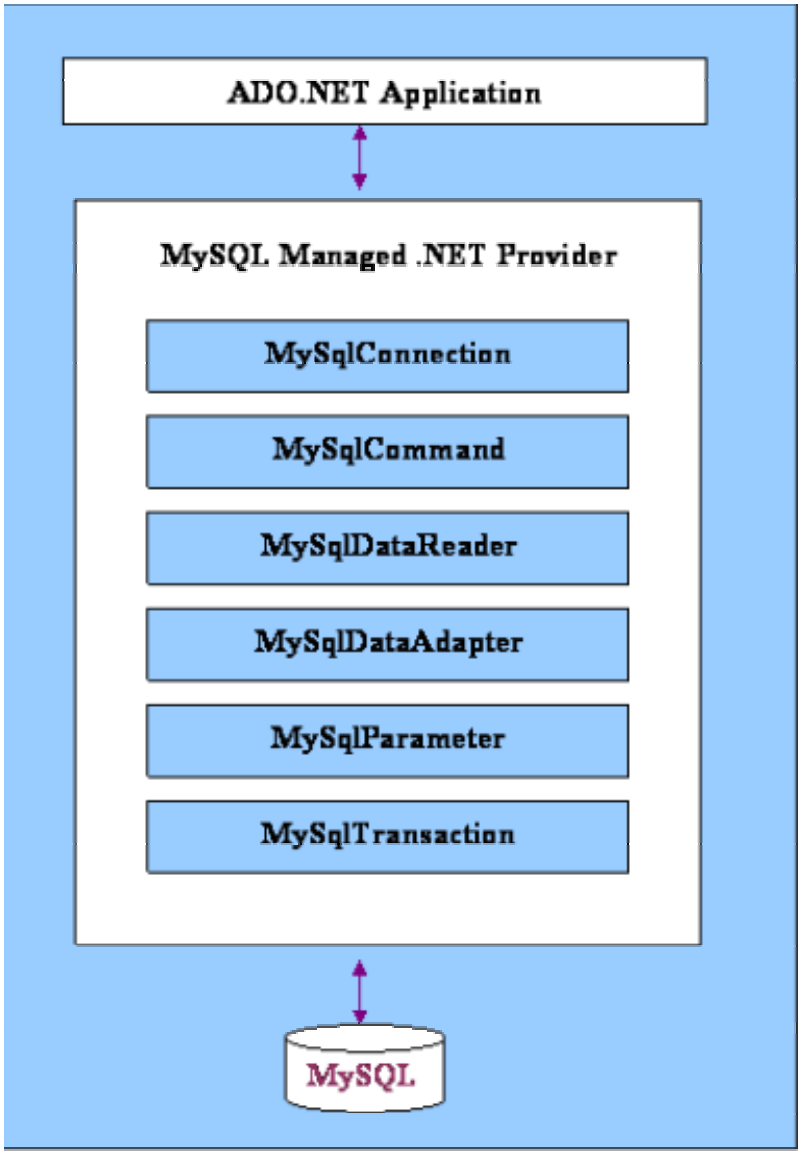
Il Framework .NET mette a disposizione un Native Provider in grado di fornire classi per l'interazione con un database implementato tramite MySQL; questa soluzione alternativa all'uso dei driver ODBC ha il pregio di essere ottimizzata (quindi più veloce) per il Framework, ma ha il grosso svantaggio di non essere standard (come ogni soluzione nativa) e rende quindi impossibile sviluppare un'applicazione Web che comporti una variazione dinamica (o anche statica, implementando versioni differenti) della piattaforma su cui è sviluppato il database.

Si dovrà ricorre a questa soluzione solo se si è sicuri che la propria applicazione non dovrà mai essere trasferita su database implementati con differenti tecnologie, se questo è vero allora utilizzando il Native Provider e MySQL si possono ottenere applicazioni Web veramente veloci, anche se il codice è molto meno leggibile.

E' necessario quindi installare i seguenti componenti sulla piattaforma che si intende utilizzare per sviluppare applicazioni .NET che facciano riferimento a un database MySQL:

- Si deve installare .NET Framework SDK 1.0.
- Si deve installare Microsoft Data Access Components (MDAC) 2.7.
- Si deve installare MySQL Server 4.0.12.
- Sid eve installare un MySQL .NET provider.
- Si devono registrare le DLL fornite dal provider

Una volta installati tutti i componenti il Native Provider farà da “ponte” per il trasporto dei dati fra le applicazioni sviluppate tramite le classi e gli oggetti del Framework e il database MySQL.



Capitolo 3, Figura 2: Architettura MySQL ADO .NET

Il provider mette a disposizione le seguenti classi ADO.NET che dovranno essere usate dallo sviluppatore dell'applicazione che intende utilizzare il Native Provider per connettersi al database MySQL.

Classi	Descrizione
MySqlConnection	La connessione principale al database MySQL.
MySqlCommand	Attiva l'esecuzione di un qualsiasi commando al database.
MySqlDataReader	Fornisce un veloce servizio di lettura forward-only sul database.
MySqlDataAdapter	Serve da interfaccia tra il database MySQL e il DataSet.
MySqlParameter	Viene utilizzato per archiviare parametric da fornire ai comandi.
MySqlTransaction	Usato per rappresentare le transzaioni MySQL.

Per usare il Native Provider, si deve cambiare il "system.data.sqlclient" namespace con il namespace dello specifico provider.

```
using ByteFX.Data.MySQLClient;
```

Si devono dichiarare gli oggetti che si intende usare seguendo la sintassi prevista dal Native Provider.

```
protected MySqlConnection cn;
...
...
protected MySqlCommand command;
```

Sarà quindi necessario creare una stringa di connessione che possa utilizzare il Native Provider per connettersi al server MySQL; questa stringa dovrà quindi indicare:l'indirizzo del server sul quale è fisicamente dislocato il database MySQL, il nome del Database a cui ci si vuole connettere e lo user name dell'utente che vuole connettersi.
Una volta creata la stringa di connessione, la si utilizzerà per aprire e chiudere la connessione ogni volta che sarà necessario.

```
string StringaDiConnessione="
                        Data Source=192.168.1.27;Database=MioDB;User ID=root;";
miaCon = new MySqlConnection(StringaDiConnessione);
……
miaCon.Open();
command = new MySqlCommand("",StringaDiConnessione);
……
miaCon.Close();
```

Una volta scritto il codice che descrive l'applicazione .NET, per compilare adeguatamente i file di progetto (.cs) si deve aggiungere nel file batch (.bat) che si occupa della compilazione una istruzione che contenga il riferimento al Native Provider.

```
@echo off
csc /debug+ /debug:full /out:..\..\bin\contatti.dll /t:library /r:System.dll
/r:System.Data.dll  /r:System.Web.dll
/r: "C:\Programmi\eInfoDesigns\dbProvider Personal Edition\dbProvider.dll"

FileConfigurazione.cs  File1.cs database_File1.cs
```

Dopo aver editato un file batch come sopra, si dovrà: aprire un promt comandi, entrare nella directory dove si trovano i file da compilare ed eseguire il file per la compilazione. Si otterrà quindi il file eseguibile che rappresenta l'applicazione perfettamente integrato col Native Provider e quindi con MySQL.

A questo punto per visualizzare in locale la pagina ASP .NET in modo che venga interpretata dal compilatore JIT correttamente si deve creare una directory virtuale tramite l'IIS (Internet Information Service) dopo di che lanciare la pagina ASP .NET dal localhost.
Dopo di che si devono usare le istruzioni proprietarie del Provider che essendo sviluppate per dare le massime performance non sono standard tanto che per esempio non sono stati implementati i costruttori degli oggetti che si devono inizializzare esplicitamente.

SPERIMENTAZIONE DI ALCUNE ISTRUZIONI TRAMITE PROVIDER

```
using System;
using System.Data;
using System.Diagnostics;
using eInfoDesigns.dbProvider.MySqlClient;
namespace dbProviderTest
{class ConsoleTest
        {static void Main(string[] args)
                {// IMPOSTO I DATI PER LA CONNESSIONE
                string DataSource = "localhost";
                string Database = "test";
                string UserID = "root";
                string Password = "root";
                // CREO LA STRINGA DI CONNESSIONE
                MySqlConnection c = new MySqlConnection("Data Source=" +
                                DataSource";Database=" + Database
                                + ";" + "User ID=" + UserID + ";Password="
                                + Password + ";COMMAND LOGGING=false");
                c.Open();
                MySqlCommand cmd;
                MySqlDataReader dr;
                try
                        {// SELECT CON PARAMETRI
                        cmd = new MySqlCommand(
                                        "SELECT e.Employee_ID, e.First_Name, “
                                        + “e.Last_Name, e.Date_Hired"
                                        + " FROM employee e"
                                        + " WHERE e.Employee_ID = @EmployeeID"
                                        + " ORDER BY e.Last_Name, e.First_Name asc", c);

                        //  AGGIUNGO I PARAMETRI E IMPOSTO I LORO VALORI
                        cmd.Parameters.Add("@EmployeeID", DbType.Int32);
                        cmd.Parameters["@EmployeeID"].Value = 1;

                        // ESEGUO LA LETTURA CHE RESTITUISCE UN DataReader
                        dr = (MySqlDataReader)cmd.ExecuteReader();
                        int i, f;
                        f = dr.FieldCount;
                        for (i = 0; i < f; i++)
                                Console.WriteLine ("Field[" + i + "]=" + dr.GetName(i) + "," +
                                        dr.GetFieldType(i) + "," + dr.GetDataTypeName(i));
                                        while (dr.Read())
                                {for (i = 0; i < dr.FieldCount; i++)
                                        {if (dr.IsDBNull(i))
                                                Console.Write ("null" + ",");
                                        else
                                        {if (dr.GetDataTypeName(i).Equals("varchar"))
```

```
                    Console.Write(dr.GetString(i) + ",");
                else if (dr.GetDataTypeName(i).Equals("int16") ||
                    dr.GetDataTypeName(i).Equals("int32") ||
                    dr.GetDataTypeName(i).Equals("int64") ||
                    dr.GetDataTypeName(i).Equals("int"))
                    Console.Write(dr.GetInt32(i) + ",");
                else if (dr.GetDataTypeName(i).Equals("datetime"))
                    Console.Write(dr.GetDateTime(i) + ",");}}
        Console.WriteLine("");}
dr.Close();

//  ESEGUO UN UPDATE
cmd.CommandText = "UPDATE department SET Name='Accounts'" +
                " WHERE Department_ID = 1";
                int RowsUpdated = cmd.ExecuteNonQuery();
Console.WriteLine ("Rows updated (first update):  " + RowsUpdated);

//  USO UN DataSet PER GENERARE UN DOCUMENTO XML
MySqlDataAdapter da = new MySqlDataAdapter();
MySqlCommand cmdDa = new MySqlCommand();

cmdDa.CommandText =
                "SELECT d.Name \"Department_Name\"," +
                " Concat(e.First_Name, ' ', e.Last_Name)
                + \"Employee_Name\""
                +" FROM departmentemployee de "
                +" INNER JOIN department d on de.Department_
                + ID = d.Department_ID"
                +"INNER JOIN employee e on de.Employee_
                + ID = e.Employee_ID";
cmdDa.Connection = c;
da.SelectCommand = cmdDa;
da.TableMappings.AddRange(new
System.Data.Common.DataTableMapping[] {
new System.Data.Common.DataTableMapping("Table",
"DepartmentEmployees", new
System.Data.Common.DataColumnMapping[] {
new System.Data.Common.DataColumnMapping("Department Name",
"Department Name"),
new System.Data.Common.DataColumnMapping("Employee Name",
"Employee Name")})});

//  CREO IL DataSet USANDO UN DatAdaopter PER  RIEMPIRLO
DataSet ds = new DataSet();
da.Fill(ds);

Console.WriteLine ("DataSet results:");
foreach (DataTable t in ds.Tables)
```

```
                    {Console.WriteLine("table name=" + t.TableName);
                    foreach (DataRow r in t.Rows)
                    foreach (DataColumn co in t.Columns)
                    Console.WriteLine ("column name=" + co.ColumnName +
                                ",type=" + co.DataType + "," +
                                "value=" + r[co]);}
                    Console.WriteLine ("Finished with the DataSet!");

                    //  SCRIVO L'XML
                    ds.WriteXml("DepartmentEmployees.xml",
                    XmlWriteMode.WriteSchema);
                    Console.WriteLine ("Wrote the XML!");

                    //  ESEGUO UNA TRANSAZIONE SICURA
                    try
                    {tr = (MySqlTransaction)c.BeginGuardedTransaction();

                    //  FACCIO UN UPDATE
                    cmd.CommandText = "UPDATE account SET Balance=500
                    WHERE Account_ID=1";
                    cmd.ExecuteNonQuery();

                    //  FACCIO UN ALTRO UPDATE.  MENTRE IL PRIMO
                    //UPDATE E'ATTIVO SULLA TABELLA.
                    //  SENZA TRANSAZIONI SICURE QUESTA
                    // OPERAZIONE FALLIREBBE
                    cmd.CommandText =
                    "UPDATE employee SET First_Name='Nothing' WHERE
                    Employee_ID=1";
                    cmd.ExecuteNonQuery();

                    // NON SUCCEDE MAI, MA NEL CASO DI ERRORE
                    // FACCIO IL ROLLBACK
                    tr.Rollback();
                    }
                    catch (Exception e)
                    {Console.WriteLine("Exception in guarded transaction--" +
                     e.Message);}
            }
    }}
```

Abbiamo quindi verificato che rendendo il codice ancora più complesso di quello generato dagli ODBC è comunque possibile connettersi a un database MySQL con soluzioni native che garantiscono elevate prestazioni a scapito della facilità di progettazione e della standardizzazione del codice sorgente.

3.5.2.3 Connessione tramite OLEDB

Mentre SLQ Server 2000 si connette perfettamente utilizzando il recente supporto fornito dalle OLEDB, questo è ancora pericoloso per MySQL.
Infatti le procedure di connessione tramite driver OLEDB sono del tutto simili a quelle precedentemente esposte per i più vecchi driver ODBC, sono anche presenti sul mercato driver OLEDB che forniscano questi servizi al server MySQL, ma purtroppo MySQL non è ancora pronto per supportare perfettamente questo standard che deve essere quindi abbandonato dagli sviluppatori.

3.5.2.4 Conclusioni sulla connessione tra MySQL e Framework .NET

SQL Server 2000 è compatibile con il Framework .NET tramite qualsiasi soluzione presente sul mercato, quindi è opportuno scegliere per la connessione fra applicazioni web e database i driver OLEDB di ultima generazione che permettono una programmazione intuitiva e semplice offrendo elevate prestazioni.

Per quanto riguarda MySQL il discorso è più complesso, mentre si attende una standardizzazione delle OLEDB la scelta deve ricadere o su i driver ODBC o su una soluzione tramite Native Provider.

ODBC è progettato per la massima intercomunicabilità, ogni applicazione ha quindi la possibilità di accedere a diversi DBMS con lo stesso codice sorgente, inoltre ODBC è abbastanza facile da utilizzare e genera un codice abbastanza ridotto e intuitivo anche per uno sviluppatore meno esperto degli oggetti Microsoft.

Il Native .NET Provider offre una velocità di esecuzione notevole e genera un codice ottimizzato per il provider e per il database sfruttando al meglio la già elevata velocità di un database MySQL; purtroppo però per implementare questa soluzione sono necessarie profonde conoscenze del Framework da parte dello sviluppatore e il codice così creato è più prolisso e molto meno leggibile. Un applicazione scritta utilizzando un Native provider è impossibilitata per definizione a reagire dinamicamente a un cambio di tecnologia a livello del database e deve essere completamente riscritta o anche accantonata se il database dovesse cambiare.

Prendendo in considerazione questi elementi, sono del parere che convenga utilizzare ODBC sacrificando un po' di velocità e puntando sulla possibilità di esportare facilmente il codice sorgente su diverse piattaforme; ma la scelta è soprattutto basata sulle esigenze dello sviluppatore che dovrà a questo punto decidere da solo.

4. Uso di XML come tecnologia per l'interoperabilità tra DBMS Relazionali

4.1 XML e HTML

HTML (Hyper Text Markup Language) nasce come lingua franca per pubblicare informazioni ipertestuali sul World Wide Web. L'HTML è un formato, non proprietario, basato su SGML (Standard Generalized Markup Language) strutturato in modo da poter essere sviluppato e processato da una grande gamma di tool più o meno complessi: dal blocco note di Windows ai sofisticati WSISIWSIG authoring tools.
HTML usa i tag, ad esempio <H1> e la sua chiusura </H1> per strutturare il testo in headings, paragraphs, lists, hyiperlinks, ecc. in questo modo HTML è in grado, formattando i dati senza utilizzare validazioni o strutture rigorose di diffondere ipertesti tramite il protocollo HTTP attraverso la rete.

Con lo sviluppo della rete e la sempre maggiore diffusione di nuovi Borwser concorrenti, proprio le caratteristiche che hanno dato forza all'HTML si sono trasformate nelle sue debolezze: divenne sempre più necessario che l'informazione ipertestuale portasse con sé, oltre che all'informazione in se, anche informazioni aggiuntive sulla rappresentazione e la struttura dei dati contenuti nel testo, allo scopo di rendere le pagine HTML sempre più multimediali e vicine all'utente.

Questo processo ha reso arduo il compito degli sviluppatori che avevano la loro forza nella semplicità e nella standardizzazione del protocollo HTML; facendo comparire sul mercato sempre più versioni proprietarie, ci si rese sempre più conto del fatto che lo strumento HTML non era adatto allo scambio di informazioni sul web proprio per le caratteristiche di scarso controllo sul formato e sulla struttura dei dati che lo contraddistinguevano.

Si decise quindi di sviluppare un più appropriato protocollo orientato verso l'interscambio di dati sul web, passando così all'XML.

L'XML (Extensible Markup Language) è uno standard per la gestione dei documenti proposto, come l'HTML prima di lui, dal World Wide Web Consortium (W3C).
Si può dire che l'XML si una forma semplificata dello Standard SGML (estremamente complicato e troppo pesante per il web), cioè un meta-linguaggio che permette di creare e di formattare i propri tag (maracatori) personali per i documenti.
Mentre con l'HTML i tag erano statici: ad esempio <HEAD> o <BODY> invece, con l'XML, si possono creare i propri tag e definirli a piacimento, ad esempio <TitoloGrande>, <Commento> o <MioFont>; rendendo obsoleta la definizione di teg "corretti" all'interno del documento.

Ciascuno di questi nuovi tag sarà poi definito e spiegato ramite una serie di strumenti a scelta dello sviluppatore quali: DTD (Document Type Definition), XSL (Extensible Styleseet Language), XSD Schema o altri.
L'XML ha dunque messo a disposizione del Web uno strumento dinamico, semplice e strutturato (è possibile vedere un documento XML come un albero a nodi che rappresenta tramite la nidificazione dei tag la complessità dell'informazione) per scambiare dati in modo formale e conciso tramite applicazioni relativamente semplici, creando documenti leggibili tanto dagli elaboratori che dagli utenti.

4.2 Struttura dei Documenti XML

4.2.1 Terminologia dell'XML

Per prima cosa è necessario chiarire i concetti elementari che stanno alla base di un documento XML. Un documento è costituito da uno o più elementi marcati con la seguente sintassi:

```
<TESTO>
Questo testo è formattato secondo l'elemento Testo.
</TESTO>
```

Questo elemento consiste di due tag, uno di apertura (nome dell'elemento chiuso fra un segno di minore (<) e un segno di maggiore (>)), ed uno di chiusura (identico al primo ma con l'aggiunta di una barra (/) prima del nome dell'elemento). Come per un documento HTML il testo contenuto fra i due tag è considerato parte dell'elemento stesso e viene formattato secondo le sue regole.

Gli elementi possono avere degli attributi così espressi:

```
<TESTO lingua="Italiano">
Questo testo è formattato secondo l'elemento Testo.
</TESTO>
```

L'attributo viene specificato nel tag di apertura e in questo caso si chiama "lingua". Il Suo valore è "Italiano"; gli attributi vengono spesso usati per rendere più specifico o per modificare il comportamento di default di un elemento, anche se nulla vieta di usare un attributo per la rappresentazione di un dato (Vedi Uso per XPath).
Oltre agli elementi visti in precedenza, l'XML supporta anche elementi vuoti: cioè senza testo all'interno dei suoi tag di apertura e chiusura.

Questi elementi vengono in genere utilizzati per inserire del contenuto che non si testo in un documento o per fornire informazioni aggiuntive all'applicazione che elabora il documento XML.
E' importante notare che la barra non può essere tralasciata come per l'HTML.

4.2.2 Perdere le Brutte Abitudini

I Browser HTML ignorano spesso semplici errori contenuti nei documenti, purtroppo per gli sviluppatori di XML, le applicazioni che elaborano XML non sono altrettanto condiscendenti verso il progettista che deve assolutamente perdere alcune brutte abitudini:

I Valori degli attributi devono essere racchiusi tra virgolette: non è possibile specificare il valore di un attributo in questo modo: src:/immagini/MiaImmagine.jpg; mentre questo errore sarebbe spesso ignorato da un Browser HTML, per un applicazione XML, il valore deve essere incluso tra apici o virgolette, o il parser XML darà un errore. Il Modo corretto di assegnare un valore ad un attributo è il seguente:

```
<FIGURA src=”/immagini/MiaImmagine.jpg”>
```

Gli elementi non vuoti devono avere un tag di apertura ed uno di chiusura o il parser XML darà un errore; non è possibile scrivere un documento XML in questo modo:

```
<RIGA>
Questa è una riga di un testo.
<RIGA>
Questa è un'altra riga del testo.
```

Ciascun elemento deve, invece, possedere un tag di apertura e uno di chiusura:

```
<RIGA>Questa è una riga di un testo.</RIGA>
<RIGA>Questa è un'altra riga del testo.</RIGA>
```

I vari tag devono essere nidificati correttamente, non è possibile scrivere:

```
<RIGA><CELLA>Questo è sbagliato.</RIGA></CELLA>
```

Il tag di chiusura per l’elemento Cella deve essere interno a quello dell’elemento Riga, in modo da corrispondere al tag di apertura più vicino, e da preservare una corretta nidificazione. E’ fondamentale per un applicazione che deve elaborare un documento XML rappresentare correttamente la gerarchia degli elementi; il giusto modo di nidificare i tag è il seguente:

```
<RIGA><CELLA>Questo è giusto.</CELLA></RIGA>
```

Se il documento XML è conforme a queste poche regole sintattiche può essere definito ben impostato.

4.3 XSD Schema

4.3.1 Perché XSD Schema

Una volta definito in modo rigoroso un documento XML, si deve rendere questo documento valido; cioè si devono esprimere regole generali a cui il documento stesso deve essere conforme.
Infatti, se si vuole che i dati scambiati tramite un documento XML possano essere univocamente interpretati da tutti i partecipanti alla comunicazione, è necessario che venga codificato un unico documento di riferimento che funga da prototipo.

In questo modo possono essere definite una volta per tutte sia la struttura che la grammatica degli elementi che compongono tutti i documenti XML scambiati.
E' possibile supplire a questo problema in vari modi, utilizzando diversi strumenti che nel tempo sono stati resi disponibili per gli sviluppatori; inizialmente un documento XML veniva fornito di due file aggiuntivi: un file DTD e un file XSL.

Lo scopo di una DTD (Document Type Definition) è di dichiarare ciascuno degli elementi usati nel documento XML definendo una serie di regole che definiscano tutti gli aspetti architetturali e semantici da verificare nel documento XML.

Un XSL (Extensible Staylesheet Language) consiste in una serie di tag atta a fornire regole di formattazione a ciascuno degli elementi interni a un documento XML.

Questi strumenti sono comunque limitati e recentemente è stato sviluppato un nuovo strumento per lo sviluppo l'XSD Schema (rilasciato dal W3C); oltre a possedere maggiori potenzialità descrittive e di controllo questo schema è anche ufficialmente supportato dal Framework di .NET che dispone di veri e propri oggetti orientati verso l'utilizzo di schemi XSD.

Si è deciso dunque di scegliere questa opzione che garantisce notevoli strumenti per il controllo e la formattazione di un file XML e anche una notevole compatibilità con i vari strumenti di sviluppo utilizzati per questa ricerca.

4.3.2 Strumenti di XSD Schema

4.3.2.1 Introduzione

Uno schema è contenuto in un file del tipo MioSchema.xsd e consiste in vari elementi (definiti tramite tag) dei quali i più importanti sono: element, complexType e simpleType che definisco la struttura e i contenuti degli elementi nel documento sorgente XML.

Ogni elemento dello schema ha come prefisso xs: associato all'XML Schema Namespace nella dichiarazione: xmlns:xs=http://www.w3.org/2001/XMLSchema che appare nell'elemento schema; nel caso si trovi il prefisso msdata: esso si riferisce alla dichiarazione: xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" posta sempre nell'elemento schema.

4.3.2.2 Definizione dei Tipi Complessi e Dichiarazione di Elementi e Attributi

Al contrario dei Tipi Semplici i Tipi Complessi dispongono della possibilità di possedere sia elementi che attributi. I Tipi complessi si definiscono usando l'elemento **complexType** che in genere contiene una serie dichiarazioni di elementi e attributi. Gli elementi all'interno del Tipo Complesso sono definiti usando l'elemento **element**, mentre gli attributi sono identificati dall'elemento **attribute**.

```
<xs:complexType name="T_Software">
  <xs:sequence>
        <xs:element name="ID"  type="xs:int" />
        <xs:element name="DataInserimento" type="xs:dateTime" />
        ……
        ……
        <xs:element name="DaSviluppare" type="xs:string"/>
        <xs:element name="Note" type="xs:string"/>
  </xs:sequence>
              <xs:attribute versione=”1.0” type=”xs:decimal”/>
</xs:complexType>
```

Si può impostare il numero di volte che un elemento può essere presente indicando il numero minimo di volte (**minOccurs**) e il numero massimo di volte (**maxOccurs**, senza limiti se dichiarato **unbounded**) .

```
<xs:element name="DaSviluppare" type="xs:string"  minOccurs="0"
                                    maxOccurs=”unbounded”/>
```

Gli attributi possono comparire una volta o nessuna (ma non un certo numero di volte) e possono essere dichiarati (tramite **use**) come opzionali, richiesti o proibiti (**richiesto, optional, proibhited**).

```
<xs:attribute versione=”1.0” type=”xs:decimal” use=”richiesto”/>
```

Possono essere messi dei valori di Default sia agli attributi che agli elementi tramite l’istruzione **default**; nel primo caso se un attributo non è stato inizializzato nel documento il compilatore mette il valore predefinito, nel caso dell’elemento se esso non è presente non viene inizializzato col valore di default.

```
<xs:element name="ID" type="xsd:integer" default="10"/>
```

Si può usare l’istruzione **fixed** per dire che se un attributo o un elemento sono presenti devono avere un certo valore.

```
<xs:attribute versione=”1.0” type=”xs:decimal” use=”richiesto” fixed=”0.0”/>
```

Questa tabella riassume i vincoli per gli elementi e gli attributi.

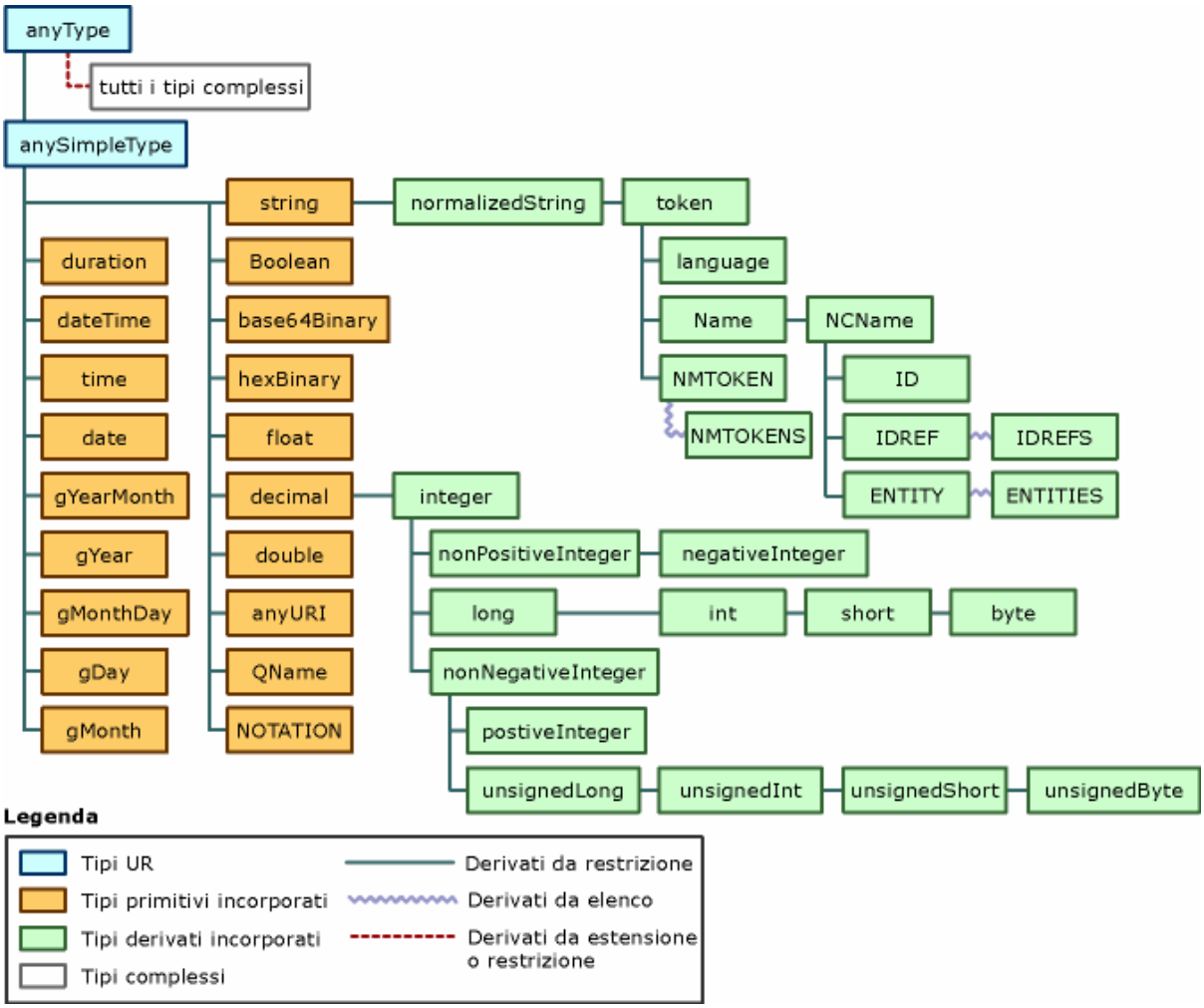
Elementi (minOccurs, maxOccurs) fixed, default	**Attrinuti (use, fixed, default)**	**Note**
(1, 1) -, -	richiesto, -, -	L’elemento/attributo deve apparire almeno una volta, può avere qualsiasi valore.
(1, 1) 3, -	richiesto, 3, -	L’elemento/attributo deve apparire almeno una volta, il suo valore deve essere 3.
(2, unbounded) 3, -	n/a	L’elemento deve apparire due volte o più, il suo valore deve essere 3; in generale minOccurs e maxOccurs devono avere valori interi positivi e maxOccurs può assumere il valore "unbounded" .
(0, 1) -, -	opzionale, -, -	L’elemento/attributo può apparire una volta, può assumere qualsiasi valore.
(0, 1) 3, -	opzionale, 3, -	L’Elemento/attributo può apparire una volta, se appare il suo valore deve essere 3, se non appare il suo valore è 3.
(0, 1) -, 3	opzionale, -, 3	L’Elemento/attributo può apparire una volta, se non appare il suo valore è 3 di default altrimenti il suo valore è dato.
(0, 2) -, 3	n/a	L’Elemento può apparire due volte o nessuna, se non appare non ha nessun valore, se appare ma è vuoto il suo valore è 3 di default, altrimenti il suo valore è dato.
(0, 0) -, -	proibito, -, -	L’Elemento/attributo non deve apparire mai.

4.3.2.3 Tipi Semplici

XSD fornisce un certo numero di Tipi Semplici che possono essere usati per definire elementi, attributi, Tipi Complessi o perfino possono essere modificati per ottenere delle restrizioni (come vedremo in seguito).

Simple Type	Esempio
string	---
token	---
bSite	-1, 126
unsignedBSite	0, 126
base64BinarSi	GpM7
hexBinarSi	0FB7
integer	-126789, -1, 0, 1, 126789
positiveInteger	1, 126789
negativeInteger	-126789, -1
nonNegativeInteger	0, 1, 126789
nonPositiveInteger	-126789, -1, 0
int	-1, 126789675
unsignedInt	0, 1267896754
long	-1, 12678967543233
unsignedLong	0, 12678967543233
short	-1, 12678
unsignedShort	0, 12678
decimal	-1.23, 0, 123.4, 1000.00
float	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
double	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN
boolean	true, false 1, 0
time	13:20:00.000, 13:20:00.000-05:00
dateTime	1999-05-31T13:20:00.000-05:00
duration	P1SI2M3DT10H30M12.3S
date	1999-05-31
gMonth	--05--
gSIear	1999
gSIearMonth	1999-02
gDaSi	---31
gMonthDaSi	--05-31
Name	shipTo
QName	po:USAddress
NCName	USAddress
anSiURI	http://www.example.com/, http://www.example.com/doc.html#ID5
language	en-GB, en-US, fr
ID	---
IDREF	---
ENTITIES	---
NMTOKEN	US,Brésil

Nella raccomandazione W3C è riportata la specifica per la definizione dei tipi di dati utilizzati nell'XSD Schema. Tale specifica riguarda la definizione di tipi di dati primitivi incorporati, tipi di dati derivati e facet.



Capitolo 4, Figura 3: Gerarchia dei Tipi

Usando l'istruzione **simpleType** per definire e nominare il nuovo elemento semplice si possono ricavare nuovi Tipi Semplici di dato restringendo tipi già esistenti tramite il comando **restriction**. Supponiamo di voler creare un nuovo tipo di intero chiamato MioIntero il cui range varia tra 1000 e 9999 (inclusivo). Basiamo la nostra definizione sul Tipo Semplice integre e restringiamo il suo range con i "facets" **minInclusive** e **maxInclusive.**

```
<xs:simpleType name="myInteger">
        <xs:restriction base="xsd:integer">
                <xs:minInclusive value="1000"/>
                <xs:maxInclusive value="9999"/>
        </xs:restriction>
</xs:simpleType>
```

Sono disponibili numerosi "**Facet**" ma non tutti sono applicabili a tutti i Tipi Semplici per una completa lista delle opzioni possibili si rimanda il lettore all'Appendice B.
Si possono creare elementi Lista cioè liste di elementi atomici tramite il costrutto **<xsd:lsit itemType="nometipo"/>**, in questo caso possono essere applicate delle restrizioni.

```
<xs:simpleType name="listaDiMioIntType">
  <xs:list itemType="MioInteger"/>
</xs:simpleType>
```

Se si vuole aggiungere un attributo a un tipo semplice di dato, si deve per forza definire un tipo di dato complesso che contenga il tipo semplice e aggiungere un attributo.

```
<xs:element name="internationalPrice">
 <xs:complexType>
  <xs:simpleContent>
                <xs:extension base="xsd:decimal">
                        <xs:attribute name="currency" type="xsd:string"/>
                </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

Questa struttura può essere usata anche per creare elementi vuoti con solo degli attributi.
In questo caso la struttura può essere alleggerita diventando:

```
<xs:element name="internationalPrice">
   <xs:complexType>
          <xs:attribute name="currency" type="xsd:string"/>
          <xs:attribute name="value"    type="xsd:decimal"/>
  </xs:complexType>
</xs:element>
```

4.3.2.4 Annotazioni e Commenti

Si possono usare i comandi **annotation, documentation e appInfo** per aggiungere commenti per rendere la lettura del codice più semplice per un utente.

```
<xs:element name="internationalPrice">
        <xs:annotation>
                <xs:documentation xml:lang="en">
                Elemento dichiarato con  Anonymous Type
                </xs:documentation>
        </xs:annotation>
        <xs:complexType>
                <xs:annotation>
                        <xs:documentation xml:lang="en">
                          Anonymous Type vuoto con due attributi
                        </xs:documentation>
                </xs:annotation>
        <xs:complexContent>
        <xs:restriction base="xs:anyType">
                <xs:attribute name="currency" type="xs:string"/>
        <xs:attribute name="value"    type="xs:decimal"/>
        </xs:restriction>
        </xs:complexContent>
 </xs:complexType>
</xs:element>
```

4.3.2.5 Elementi Nulli

In generale l'assenza di un elemento non ha nessun valore particolare: può indicare che l'informazione relativa all'elemento è sconosciuta, non applicabile o l'elemento può mancare per qualsivoglia altro motivo.

A volte, però, è interessante rappresentare l'elemento mancante, l'informazione inapplicabile o non conosciuta come esplicitamente assente. Per esempio, può essere desiderabile rappresentare un valore "Null" come risultato di un operazione fatta su un Database relazionale. In queste circostanze è possibile usare l'XSD nill che permette di impostare il valore "Null".

Il meccanismo che garantisce il "Null" in XSD Schema utilizza il segnale di "out of band", in altre parole non c'è nessun elemento che appare come Null, ma è compito di un attributo indicare che un elemento potrà, o meno. Essere "Null".
Per fare questo si aggiunge la parte nillable="true" nel file MioSchema.xsd

```
<xs:element name="multiportal" type="xsd:integer" nillable="true"/>
```

Dopo di chè si aggiunge la parte nello spazio dei nomi del file MioXML.xml

xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance

Gli elementi in questione devono essere presenti nel file MioXML.xml in questo modo:

```
<multiportal xsi:nil="true"></multiportal>
```

4.3.2.6 Specificare l'Unicità

XSD Schema offre la possibilità di indicare che un attributo o un elemento deve essere unico secondo un certo scopo. Per definire che un attributo o un elemento sia unico possiamo utilizzare l'elemento **unique**.

Per prima cosa, all'interno dell'elemento inique, utilizziamo l'elemento **selector** a cui viene dato un percorso **xpath** (di cui spiegheremo in seguito le particolarità e i modi di utilizzo) per identificare l'elemento o l'attributo che deve essere reso unico; dopo di che si usa l'elemento **field** per indicare l'elemento da rendere unico. E' importante che la dichiarazione dei campi unici venga eseguita all'interno dell'elemento di Root o dell'elemento padre di tutti i nodi figli che dovranno risultare unici.

```
<xs:unique name="T_Siti_Constraint1" msdata:ConstraintName="Constraint1">
    <xs:selector xpath=".//T_Siti" />
    <xs:field xpath="ID" />
</xs:unique>
```

4.3.2.7 Definizione di Chiavi e loro Riferimenti

Può succedere che definire un elemento (o un attributo) unico non sia sufficiente e si voglia, per qualche motivo, rendere ancora più forti i vincoli fra gli elementi. In questo caso è possibile utilizzare l'elemento **keyref**, che presenta una sintassi del tutto simile all'elemento inique.

L'uso delle chiavi rende unici gli elementi (o gli attributi) e impedisce che vengano definiti "Null", inoltre rende l'elemento così definito referenziabile da qualsiasi luogo.

```
<xs:keyref name="Constraint2" refer="Constraint1" msdata:ConstraintOnly="true">
    <xs:selector xpath=".//T_SitiSoftware" />
    <xs:field xpath="IDSoftware" />
</xs:keyref>
```

4.3.2.8 Corrispondenza Dati fra XSD Schema e Framework .NET

Lavorando con gli oggetti forniti dal Framework .NET si devono prendere alcune precauzione nell'utilizzo dei Tipi di Dato se si vuole mantenere la compatibilità fra XSD Schema e oggetti .NET; per questo viene fornita nell'Appendice B una tabella che descrive la codifica proprietaria del Framework in rapporto ai tipi di dato forniti dall'XSD Schema.

4.4 XPath

4.4.1 Introduzione a XPath

XPath (XML Path Language) è il risultato di uno sforzo per fornire una sintassi comune tra XSLT e XPointer; lo scopo primario di XPath è quello di indirizzare parti di un documento XML. Oltre a questo scopo maggiore, XPath fornisce anche un consistente numero di funzioni per la manipolazione di stringhe, numeri e valori booleani.

XPath utilizza una struttura compatta e non-XML per navigare all'interno della struttura gerarchica di un documento XML; infatti XPath modella il documento sorgente XML come un albero a nodi.
Esistono molti tipi di nodi, tra i quali **element nodes**, **attribute nodes** e **text nodes**.

Dovendo descrive in breve XPath si possono indicare i seguenti punti:

- XPath è una sintassi per definire parti di un documento XML.
- XPath usa dei paths per definire elementi XML.
- XPath definisce librerie di funzioni standard.
- XPath è il maggiore elemento nei XSLT.
- XPath non è scritto nel documento XML.
- XPath è uno standard W3C.

4.4.2 Funzioni XPath

XPath fornisce agli sviluppatori, che intendono utilizzarlo per muoversi all'interno di un documento XML, un certo numero di funzioni proprietarie che si dividono in quattro grandi categorie:

- Node Set Function.
- Boolean Function.
- String Function.
- Number Function.

A queste funzioni unisce un complete set di operatori algebrici e relazionali (*, +, -, div, mod, =, !=, <, <=, >, >=, and e or), per un approfondimento maggiore, il lettore è riamandato all'Appendice B.

4.4.3 Esempi di Interrogazioni XPath

La sintassi base di XPath è simile a quella del filesystem addressing.
Se il path parte con la barra "/ " allora rappresenta il percorso assoluto all'elemento richiesto.

/AAA
Seleziona l'elemento di Root AAA
<AAA> <BBB/> <CCC/> <BBB/> <BBB/> <DDD> <BBB/> </DDD> <CCC/> </AAA>

/AAA/CCC
Seleziona tutti gli elementi CCC figli dell'elemento Root AAA
<AAA> <BBB/> <CCC/> <BBB/> <BBB/> <DDD> <BBB/> </DDD> <CCC/> </AAA>

Se il path inizia con "//" allora tutti gli elementi del documento che soddisfano il criterio scelto saranno selezionati.

//BBB
Seleziona tutti gli Elementi BBB
<AAA> **<BBB/>** **<CCC/>** **<BBB/>** **<DDD>** **<BBB/>** **</DDD>** **<CCC>** **<DDD>** **<BBB/>** **<BBB/>** **</DDD>** **</CCC>** **</AAA>**

//DDD/BBB
Seleziona tutti gli elementi BBB figli degli elementi DDD
<AAA> **<BBB/>** **<CCC/>** **<BBB/>** **<DDD>** **<BBB/>** **</DDD>** **<CCC>** **<DDD>** **<BBB/>** **<BBB/>** **</DDD>** **</CCC>** **</AAA>**

Il Simbolo “*” seleziona tutti gli elementi indicate dal path che lo precede.

/AAA/CCC/DDD/*
Seleziona tutti gli elementi racchiusi nella gerarchia /AAA/CCC/DDD

```
<AAA>
      <XXX>
         <DDD/>
      </XXX>
      <CCC>
         <DDD>
            <BBB/>
            <BBB/>
            <EEE/>
            <FFF/>
         </DDD>
      </CCC>
      <CCC>
         <BBB/>
      </CCC>
   </AAA>
```

/*/*/*/BBB
Seleziona tutti gli elementi BBB che hanno tre ancestors

```
   <AAA>
      <XXX>
         <DDD>
            <BBB/>
            <BBB/>
            <EEE/>
            <FFF/>
         </DDD>
      </XXX>
      <CCC>
         <DDD>
            <BBB/>
            <BBB/>
            <EEE/>
            <FFF/>
         </DDD>
      </CCC>
      <CCC>
         <BBB>
            <BBB>
               <BBB/>
            </BBB>
         </BBB>
      </CCC>
   </AAA>
```

//*
Seleziona tutti gli elementi

```
<AAA>
   <XXX>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </XXX>
   <CCC>
      <DDD>
         <BBB/>
         <BBB/>
         <EEE/>
         <FFF/>
      </DDD>
   </CCC>
   <CCC>
      <BBB>
         <BBB>
            <BBB/>
         </BBB>
      </BBB>
   </CCC>
</AAA>
```

L'espressione racchiusa fra "[ ]" serve per specificare un elemento. Un numero fra quadre indica la posizione dell'elemento.
La Funzione **last( )** seleziona l'ultimo elemento di una selezione.

/AAA/BBB[1]
Seleziona il primo elemento BBB figlio dell'elemento AAA

```
<AAA>
      <BBB/>
      <BBB/>
      <BBB/>
      <BBB/>
</AAA>
```

/AAA/BBB[last()]
Seleziona l'ultimo elemento BBB figlio dell'elemento AAA
<AAA> <BBB/> <BBB/> <BBB/> <BBB/> </AAA>

Gli Attributi sono specificati dal prefisso " @" .

//@id
Seleziona tutti gli attributi @id
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>

//BBB[@id]
Seleziona gli elementi BBB che hanno come attributo id
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>

//BBB[@*]
Seleziona gli elementi BBB che possiedono qualsiasi attributo
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>

//BBB[not(@*)]
Selziona gli elementi BBB che non possiedono attributi
<AAA> <BBB id = "b1"/> <BBB id = "b2"/> <BBB name = "bbb"/> <BBB/> </AAA>

I Valori degli attributi possono essere utilizzati come criteri di selezione.
La Funzione **normalize-space( )** rimuove gli spazi che precedono o seguono un nome e sostituisce una sequenza di spazi con uno spazio singolo.

//BBB[@id='b1']
Seleziona gli elementi BBB che hanno come valore dell'attributo id la stringa b1
<AAA> <BBB id = "b1"/> <BBB name = " bbb "/> <BBB name = "bbb"/> </AAA>

//BBB[@name='bbb']
Seleziona gli elementi BBB che hanno come valore dell'attributo name la stringa bbb
<AAA> <BBB id = "b1"/> <BBB name = " bbb "/> <BBB name = "bbb"/> </AAA>

//BBB[normalize-space(@name)='bbb']
Seleziona gli elementi BBB che hanno come valore dell'attributo name la stringa bbb, eventuali spazi vengono tolti prima del confronto.
<AAA> <BBB id = "b1"/> <BBB name = " bbb "/> <BBB name = "bbb"/> </AAA>

La funzione **count( )** conta il numero di elementi selezionati.

//*[count(BBB)=2]
Seleziona tutti gli elemento che possiedono due figli BBB
<AAA> <CCC> <BBB/> <BBB/> <BBB/> </CCC> <DDD> <BBB/> <BBB/> </DDD> <EEE> <CCC/> <DDD/> </EEE> </AAA>

//*[count(*)=2]
Seleziona tutti gli elemento che possiedono due figli qualsiasi

```
<AAA>
   <CCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </CCC>
   <DDD>
      <BBB/>
      <BBB/>
   </DDD>
   <EEE>
      <CCC/>
      <DDD/>
   </EEE>
</AAA>
```

La funzione **name( )** restituisce il nome dell'elemento.
La funzione **starts-with( )** ritorna true se la stringa data come primo argomento comincia con la stringa data come secondo argomento.
La funzione **contains( )** ritorna true se la stringa data come primo argomento contiene la stringa data come secondo argomento.

//*[name()='BBB']
Seleziona tutti gli elementi di nome BBB (equivalente dell'istruzione //BBB)

```
<AAA>
   <BCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </BCC>
   <DDB>
      <BBB/>
      <BBB/>
   </DDB>
   <BEC>
      <CCC/>
      <DBD/>
   </BEC>
</AAA>
```

//*[starts-with(name(),'B')]
Seleziona tutti gli elementi il cui nome inizia con la lettera B

```
<AAA>
   <BCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </BCC>
   <DDB>
      <BBB/>
      <BBB/>
   </DDB>
   <BEC>
      <CCC/>
   </BEC>
</AAA>
```

//*[contains(name(),'C')]
Seleziona tutti gli elementi il cui nome contiene la lettera C

```
<AAA>
   <BCC>
      <BBB/>
      <BBB/>
      <BBB/>
   </BCC>
   <DDB>
      <BBB/>
      <BBB/>
   </DDB>
   <BEC>
      <CCC/>
      <DBD/>
   </BEC>
</AAA>
```

La funzione **string-length( )** restituisce il numero di caratteri contenuti in una stringa.
Si deve usare **<** come sostituto di "<" e **>** come sostituto di ">".

//*[string-length(name()) = 3]
Seleziona gli elementi con un nome lungo tre caratteri

```
<AAA>
   <Q/>
   <SSSS/>
   <BB/>
   <CCC/>
   <DDDDDDDD/>
   <EEEE/>
</AAA>
```

//*[string-length(name()) < 3]
Seleziona gli elementi con un nome lungo da uno a due caratteri
<AAA> <Q/> <SSSS/> <BB/> <CCC/> <DDDDDDDD/> <EEEE/> </AAA>

Più path possono essere collegati tramite ” | “.

//CCC | //BBB
Seleziona tutti gli elementi CCC e BBB
<AAA> <BBB/> <CCC/> <DDD> <CCC/> </DDD> <EEE/> </AAA>

/AAA/EEE | //DDD/CCC | /AAA | //BBB
Il Numero di combinazioni non ha restrizioni di sorta
<AAA> <BBB/> <CCC/> <DDD> <CCC/> </DDD> <EEE/> </AAA>

L'Asse del Figlio contiene i filgi del context node ed essendo l'asse di default può essere omesso.

/AAA
Equivalente a /child::AAA
<AAA> <BBB/> <CCC/> </AAA>

/child::AAA/BBB
Entrambe le possibilità sono permesse
<AAA> <BBB/> <CCC/> </AAA>

L'asse dei discendenti contiene tutti i discendenti del context node; un discendente è un figlio o un figlio del figlio e così via. L'asse dei discendenti non contiene mai un attribute node o un namespace node.

/descendant::*
Seleziona tutti i discendenti della Root del documento
<AAA> <BBB> <DDD> <CCC> <DDD/> <EEE/> </CCC> </DDD> </BBB> <CCC> <DDD> <EEE> <DDD> <FFF/> </DDD> </EEE> </DDD> </CCC> </AAA>

/AAA/BBB/descendant::*
Seleziona tutti i discendenti dell'elemento /AAA/BBB

```
<AAA>
    <BBB>
        <DDD>
            <CCC>
                <DDD/>
                <EEE/>
            </CCC>
        </DDD>
    </BBB>
    <CCC>
        <DDD>
            <EEE>
                <DDD>
                    <FFF/>
                </DDD>
            </EEE>
        </DDD>
    </CCC>
</AAA>
```

//CCC/descendant::DDD
Seleziona gli elementi DDD che hanno un elemento CCC come avo

```
<AAA>
    <BBB>
        <DDD>
            <CCC>
                <DDD/>
                <EEE/>
            </CCC>
        </DDD>
    </BBB>
    <CCC>
        <DDD>
            <EEE>
                <DDD>
                    <FFF/>
                </DDD>
            </EEE>
        </DDD>
    </CCC>
</AAA>
```

L’asse dei parenti contiene I perenti del context node, se ne esiste qualcuno.

//DDD/parent::*
Seleziona tutti i parenti dell’elemento DDD

```
<AAA>
   <BBB>
      <DDD>
         <CCC>
            <DDD/>
            <EEE/>
         </CCC>
      </DDD>
   </BBB>
   <CCC>
      <DDD>
         <EEE>
            <DDD/>
         </EEE>
      </DDD>
   </CCC>
</AAA>
```

L’asse degli avi contiene tutti gli avi del context node; gli avi del context node sono tutti i suoi parenti e i parenti dei suoi parenti e così via. L’asse degli avi contiene sempre l’elemento di Root a meno che il context node non si la Root stessa.

/AAA/BBB/DDD/CCC/EEE/ancestor::*
Seleziona tutti gli elementi dati in questo percorso assoluto

```
<AAA>
   <BBB>
      <DDD>
         <CCC>
            <DDD/>
            <EEE/>
         </CCC>
      </DDD>
   </BBB>
   <CCC>
      <DDD>
         <EEE>
            <DDD>
               <FFF/>
            </DDD>
         </EEE>
      </DDD>
   </CCC>
</AAA>
```

//FFF/ancestor::*
Seleziona tutti gli avi dell'elemento FFF

```
<AAA>
    <BBB>
        <DDD>
            <CCC>
                <DDD/>
                <EEE/>
            </CCC>
        </DDD>
    </BBB>
    <CCC>
        <DDD>
            <EEE>
                <DDD>
                    <FFF/>
                </DDD>
            </EEE>
        </DDD>
    </CCC>
</AAA>
```

L'asse dei vicini successivi, contiene tutti gli elementi figli del context node di primo grado.

/AAA/BBB/following-sibling::*

```
<AAA>
    <BBB>
        <CCC/>
        <DDD/>
    </BBB>
    <XXX>
        <DDD>
            <EEE/>
            <DDD/>
            <CCC/>
            <FFF/>
            <FFF>
                <GGG/>
            </FFF>
        </DDD>
    </XXX>
    <CCC>
        <DDD/>
    </CCC>
</AAA>
```

L’asse dei vicini precedenti contiene tutti gli elementi appenda precedenti il context node.

/AAA/XXX/preceding-sibling::*

```
<AAA>
   <BBB>
      <CCC/>
      <DDD/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
   </CCC>
</AAA>
```

L’asse dei successivi contiene tutti i nodi nello stesso documento del context node che sono successivi al context node nell’ordine del documento esclusi gli attribute node e i namespace node.

//ZZZ/following::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ>
         <EEE/>
      </ZZZ>
      <FFF>
         <GGG/>
      </FFF>
   </BBB>
   <XXX>
      <DDD>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
</AAA>
```

L’asse dei precedenti contiene tutti i nodi, nello steso documento del context node, che precedono il context node nell’ordine del documento, escludendo attribute node e namespace node.

//GGG/preceding::*

```
<AAA>
   <BBB>
      <CCC/>
      <DDD/>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
</AAA>
```

L’asse dei discendenti-o-se contiene il context node e tutti I suoi discendenti.

//CCC/descendant-or-self::*

```
<AAA>
   <BBB>
      <CCC/>
      <ZZZ>
         <DDD/>
      </ZZZ>
   </BBB>
   <XXX>
      <DDD>
         <EEE/>
         <DDD/>
         <CCC/>
         <FFF/>
         <FFF>
            <GGG/>
         </FFF>
      </DDD>
   </XXX>
   <CCC>
      <DDD/>
      <EEE/>
       <FFF>
```

```
            <GGG/>
        </FFF>
    </CCC>
</AAA>
```

L'asso degli avi-o-se contiene il context node e tutti I suoi avi, questo asse include sempre la Root del documento.

//GGG/ancestor-or-self::*

```
<AAA>
    <BBB>
        <CCC/>
        <ZZZ>
            <DDD/>
        </ZZZ>
    </BBB>
    <XXX>
        <DDD>
            <EEE/>
            <DDD/>
            <CCC/>
            <FFF/>
            <FFF>
                <GGG/>
            </FFF>
        </DDD>
    </XXX>
    <CCC>
        <DDD/>
    </CCC>
</AAA>
```

Gli assi degli avi, dei discendenti, dei successivi, dei precedenti e l'asse di se, partizionano il documento (escludendo gli attribute node e i namespace node). Essi non si sovrappongono e se uniti in una selezione contengono tutti i nodi del documento.

//GGG/ancestor::* | //GGG/descendant::* | //GGG/following::* | //GGG/preceding::* | //GGG/self::*

```
<AAA>
    <BBB>
        <CCC/>
        <ZZZ/>
    </BBB>
    <XXX>
        <DDD>
            <EEE/>
            <FFF>
                <HHH/>
                <GGG>
                    <JJJ>
                        <QQQ/>
                    </JJJ>
                    <JJJ/>
                </GGG>
                <HHH/>
            </FFF>
        </DDD>
    </XXX>
    <CCC>
        <DDD/>
    </CCC>
</AAA>
```

L'operatore **div** esprime una divisione in floating-point, l'operatore **mod** restituisce il resto di una divisione troncata.

La funzione **floor** restituisce il più grande numero intero che non sia superiore all'argomento dato come parametro.

La funzione **ceiling** restituisce il più piccolo numero intero che non sia minore dell'argomento dato come parametro.

//BBB[position() mod 2 = 0]
Seleziona alcuni degli elementi BBB
<AAA> **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<CCC/>** **<CCC/>** **<CCC/>** **</AAA>**

//BBB[position() = floor(last() div 2 + 0.5) or position() = ceiling(last() div 2 + 0.5)]
Seleziona l'elemento BBB mediano in una lista di elementi BBB
<AAA> **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<BBB/>** **<CCC/>** **<CCC/>** **<CCC/>** **</AAA>**

4.5 Uso di XML per la gestione dei Dati in .NET

4.5.1 .NET e XML

Tramite .NET Framework è possibile progettare applicazioni disponendo di un suite integrata di classi XML in grado di mostrare ed utilizzare al meglio le innovazioni nel mondo dell'XML. Le classi XML fornite sono elementi centrali di .NET Framework che grazie ad esse fornisce una soluzione interoperabile, aperta, e conforme agli standard per i problemi tipici degli sviluppatori che desiderano operare con queste nuove tecnologie per il trasferimento dati sul web.

Gli obiettivi della progettazione in relazione alle classi XML in .NET Framework sono:

- Aderenza agli standard W3C.
- Estensibilità.
- Architettura a collegabilità immediata.
- Prestazioni.
- Stretta integrazione con ADO.NET.

4.5.1.1 Aderenza agli Standard W3C

Per aderenza agli standard si intende conformità delle classi agli standard attualmente raccomandati dal consorzio W3C in relazione a XML, spazi dei nomi, XSLT, XPath, XSD Schema e DOM (Document Object Model). Tale conformità garantisce l'interoperabilità fra le varie piattaforme e applicazioni, agevolando lo sviluppo.

La soluzione verso la quale la tecnologia .NET spinge maggiormente è la raccomandazione W3C XML Schema Definition language (XSD) 1.0 che garantisce, come visto in precedenza, un elevato controllo e un'ottima formattazione dei dati XML.

Attraverso alcune classi XML in .NET Framework è fornita la convalida di un documento XML ed è disponibile un modello di oggetti per la generazione di Schemi XSD in memoria.
Il parser veloce, di tipo forward-only con il quale è possibile eseguire la convalida in base agli schemi, ai DTD, agli XDR e XSD è chiamato **XmlValidatingReader** ed è un parser conforme a XML.
La classe **XmlSchemaCollection** può essere utilizzata per memorizzare nella cache gli schemi XSD o XDR utilizzati di frequente quando si utilizza **XmlValidatingReader**.

In .NET Framework è inoltre disponibile un gruppo di classi XML da cui deriva un modello SOM (Schema Object Model) che consente di generare e compilare schemi XSD a livello di codice. La classe **XmlSchema** rappresenta uno schema XSD. È possibile caricare tali schemi e mantenerli tramite le classi **XmlReader** e **XmlWriter**.

Le raccomandazioni del DOM (Document Object Model) di livello 1 e livello 2 sono implementate dalla classe **XmlDocument** personalizzata in base alle indicazioni di progettazione comuni di .NET Framework.

La classe **XslTransform** è conforme alle raccomandazioni XSL Transformations (XSLT) Version 1.0 e XML Path Language (XPath) 1.0 per trasformare i documenti utilizzando l'XSLT.

4.5.1.2 Estensibilità

L'XML in .NET Framework è estensibile tramite le classi astratte di base e i metodi virtuali. Questa estensibilità, o creazione di sottoclassi, viene illustrata dalle classi astratte **XmlReader**, **XmlWriter** e **XPathNavigator**, che consentono lo sviluppo di nuove implementazioni su più archivi e origini di dati e l'esposizione dell'XML.

Nell'API **XPathNavigator**, ad esempio, è incorporato un modulo di gestione delle query XPath che può essere implementato sugli archivi, come i file system, i registri e i database relazionali; in questo modo non solo vengono visualizzati i dati come l'XML, ma viene fornito anche il supporto delle query XPath su origini di dati diverse, utilizzando l'implementazione predefinita dei metodi delle query API XPath, come **Select**.

Un altro esempio di estensibilità è costituito da **XmlReader**, che fornisce un'API per l'analisi rapida di tipo forward-only di un archivio e che espone l'Infoset XML che rileva all'interno del flusso. L'XML in .NET Framework ha implementazioni di **XmlReader** per la lettura dei flussi, le classi **XmlNodeReader** per la lettura delle strutture di nodi e **XmlValidatingReader** per il supporto della convalida nella lettura di documenti XML.

4.5.1.3 Architettura a collegabilità immediata

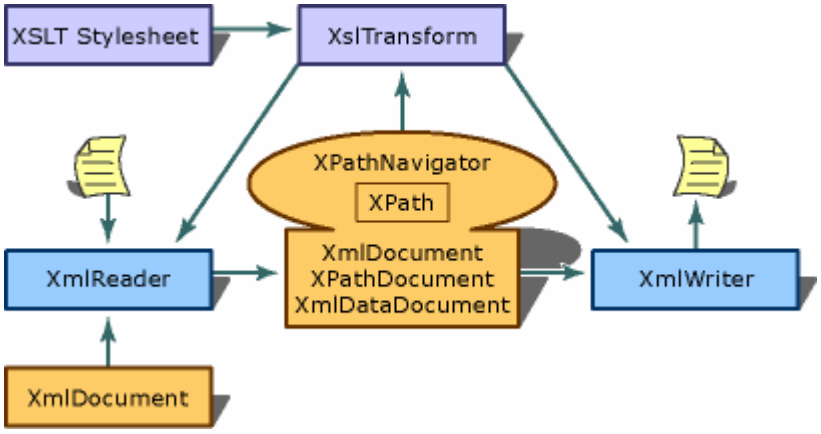
L'XML in .NET Framework è caratterizzato da un'architettura a collegabilità immediata.
In questa architettura basata su flussi, il concetto di collegabilità immediata si riferisce al fatto che i componenti basati su queste classi astratte all'interno di .NET Framework possono essere facilmente sostituiti.

Per architettura a collegabilità immediata si intende anche che è possibile creare un flusso di dati tra i componenti, e che l'elaborazione può essere alterata dall'inserimento di nuovi componenti in questo flusso. È possibile ad esempio analizzare un flusso di XML di un

servizio Web XML con **XmlReader**, che a sua voltà potrà essere utilizzato per creare un **XmlDocument**, che può essere utilizzato a sua volta per la creazione di un **XmlNodeReader**. Un altro esempio collegabilità consiste nel caricare il DOM (classe **XmlDocument**) da un **XmlReader** e nel salvare l'output utilizzando un **XmlWriter** per scrivere un documento XML.

Creando implementazioni personalizzate di queste classi estendendo quelle esistenti, si potrà influire sul comportamento della classe **XmlDocument**. Se, ad esempio, è stata creata un'implementazione di **XmlReader**, chiamata **MioXmlFileReader** in grado di esporre un file system come XML, è possibile caricare un **XmlDocument** da questo reader. In questo modo per le nuove classi basate su quelle esistenti viene fornita un'architettura collegabile.

Un altro esempio di collegamento tra componenti è l'uso di diversi archivi di dati, come **XPathDocument** e **XmlDocument**, nel processo di trasformazione. È possibile trasformare questi archivi di dati con la classe **XslTransform** e l'output potrà allora essere inserito in un flusso in un altro archivio o restituito come flusso da un servizio Web XML. Nell'illustrazione che segue viene illustrato quanto detto sopra.



Capitolo 4, Figura 4: Inserimento di dati in un flusso con la classe XslTransform

Qualsiasi archivio che consente di implementare un **XPathNavigator**, utilizzando l'interfaccia **IXPathNavigable**, può essere collegato nella classe **XslTransform** per consentire le trasformazioni XSLT su quell'archivio.

Le classi **XmlDocument**, **XPathDocument** e **XmlDataDocument** sono in grado di eseguire questa operazione. È possibile quindi inviare l'output inserito in un flusso da **XslTransform** a **XmlReader** o **XmlWriter** nello stile dell'architettura collegabile.

4.5.1.4 Prestazioni

Le classi XML in .NET Framework rappresentano i componenti per l'elaborazione XML di basso livello che vengono utilizzati, non solo come parte di .NET Framework, ma per integrare l'XML nelle applicazioni. È necessario che le classi abbiano prestazioni estremamente buone.

Le classi XML in .NET Framework sono progettate per supportare un modello basato sullo streaming grazie alle seguenti caratteristiche:

- Minimo utilizzo della cache durante l'analisi di tipo forward-only svolta con **XmlReader** secondo il metodo pull.

- Convalida di tipo forward-only con **XmlValidatingReader**.

- Nuovo tipo di navigazione a cursore di **XPathNavigator** grazie alla quale la creazione di nodi è ridotta a un singolo nodo virtuale, fornendo tuttavia l'accesso casuale al documento. In questo modo non è necessario che venga creata in memoria una struttura di nodi come il DOM.

- Output incrementale di flusso dalla classe **XslTransform**.

- **XPathDocument** è un archivio ottimizzato, di sola lettura per le query XPath ed è consigliato ogni volta che è necessaria un'elaborazione XSLT. Utilizzando questo archivio e la classe **XslTransform**, è possibile ottenere trasformazioni XSLT efficienti.

4.5.1.5 Integrazione con ADO .NET

In .NET Framework i dati relazionali e l'XML vengono combinati da una stretta integrazione tra le classi XML e ADO.NET.

Grazie al componente **DataSet**, che rappresenta un database non connesso, è possibile leggere e scrivere l'XML utilizzando le classi **XmlReader** e **XmlWriter**, mantenere la struttura interna degli schemi relazionali come gli Schemi XML (XSD), e ipotizzare la struttura dello schema da un documento XML.

XmlDataDocument va oltre i limiti dei modelli basati sui dati XML e relazionali sincronizzando un **DataSet** con un **XmlDocument** in modo che i dati gestiti in uno dei due ambienti vengano aggiornati nell'altro, dove applicabile. Poiché **XmlDocument** ha la capacità di memorizzare dati semistrutturati, tutte le funzionalità di un archivio XML vengono acquisite, mentre grazie al **DataSet** si avrà una vista relazionale dell'XML sulla base del suo schema.

4.6 Uso di XML come Base di Dati per Piccole Applicazioni Webdatabase

4.6.1 Obbiettivo nell'uso di XML

Durante lo sviluppo di applicazioni Webdatabase che richiedono la gestione dinamica di dati contenuti in database qualsiasi ci si rende presto conto che l'utilizzo di potenti, sofisticate e pesanti piattaforme per l'archiviazione e la gestione dei dati quali SQL Server 2000, DB2 o anche di servizi meno potenti come MySQL porta alti costi e un notevole rallentamento dell'applicazione durante l'accesso e la gestione dei dati stessi.
Queste applicazioni infatti possiedono grandi pregi quali: la gestione sicura degli accessi ai dati, strutture di controllo per il recupero di dati danneggiati, controlli sicuri sulla multiutenza e sicurezza nelle transazioni; oltre ad un potente linguaggio per l'interrogazione del database quale può essere SQL92.

Tutte queste caratteristiche diventano tanto più utili ed importanti quanti più dati sono archiviati, quante più sono le relazioni fra di essi e tanti più sono gli utenti che operano concorrentemente sui dati stessi; a volte però l'uso dei dati (soprattutto per applicazioni Web) è molto limitato mentre l'elemento più importante da tenere in considerazione è la velocità dell'applicazione sul Web.

Nasce quindi la necessità di utilizzare strumenti diversi per la gestione dei dati che comportino minore sicurezza ma che garantiscano un notevole aumento della velocità e una auspicabile diminuzione dei costi.

A questo proposito una delle scelte possibili è quella di utilizzare l'XML come strumento per la gestione dei dati; si deve tenere conto però che un documento XML può essere considerato un database solo nel senso più ristretto del termine; cioè come collezione di dati.
Sotto molti aspetti un documento XML non è poi molto differente da qualsivoglia altro tipo di file, dopotutto qualsiasi file contiene dati di qualche tipo; come "database" però XML dispone di qualche vantaggio.

Per esempio un documento XML è utodescitto (i suoi tag definiscono la struttura, la semantica e il tipo dei dati in essi racchiusi), è Unicode ed è in grado di descrivere i dati tramite strutture ad albero.
Purtroppo ha anche alcuni svantaggi, tra i quali: l'essere prolisso e caratterizzato da una lentezza nell'accesso ai dati e nella loro modifica.
Sono disponibili però molti strumenti che affiancano, completano e ampliano un documento XML (che rappresenta in se solo l'archivio dei dati) per renderlo più funzionale come "database".

Come abbiamo visto nei capitoli precedenti sono disponibili molti strumenti per garantire una più completa descrizione dei dati contenuti in un documento XML, quali: DTD, XSL e XSD Schema; inoltre sono stati sviluppati alcuni linguaggi per l'interrogazione di un XML

(XQuery, XPath, XQL, XML-QL, QUILT) e alcuni strumenti per l'interfaccia verso le applicazioni (SAX, DOM, JDOM).

D'altra parte un documento XML non può ancora supportare molte delle funzioni proprie di un vero database quali: indici, sicurezza, transazioni e integrità dei dati, accesso multiplo sicuro, triggers, interrogazioni da più documenti e stored procedure.

E' quindi da ritenere possibile l'utilizzo di un documento XML come database in applicazioni che necessitino di una piccola quantità di dati, pochi utenti, modeste esigenze di protezione. Queste caratteristiche sono dunque sufficienti se si pensa a piccole applicazioni Web nelle quali sia compito del codice sorgente eseguire controlli per migliorare le prestazione di un "databsae" XML.

Si è dunque pensato, invece di affidarsi a soluzioni proprietarie (comunque gestite via codice C++, C# o Java), di creare un piccolo database XML utilizzando gli strumenti della tecnologia .NET. Con questo obbiettivo è stato analizzato il codice XML associato a uno schema XSD per vedere se era possibile (per alcuni tipi di applicazioni Web) utilizzare questa soluzione sostituendola ai normali dartabase su di una piattaforma .NET.

Una volta preso in considerazione l'XML è stato scelto lo schema XSD (come struttura di controllo sintattico e di consistenza dei dati) perché, come spiegato in precedenza, supportato da .NET e si è deciso di analizzare in dettaglio le due implementazioni rese possibili dal Framework .NET:

- Uso del DOM con interrogazioni XPath
- Uso del DataSet

4.6.2 Soluzione tramite l'uso del DOM e di interrogazioni XPath

4.6.2.1 Introduzione al DOM

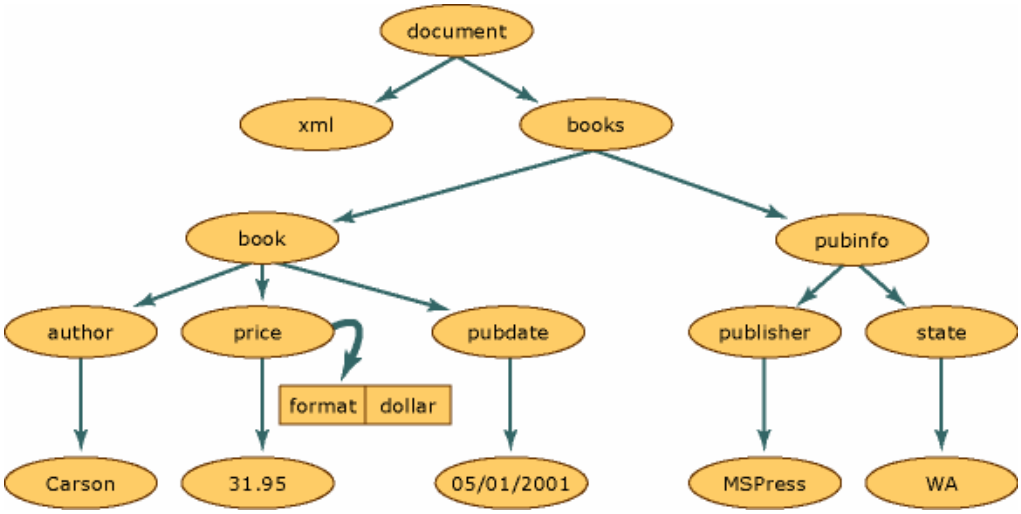
La classe DOM (Document Object Model) è una rappresentazione in memoria di un documento XML; grazie ad essa è possibile leggere e modificare un documento XML a livello di programmazione.

Nonostante l'XML venga letto anche dalla classe **XmlReader**, l'accesso fornito da tale classe è in sola lettura, di tipo forward-only e non memorizzato nella cache. Questo significa che con **XmlReader** non sono disponibili funzionalità per la modifica dei valori di un attributo o del contenuto di un elemento, né per l'inserimento e la rimozione di nodi.

La modifica è la funzione primaria del DOM, è il modo comune e strutturato in cui i dati XML vengono rappresentati nella memoria, sebbene i dati XML effettivi siano memorizzati in modo lineare all'interno di un file o quando provengono da un altro oggetto.
Se consideriamo un documento XML di questo genere:

```
<?xml version="1.0"?>
        <books>
                <book>
                        <author>Carson</author>
                        <price format="dollar">31.95</price>
                        <pubdate>05/01/2001</pubdate>
                <book>
                <pubinfo>
                        <publisher>MSPress</publisher>
                        <state>WA</state>
                </pubinfo>
        </books>
```

Tramite l'uso del DOM questo documento XML verrà strutturato in memoria secondo questo schema.



Capitolo 4, Figura 5: Struttura del documento XML in Memoria tramite DOM

All'interno della struttura del documento XML, ogni cerchio di questa illustrazione rappresenta un nodo, chiamato oggetto **XmlNode**, che costituisce l'oggetto di base nella struttura DOM.

La classe **XmlDocument**, che estende **XmlNode**, supporta i metodi per l'esecuzione di operazioni sul documento nella sua totalità, come ad esempio il caricamento del documento in memoria o il salvataggio del documento XML in un file.

XmlDocument costituisce inoltre un modo per visualizzare e modificare i nodi nell'intero documento XML. **XmlNode** e **XmlDocument** sono stati entrambi migliorati dal punto di vista delle prestazioni e dell'utilizzo e provvisti di metodi e proprietà per accedere e modificare i nodi specifici del DOM, come i nodi degli attributi, i nodi degli elementi, i nodi di riferimento alle entità e così via.

E' inoltre possibile recuperare interi nodi, oltre alle informazioni contenute nel nodo, come il testo in un nodo dell'elemento.

Gli oggetti **Node** dispongono di un gruppo di metodi e proprietà, oltre a caratteristiche di base ben definite; possiedono un unico nodo padre, ovvero quello al livello immediatamente superiore. Gli unici nodi privi di padre sono quelli di primo livello, ovvero Document, in quanto si tratta dei nodi di livello superiore contenenti il documento stesso e frammenti del documento.

La maggior parte dei nodi può avere più nodi figlio, vale a dire nodi al livello immediatamente inferiore.
Una caratteristica del DOM è il modo in cui gestisce gli attributi. Gli attributi non sono nodi che fanno parte di relazioni padre-figlio o tra nodi di pari livello, ma sono considerati una proprietà del nodo dell'elemento e sono costituiti da una coppia composta da nome e valore. Per trovare l'attributo format="dollar" del nodo **price**, richiamare il metodo **GetAttribute** del nodo dell'elemento price.

Le API disponibili nel DOM W3C di Livello 1 e Livello 2 sono state estese per facilitare l'utilizzo di documenti XML e le classi, i metodi e le proprietà aggiuntive estendono la funzionalità aldilà di quella già consentita dall'uso del DOM XML W3C, i cui standard sono completamente supportati.

Grazie alle nuove classi è possibile accedere ai dati relazionali e usufruire di metodi per la sincronizzazione con i dati ADO .NET, esponendo simultaneamente i dati come XML.
Il DOM è particolarmente utile per la lettura dei dati XML in memoria per modificarne la struttura, per aggiungere o rimuovere nodi o per modificare i dati appartenenti a un nodo come nel testo contenuto da un elemento. Sono, tuttavia, disponibili altre classi le cui prestazioni sono migliori in altri scenari.
Inoltre il DOM permette un accesso casuale ai dati XML tramite interrogazioni **XPath** 1.0 come da standard W3C.

E' inoltre stata sviluppata una classe chiamata **XmlDataDocument** che estende la classe **XmlDocument**.
Questo nuovo costrutto consente di caricare dati relazionali o dati XML e modificarli utilizzando il modello DOM (Document Object Model) di W3C. In base al DOM, i dati vengono rappresentati come nella gerarchia di oggetti nodo precedentemente esposta e poiché **XmlDataDocument** implementa l'interfaccia **XPathNavigable**, può essere utilizzata anche come documento di origine per la classe **XslTransform**.
XmlDataDocument ha una stretta connessione con la classe **DataSet** che fornisce una visualizzazione relazionale dei dati XML caricati. Qualsiasi modifica apportata alla classe **XmlDataDocument** viene applicata anche al **DataSet** e viceversa.

4.6.2.2 Documento XML orientato all'uso di XPath

Il documento XML che si intende utilizzare come rappresentazione fisica dei dati deve essere scritto facendo un forte uso degli attributi e un utilizzo quasi nullo degli elementi, questa soluzione ottimizza il lavoro che verrà fatto da XPath e rende i dati anche più standardizzati verso i più comuni tool di esportazione dei vari database.

Inoltre per compatibilità con l'Oggetto DataSet di .NET che pur essendo in grado di importare schemi e documenti XML separatamente, genera (se questa operazione viene eseguita automaticamente da un applicazione .NET) un File XML con Schema annesso; si è deciso di utilizzare un file XML con XSD Schema annesso per mantenere un alta compatibilità con le operazioni standard del Farmework .NET.

```
<?xml version="1.0" standalone="yes"?>

<NewDataSet>


<xs:schema id="NewDataSet"
        xmlns=""
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

        <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="it-IT">
        ….
        ….
</xs:schema>

<T_Software      ID="0" DataInserimento="2003-04-30T13:21:40.1616800+02:00"
DataUltimaModifica="2003-05-01T11:22:25.0668672+02:00" NomeSoftware="Software1"
Versione="4.0.1" VersionePrecedente="4.0" Sviluppatore="Yuri"
DataInizioSviluppo="20/04/2003" DataFineSviluppo="21/05/2003" Sviluppo=""
Correzioni="Correzzione" Test="Molti" Errori="Nessuno" DaSviluppare="Ancora Molto"
Note="Primo Software della Lista" />

<T_Software     ID="1" DataInserimento="2003-04-30T13:22:27.1292160+02:00"
DataUltimaModifica="2003-05-01T11:12:42.9698528+02:00" NomeSoftware="Software2"
Versione="1" VersionePrecedente="0" Sviluppatore="Maria"
DataInizioSviluppo="12/01/2002" DataFineSviluppo="20/03/2003" Sviluppo=""
Correzioni="Nessuna" Test="Pochi" Errori="Nessuno" DaSviluppare="" Note="Secondo
Software della Lista" />

<T_Software  ID="2" DataInserimento="2003-05-01T11:14:08.6029872+02:00"
DataUltimaModifica="2003-05-01T11:14:08.6029872+02:00" NomeSoftware="Software3"
Versione="12" VersionePrecedente="10.1" Sviluppatore="Daniele"
DataInizioSviluppo="03/10/2001" DataFineSviluppo="01/04/2003" Sviluppo=""
Correzioni="Alcune" Test="Nessuno" Errori="Alcuni" DaSviluppare="" Note="Terzo
Software della Lista" />
```

```
<T_Siti ID="0" NomeSito="Sito1" />
<T_Siti ID="1" NomeSito="Sito2" />
<T_Siti ID="2" NomeSito="Sito3" />

<T_SitiSoftware IDSito="0" IDSoftware="0" />
<T_SitiSoftware IDSito="1" IDSoftware="1" />
<T_SitiSoftware IDSito="2" IDSoftware="2" />

</NewDataSet>
```

L'XSD Schema che fornisce i controlli per i dati XML precedenti, usa in modo massiccio gli attributi per facilitare il successivo uso di XPath, inoltre lo schema esegue i molti controlli sul file sorgente XML che saranno spiegati nel dettaglio nella soluzione tramite il DataSet (le uniche differenze saranno dato dall'uso degli attributi al posto degli elementi).

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
<xs:schema id="NewDataSet"
          xmlns=""
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

   <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="it-IT">
     <xs:complexType>
       <xs:choice maxOccurs="unbounded">

        <xs:element name="T_Software">
          <xs:complexType>
            <xs:attribute name="ID" msdata:AutoIncrement="true" type="int" use="required" />
            <xs:attribute name="DataInserimento" type="xs:dateTime" use="required" />
            <xs:attribute name="DataUltimaModifica" type="xs:dateTime" use="required" />
            <xs:attribute name="NomeSoftware" type="xs:string" use="required" />
            <xs:attribute name="Versione" type="xs:string" use="required" />
            <xs:attribute name="VersionePrecedente" type="xs:string" />
            <xs:attribute name="Sviluppatore" type="xs:string" use="required" />
            <xs:attribute name="DataInizioSviluppo" type="xs:string" />
            <xs:attribute name="DataFineSviluppo" type="xs:string" />
            <xs:attribute name="Sviluppo" type="xs:string" />
            <xs:attribute name="Correzioni" type="xs:string" />
            <xs:attribute name="Test" type="xs:string" />
            <xs:attribute name="Errori" type="xs:string" />
            <xs:attribute name="DaSviluppare" type="xs:string" />
            <xs:attribute name="Note" type="xs:string" />
          </xs:complexType>
        </xs:element>
```

```
      <xs:element name="T_Siti">
        <xs:complexType>
          <xs:attribute name="ID" msdata:AutoIncrement="true" type="int" use="required" />
          <xs:attribute name="NomeSito" type="xs:string" use="required" />
        </xs:complexType>
       </xs:element>

      <xs:element name="T_SitiSoftware">
        <xs:complexType>
          <xs:attribute name="IDSito" type="xs:int" use="required" />
          <xs:attribute name="IDSoftware" type="xs:int" use="required" />
        </xs:complexType>
       </xs:element>

    </xs:choice>
    </xs:complexType>

    <xs:unique name="Constraint1">
      <xs:selector xpath=".//T_Software" />
      <xs:field xpath="@ID" />
    </xs:unique>

   <xs:unique name="T_Siti_Constraint1" msdata:ConstraintName="Constraint1">
      <xs:selector xpath=".//T_Siti" />
      <xs:field xpath="@ID" />
    </xs:unique>

    <xs:keyref name="Constraint2" refer="Constraint1" msdata:ConstraintOnly="true">
      <xs:selector xpath=".//T_SitiSoftware" />
      <xs:field xpath="@IDSoftware" />
    </xs:keyref>

    <xs:keyref name="T_SitiSoftware_Constraint1" refer="T_Siti_Constraint1"
            msdata:ConstraintName="Constraint1"
            msdata:ConstraintOnly="true">
      <xs:selector xpath=".//T_SitiSoftware" />
      <xs:field xpath="@IDSito" />
    </xs:keyref>

   </xs:element>
  </xs:schema>
....
```

Codice XML

```
....
....
</NewDataSet>
```

4.6.2.3 Esempio di utilizzo di un Documento XML in una pagina .aspx

Per testare questa soluzione si è scelto un esempio abbastanza semplice a causa della complessità di questa soluzione.
Il seguente codice C# inserito all'interno di una pagina ASP .NET (file.aspx) sarà spiegato nelle sue parti più importanti per descriverne le funzionalità:

- Carica il file origine contenente XML e XSD in un oggetto DataSet di .NET che struttura i dati come un DB relazionale implementando i controlli e le relazioni espresse dallo Schema.

- Crea un oggetto XmlDataDocument di .NET che importa tutto il DataSet e lo indicizza secondo un astruttura ad albero data dal file XML di origine.

- Esegue una selezione tramite il linguaggio XPath che si muove all'interno dell'Ogetto XmlDataDocument.

- Copio i dati così ottenuti in una tabella temporanea costruita sui dati ottenuti dalla selezione.

- Inserisco la tabella in un DataGrid e la stampo a video.

```
<HTML>
	<HEAD>
		<link rel="stylesheet" href="../../stile/stilecorpo_area1.css">
	</HEAD>

<BODY	aLink=#000066 bgColor=#FFFFF4 leftMargin=0 link=#000066
	topMargin=0 vLink=#000066 marginwidth="0" marginheight="0">

<%@ Page Debug="True"%>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Globalization" %>
<%@ Import Namespace="System.Threading" %>
<%@ Import Namespace="System.Xml" %>

<SCRIPT  language="C#" runat="server">

protected int i;
protected DataRow miaRiga;
protected DataTable miaTab=new DataTable();
protected ArrayList mioAL = new ArrayList();
```

Dichiaro una variabile chiamata **miaRiga** di tipo riga che conterrà di volta in volta i valori delle righe relative alle tabelle contenute nel file XML. La variabile **miaRiga** non viene inizializzata in modo che si adatti automaticamente ai tipi di dato che deciderò di inserire rendendo così il codice ancora più flessibile e dinamico.

Allo stesso modo dichiaro una variabile chiamata **mioAL** che conterrà una lista di array, cioè una lista di righe del tipo miaRiga; questi dati saranno poi trasferiti nella terza variabile creata, ma non inizializzata, denominati **miaTab**; questa variabile conterrà il risultato finale delle mie operazioni cioè la tabella risultante da una selezione fatta sui dati contenuti nel "database" XML.

```
void Page_Load(object sender, EventArgs e)
{
String urlxml = "C:/Yuri/XML/gestione_software/db/Documentazioni_Software.xml";
DataSet ds = new DataSet() ;
```

Dichiaro una stringa denominata **urlxml** che indica il percorso assoluto dove risiede fisicamente il mio documento XML che funga da database. Dopo di chè definisco una variabile denominata ds di tipo DataSet (che sarà ampliamente spiegata in seguito) che porterà in cache i dati contenuti nel documento XML, mantenendo sia il contenuto informativo che le relazioni.

```
ds.ReadXml(urlxml);
XmlDataDocument xmlDoc = new XmlDataDocument(ds);
```

tramite il metodo **ReadXml** leggo il contenuto informativo del documento XML e lo porto in memoria mantenendo inalterati si a i dati che le relazioni; dopo di chè dichiaro una nuova variabile di tipo **XmlDataDocument** che importando i dati contenuti nel **DataSet** sovrappone ad esso la struttura del DOM.

```
XmlNodeList ListaNodi =
mlDoc.DocumentElement.SelectNodes("//T_Software[@ID<2]");
// Dichiaro una riga e una colonan temporane per la duplicazione dei dati
DataRow rigaTabella;
DataColumn colonna;
```

Una volta che ho a disposizione in cache i dati rappresentati secondo lo schema DOM posso navigare al loro interno, in modo estremamente veloce ed intuitivo tramite istruzioni XPath (precedentemente spiegato nel dettaglio) servendomi dei metodi e delle classi messi a disposizione dal Framework .NET per la gestione e la modifica degli elementi XML.
In questo caso vengono selezionati tutti i tag con l'attributo IDAgenete < 3, il risultato di questa interrogazione XPath viene scaricato nella variabile **ListaNodi** opportunamente dichiarata di tipo **XmlNodeList**.

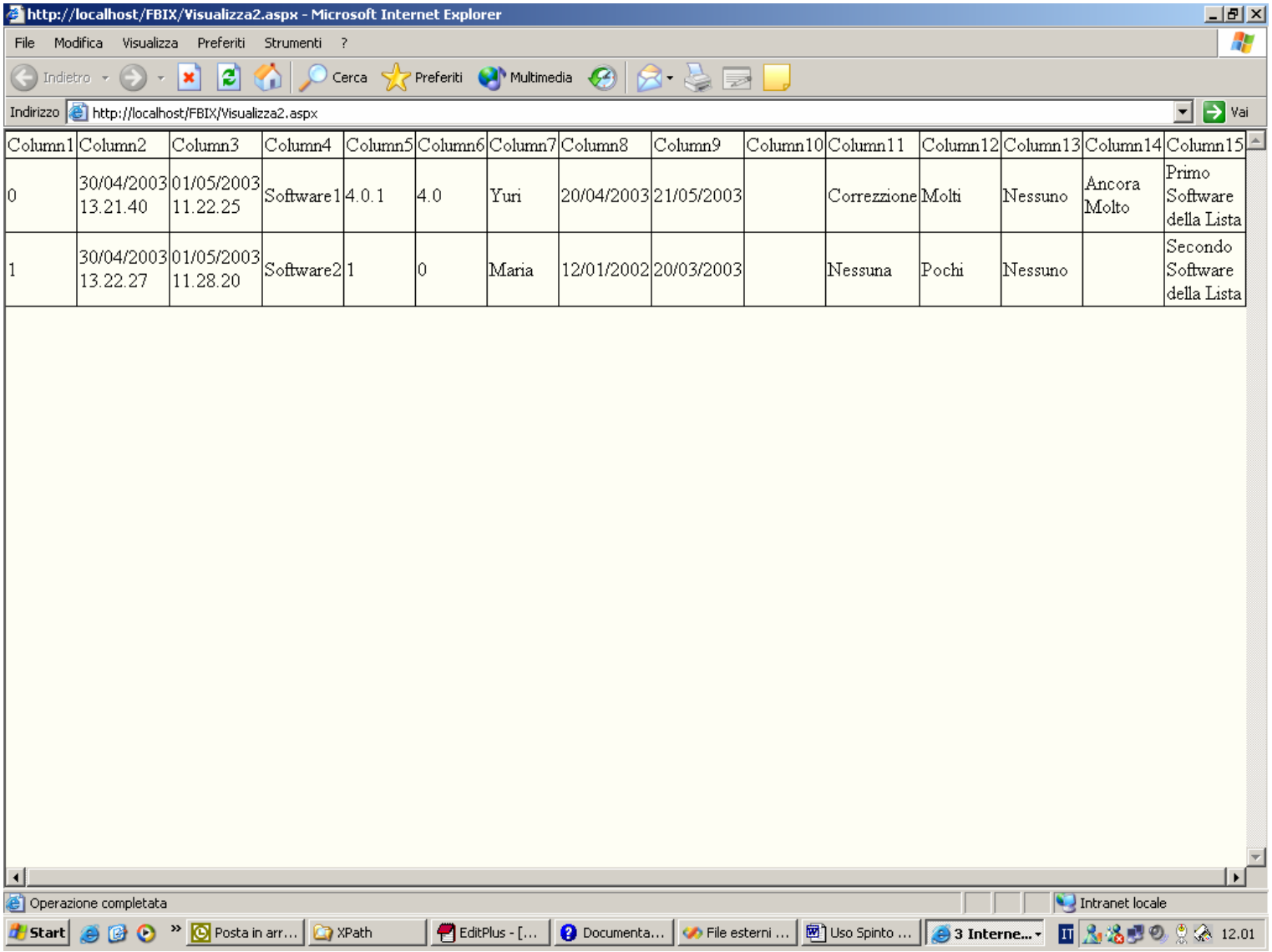
```
foreach (XmlNode mioNodo in ListaNodi)
{// Creo un puntatore alla riga della tabella sorgente per Ogni Elemento Trovato
miaRiga = xmlDoc.GetRowFromElement((XmlElement)mioNodo);
```

```
// Se la Mia Tabella temporanea è ancora vuote creo la struttura per contenere le righe
if (miaTab.Columns.Count==0)
        {       // Ciclo da zero al numero di colonne presenti nella riga
        for (int i=0;i<miaRiga.Table.Columns.Count;i++)
                {// Inizializzo la Colonna
                colonna = new DataColumn();

                // Do alla colonna il tipo della riga sorgente
                colonna.DataType = miaRiga.Table.Columns[i].DataType;

                // Aggiungo la colonna alla tabella avendo così la
                // struttura della riga sorgente
                miaTab.Columns.Add(colonna);
                }
        }
rigaTabella = miaTab.NewRow();
for (int i=0;i<miaRiga.Table.Columns.Count;i++)
        {// Riempio la mia Riga temporanea coi valori della riga sorgente
        rigaTabella[i] = miaRiga[i];}
// Aggiungo la riga alla mia tabella
miaTab.Rows.Add(rigaTabella);}
```

Il primo di questo ciclo For ha il compito di preparare le variabili temporanee per contenere i dati prelevati dal documento XML; è infatti necessario definire i tipi di dato delle variabili che verranno poi definite dal secondo ciclo per prepara in memoria una struttura dati adeguata ai dati letti dal documento XML. Il secondo ciclo serve per inserire i dati veri e propri nella tabella presente in memoria e precedentemente inizializzata e formattata.
In questo modo ho ottenuto i dati voluti e gli ho rappresentati in memoria; ora i dati sono pronti per essere utilizzati per qualsiasi operazione se dovesse essere necessaria una modifica della sorgente sarà possibile riscrivere il file XML modificato attraverso gli opportuni oggetti.

Le ultime righe di codice si occupano della visualizzazione a video dei dati rappresentati in memoria, dimostrando così la giusta rappresentazione dell'informazione acquisita attraverso il documento XML utilizzato come database. La visualizzazione a video della tabella è affidata alla Classe **DataSurce** e **DataGrid** del Framework .NET.

```
        // Riempio il DataGrid con al mia Tabella temporanea
        dg.DataSource=(miaTab);
        // Stampo il DataGrid
        dg.DataBind();}

</SCRIPT>
<ASP:Datagrid id="dg" runat="server" autogeneratecolumns="True"
                BorderColor="black" BorderWidth="1"/>
<ASP:DataList id="DL" horizontalAlign="center" runat="server">
</ASP:datalist></BODY>
</HTML>
```



Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11	Column12	Column13	Column14	Column15
0	30/04/2003 13.21.40	01/05/2003 11.22.25	Software1	4.0.1	4.0	Yuri	20/04/2003	21/05/2003		Correzzione	Molti	Nessuno	Ancora Molto	Primo Software della Lista
1	30/04/2003 13.22.27	01/05/2003 11.28.20	Software2	1	0	Maria	12/01/2002	20/03/2003		Nessuna	Pochi	Nessuno		Secondo Software della Lista

Capitolo 4, Figura 6: Output della Pagina ASP .NET che Implementa XPath

4.6.2.4 Valutazione del Servizio

Questa soluzione offre i seguenti vantaggi:

- Velocità e potenza nelle selezioni dello strumento XPath;
- Intuitività dello Strumento XPath;
- Possibilità di una gestione separata dei dati selezionati.

Purtroppo però questa implementazione soffre dei seguenti svantaggi:

- Codice complesso;
- Ridondanza dei Dati;
- Duplicazione dell'intero DataSet;
- Duplicazione dei Dati delle Selezioni in tabelle temporanee.

4.6.3 Soluzione tramite l’uso del DataSet

4.6.3.1 Introduzione al DataSet

La classe **DataSet** rappresenta una cache in memoria dei dati recuperati da un database, costituisce quindi un componente fondamentale dell'architettura ADO .NET.
L'oggetto **DataSet** è costituito da un insieme di oggetti **DataTable** che è possibile porre in relazione tra loro mediante oggetti **DataRelation**. È inoltre possibile applicare l'integrità dei dati nell'oggetto **DataSet** utilizzando gli oggetti **UniqueConstraint** e **ForeignKeyConstraint**. Mentre gli oggetti **DataTable** contengono i dati, l'insieme **DataRelationCollection** consente di spostarsi all'interno della gerarchia delle tabelle, che sono contenute in un insieme **DataTableCollection** cui si accede tramite la proprietà Tables.
Quando si accede agli oggetti **DataTable**, è opportuno ricordare che tali oggetti prevedono la distinzione tra maiuscole e minuscole in modo condizionale: se ad esempio un oggetto **DataTable** viene denominato "miodatatable" e un altro viene denominato "Miodatatable", alla stringa utilizzata per eseguire la ricerca di una delle tabelle verrà applicata la distinzione tra maiuscole e minuscole; se tuttavia esiste soltanto "miodatatable" e non "Miodatatable", alla stringa di ricerca non verrà applicata alcuna distinzione tra maiuscole e minuscole.
Tramite l’oggetto **DataSet** i dati e gli schemi vengono letti e scritti come documenti XML, dopo di chè i dati e gli schemi possono essere trasferiti tramite HTTP e utilizzati da qualsiasi applicazione su qualsiasi piattaforma con supporto XML. È possibile salvare lo schema come schema XML mediante il metodo **WriteXmlSchema**, mentre lo schema e i dati possono essere salvati nello stesso documento XML utilizzando il metodo **WriteXml**.
Per leggere un documento XML che includa sia lo schema sia i dati, utilizzare il metodo **ReadXml**.
In una tipica implementazione a più livelli per creare e aggiornare un oggetto **DataSet** e per aggiornare i dati originali, le principali operazioni da svolgere sono le seguenti:

- Generazione e immissione in ciascun oggetto **DataTable** di un oggetto DataSet dei dati provenienti da un'origine dati mediante **SqlDataAdapter** o **OleDbDataAdapter**.

- Modifica dei dati nei singoli oggetti **DataTable** mediante aggiunta, aggiornamento o eliminazione degli oggetti **DataRow**.

- Chiamata al metodo **GetChanges** per creare un secondo oggetto DataSet in cui sono presenti solo le modifiche ai dati.

- Chiamata al metodo **Update** di **SqlDataAdapter** o di **OleDbDataAdapter** passando come argomento il secondo oggetto DataSet.

- Chiamata al metodo **Merge** per unire le modifiche del secondo oggetto DataSet al primo oggetto.

- Chiamata al metodo **AcceptChanges** sull'oggetto DataSet. In alternativa, chiamata al metodo **RejectChanges** per annullare le modifiche.

4.6.3.2 Documento XML orientato all'uso del DataSet

Il documento XML che si intende utilizzare come rappresentazione fisica dei, può essere scritto nello stesso modo della precedente soluzione, ma se si sceglie una rappresentazione orientata verso l'uso degli elementi (e non degli attributi) i controlli possibili sui dati sono molto più completi ed esaurienti (come visto nei capitoli precedenti).

Si risparmiano anche le righe di codice .NET che dovrebbero definire l'uso degli attributi alleggerendo così il codice C#, vengono tolte tutte le istruzioni di questo tipo:

IDSoftware.ColumnMapping=MappingType.Attribute;

Questa rappresentazione rende il codice XML meno intuitiva se letta da un utente, ma permette una maggiore descrizione dei dati; è utile comunque ricordare che la funzione principale di un documento XML è quella di fornire dati correttamente formati a un applicazione non a un utente umano.
L'istruzione iniziale standalone="yes" indica che il file XML contiene al suo interno anche l'XSD Schema per mantenera la massima compatibilità con la Classe DatSet che se utilizzata in modo automatico scrive in uno stesso file sia il documento XML che il suo XSD Schema.

```
<?xml version="1.0" standalone="yes"?>

<NewDataSet>

<xs:schema    id="NewDataSet"
              xmlns=""
              xmlns:xs="http://www.w3.org/2001/XMLSchema"
              xmlns:msdata="urn:schemasmicrosoft-com:xml-msdata">
….
….
</xs:schema>

<T_Software>
   <ID>0</ID>
   <DataInserimento>2003-05-01T12:27:13.6183296+02:00</DataInserimento>
   <DataUltimaModifica>2003-05-01T12:30:13.2065648+02:00</DataUltimaModifica>
   <NomeSoftware>Software1</NomeSoftware>
   <Versione>4.0.1</Versione>
   <VersionePrecedente>4.0</VersionePrecedente>
   <Sviluppatore>Yuri</Sviluppatore>
   <DataInizioSviluppo>20/04/2003</DataInizioSviluppo>
   <DataFineSviluppo>21/05/2003</DataFineSviluppo>
   <Sviluppo />
   <Correzioni>Correzzione</Correzioni>
   <Test>Molti</Test>
   <Errori>Nessuno</Errori>
   <DaSviluppare>Ancora Molto</DaSviluppare>
```

```
    <Note>Primo Software della Lista</Note>
</T_Software>

<T_Software>
    <ID>1</ID>
    <DataInserimento>2003-05-01T12:28:23.7191296+02:00</DataInserimento>
    <DataUltimaModifica>2003-05-01T12:30:21.5485600+02:00</DataUltimaModifica>
    <NomeSoftware>Software2</NomeSoftware>
    <Versione>1</Versione>
    <VersionePrecedente>0</VersionePrecedente>
    <Sviluppatore>Maria</Sviluppatore>
    <DataInizioSviluppo>12/01/2002</DataInizioSviluppo>
    <DataFineSviluppo>20/03/2003</DataFineSviluppo>
    <Sviluppo />
    <Correzioni>Nessuna</Correzioni>
    <Test>Pochi</Test>
    <Errori>Nessuno</Errori>
    <DaSviluppare />
    <Note>Secondo Software della Lista</Note>
</T_Software>

<T_Software>
    <ID>2</ID>
    <DataInserimento>2003-05-01T12:29:32.8885904+02:00</DataInserimento>
    <DataUltimaModifica>2003-05-01T12:30:27.4770848+02:00</DataUltimaModifica>
    <NomeSoftware>Software3</NomeSoftware>
    <Versione>12</Versione>
    <VersionePrecedente>10.1</VersionePrecedente>
    <Sviluppatore>Daniele</Sviluppatore>
    <DataInizioSviluppo>03/10/2001</DataInizioSviluppo>
    <DataFineSviluppo>01/04/2003</DataFineSviluppo>
    <Sviluppo />
    <Correzioni>Alcune</Correzioni>
    <Test>Nessuno</Test>
    <Errori>Alcuni</Errori>
    <DaSviluppare />
    <Note>Terzo Software della Lista</Note>
</T_Software>

<T_Siti>
   <ID>0</ID>
   <NomeSito>Sito1</NomeSito>
</T_Siti>

<T_Siti>
    <ID>1</ID>
    <NomeSito>Sito2</NomeSito>
</T_Siti>
```

```
<T_Siti>
   <ID>2</ID>
   <NomeSito>Sito3</NomeSito>
</T_Siti>

<T_SitiSoftware>
        <IDSito>0</IDSito>
        <IDSoftware>0</IDSoftware>
</T_SitiSoftware>


<T_SitiSoftware>
        <IDSito>1</IDSito>
        <IDSoftware>1</IDSoftware>
</T_SitiSoftware>

<T_SitiSoftware>
        <IDSito>2</IDSito>
        <IDSoftware>2</IDSoftware>
</T_SitiSoftware>

</NewDataSet>
```

L'XSD Schema esegue molti controlli sul file sorgente XML (come visto nei capitoli precedenti) rendendo il documento XML sorgente abbastanza sicuro per essere utilizzato come database.
I controlli più importanti eseguiti sulla struttura e sui dati del documento sorgente XML sono i seguenti:

- L'XSD Schema importa il NameSpace: xmlns:msdata="urn:schemas-microsoft-com:xml-msdata" definito dalla microsoft che comprende definizioni di elementi e di tipi propri di .NET.

- Definisce come deve essere la struttura dei nodi:
  Il DataBase Virtuale "NewDataSet" (elemento di root) potrà contenere un numero non specificato di figli (<xs:choice maxOccurs = "unbounded">) ma solo dei tipi:

  ```
  <xs:element name="T_Software">,
  <xs:element name="T_Siti">,
  <xs:element name="T_SitiSoftware">.
  ```

  Qualsiasi altro tag che venisse riscontrato nel documento XML come figlio dell'elemento <NewDataSet> darebbe luogo a un errore e visto che i figli non prevedono al loro interno altri elementi complessi, se fossero rilevati nel documento sorgente XML tag non appartenenti a elementi semplici all'interno dei figli del root lo schema genererebbe un errore.

- Grazie al Tipo msdata:AutoIncrement="true" specificato dal NameSpace della Microsoft è possibile generare elementi autoincrementati.

- L'XSD Schema controlla che i dati inseriti siano di uno specifico tipo (interi, Stringhe, o altro), se nel documento sorgente XML fosse scritto un dato di un tipo diverso lo Schema genererebbe un errore.

- L'XSD Schema controlla che gli elementi dichiarati come non NULL debbano comparire nel file XML almeno una volta tramite l'istruzione: minOccurs="0".

- All'interno dell'elemento di root vengono specificati anche i vincoli dei figli in esso contenuti: tramite il Costrutto Unique vengono dichiarati unici gli ID delle tabelle (è possibile anche definire chiavi primarie, di una o più colonne tramite il tipo contenuto nel NameSpace Microsoft msdata:PrimaryKey="true") e tramite il costrutto Keyref vengono implementate le Foreign Key.
  Questi controlli garantiscono l'integrità delle relazioni e danno errori se il File XML non rispetta i vincoli così definiti.

```
<?xml version="1.0" standalone="yes"?>

<NewDataSet>

<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

   <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="it-IT">
    <xs:complexType>
     <xs:choice maxOccurs="unbounded">

       <xs:element name="T_Software">
        <xs:complexType>
         <xs:sequence>

           <xs:element name="ID" msdata:AutoIncrement="true" type="xs:int" />
           <xs:element name="DataInserimento" type="xs:dateTime" />
           <xs:element name="DataUltimaModifica" type="xs:dateTime" />
           <xs:element name="NomeSoftware" type="xs:string" />
           <xs:element name="Versione" type="xs:string" />
           <xs:element name="VersionePrecedente" type="xs:string" minOccurs="0" />
           <xs:element name="Sviluppatore" type="xs:string" />
           <xs:element name="DataInizioSviluppo" type="xs:string" minOccurs="0" />
           <xs:element name="DataFineSviluppo" type="xs:string" minOccurs="0" />
           <xs:element name="Sviluppo" type="xs:string" minOccurs="0" />
           <xs:element name="Correzioni" type="xs:string" minOccurs="0" />
           <xs:element name="Test" type="xs:string" minOccurs="0" />
           <xs:element name="Errori" type="xs:string" minOccurs="0" />
           <xs:element name="DaSviluppare" type="xs:string" minOccurs="0" />
```

```
            <xs:element name="Note" type="xs:string" minOccurs="0" />

          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name="T_Siti">
        <xs:complexType>
          <xs:sequence>

            <xs:element name="ID" msdata:AutoIncrement="true" type="xs:int" />
            <xs:element name="NomeSito" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:element name="T_SitiSoftware">
        <xs:complexType>
          <xs:sequence>

            <xs:element name="IDSito" type="xs:int" />
            <xs:element name="IDSoftware" type="xs:int" />

          </xs:sequence>
        </xs:complexType>
      </xs:element>

    </xs:choice>
  </xs:complexType>

  <xs:unique name="Constraint1">
    <xs:selector xpath=".//T_Software" />
    <xs:field xpath="ID" />
  </xs:unique>

  <xs:unique name="T_Siti_Constraint1" msdata:ConstraintName="Constraint1">
    <xs:selector xpath=".//T_Siti" />
    <xs:field xpath="ID" />

  </xs:unique>

<xs:keyref name="T_SitiSoftware_Constraint1" refer="T_Siti_Constraint1"
           msdata:ConstraintName="Constraint1" msdata:ConstraintOnly="true">
    <xs:selector xpath=".//T_SitiSoftware" />
    <xs:field xpath="IDSito" />
  </xs:keyref>

  <xs:keyref name="Constraint2" refer="Constraint1" msdata:ConstraintOnly="true">
```

```
      <xs:selector xpath=".//T_SitiSoftware" />
      <xs:field xpath="IDSoftware" />
    </xs:keyref>

  </xs:element>
 </xs:schema>
....
```
Codice XML
```
....
</NewDataSet>
```

4.6.3.3 Esempio di utilizzo di un Documento XML in una pagina .aspx e tramite codice C#

Innanzi tutto, viene dichiarato un **DataSet** che conterrà il database letto dal file XML.

```
protected  DataSet DB ;
```

Per inizializzare il DataSet tramite un documento sorgente XML è stata realizzata una funzione di caricamento, denominata **LoadDB** che controlla come prima cosa se esiste un documento XML e nel caso esista carica dalla sorgente i dati all'interno del DataSet; altrimenti viene creato il novo documento XML vuoto nel DataSet e sarà poi il DataSet a creare un documento XML vero e proprio che fungerà all'avvio successivo da database.

Nel caso si debba creare un nuovo database XML, si utilizzeranno i tipi di dato **DataColumn** e **DataTable** per creare in memoria prima le colonne di ogni tabella poi la tabella stessa e tramite funzioni apposite (**MiaColonna.ColumnName**, **MiaColonna.AutoIncrement**, **MiaColonna.AllowDBNull**, **MiaTabellaabella.Columns.Add**) queste strutture verranno riempite coi dati voluti precedentemente definiti tramite la funzione **MiaColonna.DataType** e **System.Type.GetType("Tipo di Dato")**.

Una Volta che queste operazioni saranno state fatte per tutte le tabelle che dovranno essere contenute nel database verranno definite le relazioni tramite apposite istruzioni come:

```
MioDB.Tables["MiaTabella1"].Constraints.Add(new
ForeignKeyConstraint(MiaTabella2,MiaColonna2));
```

A questo punto il database è completamente formato e formattato in memoria all'interno dell'oggetto DataSet; possiamo quindi scriverlo sul disco tramite l'istruzione:

```
MioDB.WriteXml(Server.MapPath(@"..\..\db\Documentazioni_Software.xml"),
XmlWriteMode.WriteSchema);
```

Abbiamo così ottenuto un database strutturato come documento XML.

```
/*--- CREA STRUTTURA DB ---*/
protected void LoadDB()
{
/*--  Dichiaro il DB --*/
DB = new DataSet();

/*-- Controllo l'esistenza della fonte di Dati XML --*/
if (!File.Exists(Server.MapPath(@"..\..\db\Documentazioni_Software.xml")))
                {
                /* -- Se il DB XML non esiste ne creo uno vuoto con solo
                la struttura e quindi lo Schema XSD --*/

                DataTable Tabella;
                DataColumn Colonna;

                //STRUTTURA DELLA TABELLA
                //DELLE DOCUMENTAZIONI T_SOFTWARE

                Tabella = new DataTable("T_Software");

                DataColumn IDSoftware = new DataColumn();
                IDSoftware.DataType = System.Type.GetType("System.Int32");
                IDSoftware.ColumnName = "ID";
                IDSoftware.AutoIncrement = true;
                IDSoftware.AllowDBNull = false;
                Tabella.Columns.Add(IDSoftware);

                Colonna = new DataColumn();
                Colonna.DataType = System.Type.GetType("System.DateTime");
                Colonna.ColumnName = "DataInserimento";
                Colonna.AllowDBNull = false;
                Tabella.Columns.Add(Colonna);
                Colonna = new DataColumn();
                Colonna.DataType = System.Type.GetType("System.DateTime");
                Colonna.ColumnName = "DataUltimaModifica";
                Colonna.AllowDBNull = false;
                Tabella.Columns.Add(Colonna);

                ……
                ……
                ……

                Colonna = new DataColumn();
                Colonna.DataType = System.Type.GetType("System.String");
                Colonna.ColumnName = "Note";
                Colonna.AllowDBNull = true;
                Tabella.Columns.Add(Colonna);
```

```
            DB.Tables.Add(Tabella);

            //STRUTTURA DELLA TABELLA DEI SITI T_SITI
            Tabella = new DataTable("T_Siti");

            DataColumn IDSito = new DataColumn();
            IDSito.DataType = System.Type.GetType("System.Int32");
            IDSito.ColumnName = "ID";
            IDSito.AutoIncrement = true;
            IDSito.AllowDBNull = false;
            Tabella.Columns.Add(IDSito);

            Colonna = new DataColumn();
            Colonna.DataType = System.Type.GetType("System.String");
            Colonna.ColumnName = "NomeSito";
            Colonna.AllowDBNull = false;
            Tabella.Columns.Add(Colonna);

            DB.Tables.Add(Tabella);

            //STRUTTURA DELLA TABELLA
            // SITI-SOFTWARE T_SITISOFTWARE

            Tabella = new DataTable("T_SitiSoftware");

            DataColumn FK_IDSito = new DataColumn();
            FK_IDSito.DataType = System.Type.GetType("System.Int32");
            FK_IDSito.ColumnName = "IDSito";
            FK_IDSito.AllowDBNull = false;
            Tabella.Columns.Add(FK_IDSito);

            DataColumn FK_IDSoftware = new DataColumn();
            FK_IDSoftware.DataType = System.Type.GetType("System.Int32");
            FK_IDSoftware.ColumnName = "IDSoftware";
            FK_IDSoftware.AllowDBNull = false;
            Tabella.Columns.Add(FK_IDSoftware);

            DB.Tables.Add(Tabella);

        DB.Tables["T_SitiSoftware"].Constraints.Add(new
        ForeignKeyConstraint(IDSito,FK_IDSito));
        DB.Tables["T_SitiSoftware"].Constraints.Add(new
        ForeignKeyConstraint(IDSoftware,FK_IDSoftware));

            /*-- Scrivo tramite il DataSet il file XML e il suo XSD --*/
```

```
        DB.WriteXml(Server.MapPath(@"..\..\db\Documentazioni_Software.xml"),XmlWrite
        Mode.WriteSchema);
                        }
                        else
                        {

        /*-- Se il file XML e il suo XSD esistono li carico nel DataSet --*/
        DB.ReadXml(Server.MapPath(@"..\..\db\Documentazioni_Software.xml"),XmlRead
        Mode.ReadSchema);
                        }
        /*-- Nel caso abbia creato io il DB inserisco i cambiamenti al DataSet --*/
        DB.AcceptChanges();
}//Fine LoadDB
```

Quando devo localizzare una riga non potendo contare su un istruzione SQL del tipo:

```
        SELECT * FROM nomeTab WHERE ID=mioID
```

Devo creare una funzione che cerchi i dati all'interno del DataSet e restituisca una tabella che contenga le righe selezionate.

```
/*--- LOCALIZZA RECORD ---*/
protected void LocalizzaRecord(string nomeTabella)
{
//TROVO L'INDICE DELLA RIGA SELEZIONATA
// Parto dalla riga zero e vado fino all'ultima righa della tabella
        for (int i=0;i<DB.Tables[nomeTabella].Rows.Count;i++)
                {
                // Controllo che il CAST a Intero del valore del campo ID
// di ogni righa Soddisfa il mioID
if ((int)DB.Tables[nomeTabella].Rows[i]["ID"] == idRecord)
        {
        mioRecord = DB.Tables[nomeTabella].Rows[i];
        break;
        }
                }
                if (mioRecord == null)
                {
                throw new ArgumentException("Non è stato trovato alcun Record con il
                parametro 'ID' selezionato");
                }
}//Fine LocalizzaRecord
```

Quando devo Inserire, Modificare, Copiare o Eliminare un Record non disponendo di istruzioni SQL come: INSERT, UPDATE o DELETE; devo creare funzioni apposite che sfruttino le potenzialità del DataSet e poi riscrivere tramite il DataSet le modifiche al file XML.

```
/*--- SALVA NEL DB ---*/
```

```
protected override void SalvaNelDB(object o,EventArgs e)
{
if (Action == "Inserisci" || Action == "Copia")
        {
        //Se si tratta si un inserimento o di una copia prepare una riga nuova da riempire
        mioRecord = DB.Tables["T_Software"].NewRow();
        mioRecord["DataInserimento"] = DateTime.Now;
        }

// Riempio la mia riga temporanea
mioRecord["DataUltimaModifica"] = DateTime.Now;
mioRecord["NomeSoftware"] = InputNomeSoftware.Value;
mioRecord["Versione"] = InputVersione.Value;
mioRecord["VersionePrecedente"] = InputVersionePrecedente.Value;
mioRecord["Sviluppatore"] = InputSviluppatore.Value;
mioRecord["DataInizioSviluppo"] = InputDataInizioSviluppo.Value;
mioRecord["DataFineSviluppo"] = InputDataFineSviluppo.Value;
mioRecord["Sviluppo"] = InputSviluppo.Value;
mioRecord["Correzioni"] = InputCorrezioni.Value;
mioRecord["Test"] = InputTest.Value;
mioRecord["Errori"] = InputErrori.Value;
mioRecord["DaSviluppare"] = InputDaSviluppare.Value;
mioRecord["Note"] = InputNote.Value;

if (Action == "Elimina")
        {
        // Se si tratta di un eliminazione cancello al DataSet la riga selezionata
        DB.Tables["T_Software"].Rows.Remove(mioRecord);
        }

else
        {
        DB.Tables["T_Software"].AcceptChanges();
        }

if (Action == "Inserisci" || Action == "Copia")
        {
        DB.Tables["T_Software"].Rows.Add(mioRecord);
        }

//ELIMINO I SITI A CUI E' ASSOCIATO
for (int i=0;i<DB.Tables["T_SitiSoftware"].Rows.Count;i++)
        {
        if ((int)DB.Tables["T_SitiSoftware"].Rows[i]["IDSoftware"] ==
(int)mioRecord["ID"])
                {
                DB.Tables["T_SitiSoftware"].Rows[i].Delete();
                }
```

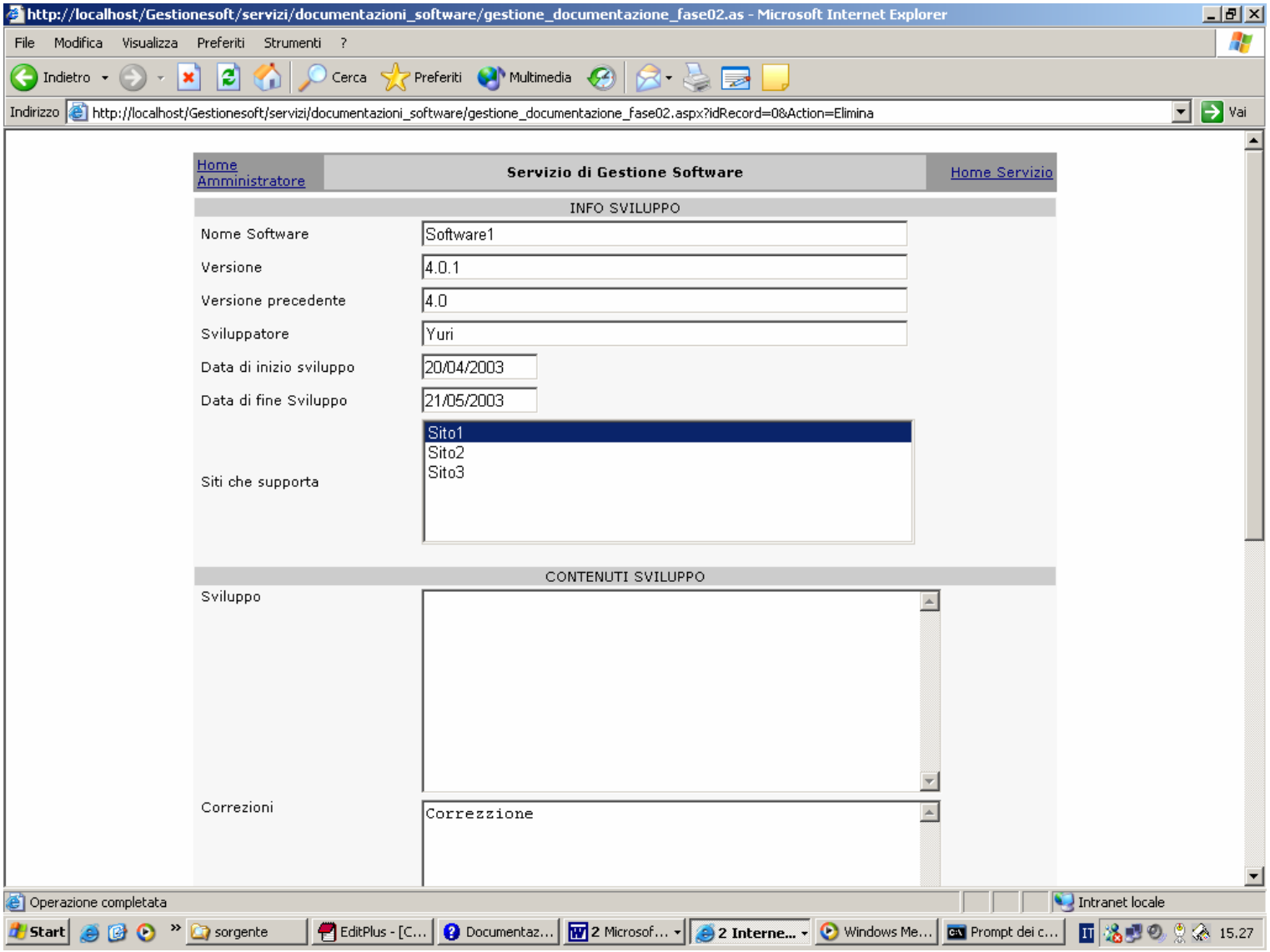
```
        }

//INSERISCO I SITI A CUI E' ASSOCIATO
DataRow recordSito = null;
for (int i=0;i<BoxSiti.Items.Count;i++ )
        {
        if (BoxSiti.Items[i].Selected)
                {
                recordSito = DB.Tables["T_SitiSoftware"].NewRow();
                recordSito["IDSoftware"] = mioRecord["ID"];
                recordSito["IDSito"] = Convert.ToInt32(BoxSiti.Items[i].Value);
                DB.Tables["T_SitiSoftware"].Rows.Add(recordSito);
                }}
// Faccio accettare I cambiamenti fatti al DataSet
DB.Tables["T_SitiSoftware"].AcceptChanges();

// Scrivo il nuovo DataSet che contiene la versione aggiornata del Db sul file XML
DB.WriteXml(Server.MapPath(@"..\..\db\Documentazioni_Software.xml"),XmlWriteMode.
WriteSchema);

Response.Redirect("gestione_documentazione_fase01.aspx?Action="+Action);
}// Fine SalvaNelDB
```

Per la visualizzazione dei dati ci appoggiamo a dei DataView che serviranno per creare del viste e a dei DataGrid che si occuperanno della visualizzazione vera e propria dei dati.

```
public class Gestione_Documentazione_Fase01 : PaginaBase_Amministratore
{
// Dichiaro un DataGrid che conterrà poi I dati per la visualizzaizone
protected DataGrid DgdDocumentazione = new DataGrid();

//Dichiaro una string ache servirà nella paginazione
protected static string Sorting;

// Dichiaro un DataView  che conterrà le viste
protected static DataView DvDocumentazione;

/*---- PAGE_LOAD ---*/
public void Page_Load(object o,EventArgs e)
{
CreaTabellaDocumentazione();
}

/*---- CREA TABELLA DOCUMENTAZIONE ----*/
protected void CreaTabellaDocumentazione()
{
// Inserisco nel DataView la tabella T_Software contenuta nel DataSet DB
DvDocumentazione = DB.Tables["T_Software"].DefaultView;
```

```
// Inserisco nel DataGrid la vista contenuta nel DAtaView e la stampo a video con un Bind
DgdDocumentazione.DataSource = DvDocumentazione;
DgdDocumentazione.DataBind();
}//CreaTabellaDocumentazione

/*--- ORDINAMENTO ----*/
protected void Ordinamento(Object sender_ordina, DataGridSortCommandEventArgs e)
{
if (Sorting == e.SortExpression.ToString())
        {
        Sorting = e.SortExpression.ToString()+" desc";
        }
else
        {
        Sorting = e.SortExpression.ToString();
        }
DvDocumentazione.Sort = Sorting;
DgdDocumentazione.DataSource = DvDocumentazione;
DgdDocumentazione.DataBind();
}//Ordinamento

/*--- PAGINAZIONE -----*/
protected void Paginazione(object o, DataGridPageChangedEventArgs e)
{
DvDocumentazione.Sort = Sorting;
DgdDocumentazione.DataSource = DvDocumentazione;
DgdDocumentazione.CurrentPageIndex = e.NewPageIndex;
DgdDocumentazione.DataBind();
}// Fine della Paginazione
}
```

Capitolo 4, Figura 7: Output della Pagina ASP .NET che Implementa il DataSet –Visualizzazione-



Capitolo 4, Figura 8: Output della Pagina ASP .NET che Implementa il DataSet –Inserimento-

4.6.3.4 Valutazione del Servizio

Questa soluzione offre i seguenti vantaggi:

- Nessuna duplicazione dei Dati una volta importati in memoria;
- Ottima implementazione di controlli software sull'integrità dei dati;
- Semplicità del Codice.

Purtroppo però questa implementazione soffre dei seguenti svantaggi:

- Codice prolisso;
- Numerosi cicli a livello di codice per sostituire le interrogazioni SQL;

4.6.4 Valutazione Complessiva del Servizio

E' opportuno ricordare brevemente lo scopo che si voleva ottenere utilizzando lo strumento fornito dai documenti XML: un database che garantisse bassi costi e un'alta velocità sacrificando parte dei controlli.

Entrambe le soluzioni proposte soddisfano in parte questi propositi, ma la soluzione sviluppata tramite l'uso del DOM e XPath, pur garantendo selezioni veloci e intuitive, ha il grande problema della duplicazione dei dati.

Ricordiamo infatti che per riuscire a sfruttare al meglio le interrogazioni XPath il framework .NET deve prima cerarsi un immagine in memoria del database XML (Duplicando per la prima volta i dati fisici) dopo di ché sovrapporre all'intera struttura dal DataSet l'indicizzazione ad albero del DOM che di fatto sovrappone una maschera data dall'oggetto XmlDataDocument al preesistente DataSet (Seconda duplicazione di Dati); questi passaggi pur riducendo e semplificando di molto il codice delle applicazioni rallentano la velocità di accesso ai dati, rendono più difficili le operazioni di modifica dei dati stessi e caricano molto la memoria rallentando tutta l'applicazione.

E' preferibile quindi, utilizzare la soluzione sviluppata appoggiandosi completamente e solamente sull'oggetto DataSet del Framework .NET.
Questa implementazione rende più arduo il compito degli sviluppatori che dovranno utilizzare il codice sorgente per implementare molti controlli e molte funzioni che si occupino della gestione dei dati e del loro recupero all'interno del database rendendo il codice più prolisso e meno intuitivo (rispetto alla soluzione che utilizza XPath); questi sforzi però saranno premiati da un oculato utilizzo delle risorse e quindi da ottime prestazioni in termini di velocità che era il nostro obbiettivo primario.

4.7 Importazione ed esportazione dati in XML con SQL Server 2000 e MySQL 4.0.12

Come si può desumere dai capitoli precedenti la tecnologia messa a disposizione dai documenti XML e dagli strumenti ad esso collegati è particolarmente utile quando si devono scambiare dati fra differenti applicazioni.

L'XML è dunque un formato rigoroso e sufficientemente libero da permettere ad applicativi molto differenti tra loro (sviluppati anche con differenti linguaggi e tecnologie) di scambiarsi dati e informazioni senza nessuna difficoltà.

E' possibile tramite l'XML e gli strumenti forniti dal Framework .NET scambiare dati tra applicazioni ASP .NET è applicativi Microsoft Office come Excel e Access ed è ancora più semplice scambiare dati tramite documenti XML fra applicazioni ASP .NET e altre applicazioni Web (anche XML Web Service) oppure fra DBMS quali SQL Server 2000 e MySQL.

Per utilizzare l'XML come fonte di dati oppure come output per un database MySQL si possono scegliere due differenti soluzioni: la prima è affidarsi a un Client MySQL, quasi tutti i client MySQL gratuiti o a pagamento danno la possibilità agli sviluppatori di importare dati o esportarli tramite documenti XML semplici che dovranno poi essere correlati da strumenti di controllo (DTD, XSL o XSD Schema); l'altra opzione è quella di utilizzare come "ponte" fra i documenti XML e il database MySQL vero e proprio gli strumenti forniti dal Framework .NET (DataSet, XmlDocument, XmlDataDocument).

E' necessario scegliere una delle due differenti scelte a seconda delle proprie esigenze: la prima ipotesi è molto semplice ed intuitiva (i tool grafici dei Client sono orientati verso l'utente) e non richiede le conoscenze specifiche di XML da parte dello sviluppatore; questa ipotesi però deve essere utilizzata separatamente dal codice di una eventuale applicazione, l'importazione (o esportazione) dei dati in formato XML avviene offline dall'applicazione Web ed è a pieno carico dell'utente.

La seconda ipotesi richiede da parte dello sviluppatore una buona conoscenza dell'XML e degli oggetti del Framework .NET, ma ha il pregio di poter utilizzare documenti XML come input e output di un'applicazione Web in fase di esecuzione.

L'utilizzo di documenti in formato XML come input o come output sono invece completamente integrati nei servizi offerti da SQL Server 2000 che permette allo sviluppatore di accedere all'XML tramite tool grafici del Client (esattamente come i tool dei Client MySQL), ma anche di scrivere automaticamente i risultati di una Query in formato XML tramite istruzioni T-SQL appositamente sviluppate come: FOR XML e OPEN XML.

XML è quindi uno strumento utile ed indispensabile per l'interoperabilità fra differenti applicazioni e la sua utilità diventa di giorno in giorno più importante via via che le applicazioni Web conquistano spazio sul mercato.

Esempio di dati esportati tramite XML

```
<?xml version="1.0" standalone="yes"?>

<DATAPACKET>
        <ROWDATA>

                <ROW ID="1" datainserimento="12/06/2002 16.02.00" multiportal="1"/>
                <ROW ID="2" datainserimento="01/30/2003 10.35.00" multiportal="1"/>
                <ROW ID="3" datainserimento="01/30/2003 10.35.00" multiportal="1"/>
                <ROW ID="4" datainserimento="01/30/2003 10.35.00" multiportal="2"/>
                <ROW ID="5" datainserimento="01/30/2003 10.35.00" multiportal="2"/>
                <ROW ID="6" datainserimento="01/30/2003 10.35.00" multiportal="2"/>
                <ROW ID="7" datainserimento="01/30/2003 10.35.00" multiportal="3"/>
                <ROW ID="8" datainserimento="01/30/2003 10.35.00" multiportal="3"/>
                <ROW ID="9" datainserimento="01/30/2003 10.35.00" multiportal="3"/>

        </ROWDATA>
</DATAPACKET>
```

5. XML Web Service

5.1 WSDL 1.1 Web Service Description Language

5.1.1 Introduzione

WSDL (Web Service Description Language) è uno standard fornito dal consorzio W3C ed è un formato XML per descrivere servizi Web come collezioni di endpoint capaci di scambiarsi messaggi contenenti informazioni sia orientate verso documenti si informazioni procedurali.

Sia le operazioni che i messaggi sono descritti in modo completamente astratto, sarà quindi il legame tra operazioni e messaggio con il reale protocollo di comunicazione ed il formato dei dati scambiati che definirà un endpoint.

La caratteristica più importante del WSDL è quella di non introdurre altri linguaggi, ma di supportare XML e XSD Schema e garantendo una forte estensibilità è in grado di adattarsi a qualsiasi situazione venga introdotta per seguire lo sviluppo delle applicazioni Web; WSDL garantisce quindi un valido e comune meccanismo di binding tramite il quale varie applicazioni possono dialogare tra loro utilizzando una piattaforma alla quale possono connettersi utilizzando il metodo di comunicazione più opportuno.
Generalmente i protocolli di comunicazioni più utilizzati per dialogare con strutture WSDL sono: SOAP, HTTP GET/POST e MIME.

5.1.2 Servizi WSDL

La grammatica definita da WSDL definisce dei **services** come collezioni di network endpoints o **ports**. Garantendo la completa astrazione della definizione delle operazioni e dei messaggi, WSDL separa l'effettivo protocollo di comunicazione dal formato dei dati utilizzati; in questo modo è possibile riutilizzare le definizioni astratte dei dati scambiati, **messages** e le collezioni astratte di operazioni (**operation**) altrimenti detti **port type**.

Lo specifico protocollo e il formato definito per un port type costituisce un **binding** riutilizzabile. Un port è dunque definito associando un indirizzo di rete ad un binding, a sua volta una collezione di port definisce un service.

5.1.3 Elementi WSDL

Un documento WSDL utilizza, per definire un servizio, i seguenti elementi:

Type
Gli elementi type racchiudono le definizioni di tipi di dati non predefiniti pertinenti ai messaggi scambiati.
Viene suggerito l'uso, come tipo di sistema canonico, di XSD Schema (anche se la decisione è lasciata agli sviluppatori) per garantire la massima interoperabilità e neutralità della piattaforma; XSD (come visto in precedenza) garantisce ottime funzioni descrittive e di controllo supportate da un elevata standardizzazione.
Un elemento type segue la seguente struttura all'interno di un documento WSDL:

```
<definitions .... >
        <types>
          <xsd:schema .... />*
        </types>
 </definitions>
```

Message
Gli elementi message consistono in una o più parti logiche definite **parts**; ognuna di esse è associata ad un tipo, a partire da un sistema di base, utilizzando attributi message-typing.
Il set degli attributi message-typing è estendibile, WSDL prevede due attributi message-tying da utilizzare con XSD Schema:

- **element**: riferisce ad un elemento XSD utilizzando un QName.
- **type**: riferisce ad un tipo di dato XSD semplice oppure ad un tipo di dato complesso utilizzando un QName.

L'attributo name del message, definisce un nome univoco valido per tutti gli elementi di tipo message del documento WSDL; a sua volta l'attributo name del part del message, definisce un nome univoco valido per tutte le part all'interno del message che si sta considerando.
Un elemento message segue la seguente struttura all'interno di un documento WSDL:

```
<definitions .... >
        <message name="nmtoken"> *
                <part name="nmtoken" element="qname"? type="qname"?/> *
        </message>
</definitions>
```

Nel caso di messaggi con più unità logiche si dovranno utilizzare più elementi part:

```
<definitions .... >
    <message name="nmtoken">
        <part name="nmtoken_1" element="qname_1"? type="qname_1"?/>
        <part name="nmtoken_2" element="qname_2"? type="qname_2"?/>
    </message>
</definitions>
```

Esiste una sintassi alternativa a quella appena vista che esprime gli stessi concetti, utilizzando però un solo elemento part; per fare questo è necessario prima definire un tipo complesso che implementi la scelta fra le varie ipotesi poi costruire un elemento message che richiami questo elemento.

```
<complexType name="Composizione">
        <choice>
            <element name="nmtoken_1" type="qname_1"/>
            <element name="nmtoken_2" type="qname_2"/>
        </choice>
</complexType>

<message name=" nmtoken ">
        <part name="composto" type="Composizione"/>
</message>
```

Port Types
Un elemento portType corrisponde ad un insieme di operazioni collegate ai rispettivi message definiti precedentemente. L'attributo name del portType definisce un nome univoco valido per tutti gli elementi portType del documento WSDL.
Un elemento portType segue la seguente struttura all'interno di un documento WSDL:

```
<wsdl:definitions .... >
        <wsdl:portType name="nmtoken">
            <wsdl:operation name="nmtoken" .... /> *
        </wsdl:portType>
</wsdl:definitions>
```

WSDL definisce quattro tipi di trasmissioni primitive che un portType può supportare:

- **One-way**: l'endpoint riceve un messaggio.

- **Request-response**: l'endpoint riceve un messaggio e invia un messaggio di risposta correlato.
- **Solicit-response**: l'endpoint invia un messaggio e riceve un messaggio di risposta correlato.

- **Notification**: l'endpoint invia un messaggio.

WSDL prevede che le operazioni possano riferire a questi tipi primitivi per indicare il tipo di trasmissione di cui necessitano; sebbene le trasmissioni di tipo request-response e solicit-response possano essere modellate in astratto tramite due messaggi di tipo one-way, è opportuno modellare queste istruzioni come operazioni primitive considerando che sono istruzioni molto comuni frequentemente utilizzate.

Un operazione **One-way** segue la seguente struttura all'interno di un documento WSDL, dove l'elemento di input specifica quale dei messaggi astratti definiti in precedenza è utilizzato come formato per l'operazione:

```
<wsdl:definitions .... >
        <wsdl:portType .... > *
                <wsdl:operation name="nmtoken">
                        <wsdl:input name="nmtoken"? message="qname"/>
                </wsdl:operation>
        </wsdl:portType >
</wsdl:definitions>
```

Un operazione **Request-response** segue la seguente struttura all'interno di un documento WSDL, dove l'elemento di input ed output specificano, rispettivamente, il formato del messaggio di richiesta e del messaggio di risposta fra i messaggi astratti definiti in precedenza; l'elemento fault è opzionale e specifica quale dei messaggi astratti è da utilizzare nel caso in cui l'output dell'operazione generi un errore.
E' importante notare che l'operazione definisce soltanto in modo astratto il modello di trasmissione adoperato; sarà dunque compito del particolare binding determinare come il messaggio sia effettivamente inviato.

```
<wsdl:definitions .... >
        <wsdl:portType .... > *
                <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
                        <wsdl:input name="nmtoken"? message="qname"/>
                        <wsdl:output name="nmtoken"? message="qname"/>
                        <wsdl:fault name="nmtoken" message="qname"/>*
                </wsdl:operation>
        </wsdl:portType >
</wsdl:definitions>
```

Un operazione **Solicit-response** segue la seguente struttura all'interno di un documento WSDL, dove l'elemento di input ed output specificano, rispettivamente, il formato del messaggio di richiesta e del messaggio di risposta fra i messaggi astratti definiti in precedenza; l'elemento fault è opzionale e specifica quale dei messaggi astratti è da utilizzare nel caso in cui l'output dell'operazione generi un errore.
E' importante notare che l'operazione definisce soltanto in modo astratto il modello di trasmissione adoperato; sarà dunque compito del particolare binding determinare come il messaggio sia effettivamente inviato.

```
<wsdl:definitions .... >
        <wsdl:portType .... > *
                <wsdl:operation name="nmtoken" parameterOrder="nmtokens">
                        <wsdl:output name="nmtoken"? message="qname"/>
                        <wsdl:input name="nmtoken"? message="qname"/>
                        <wsdl:fault name="nmtoken" message="qname"/>*
                </wsdl:operation>
        </wsdl:portType >
</wsdl:definitions>
```

Un operazione **Notification** segue la seguente struttura all'interno di un documento WSDL, dove l'elemento di output specifica quale dei messaggi astratti definiti in precedenza è utilizzato come formato per l'operazione:

```
<wsdl:definitions .... >
        <wsdl:portType .... > *
                <wsdl:operation name="nmtoken">
                        <wsdl:output name="nmtoken"? message="qname"/>
                </wsdl:operation>
        </wsdl:portType>
</wsdl:definitions>
```

L'attributo name degli elementi input ed output definisce un nome univoco per tutti gli elementi di tipo input ed output presenti all'interno dell'elemento portTyoe in esame.
WSDL fornisce dei valori di default basati sull'attributo name di operation in modo da evitare la definizione del name per ogni elemento di tipo input ed output.
Se non viene specificato il nome di un operazione One-way o Notification, di default assume il nome dell'operazione; se invece non viene specificato il nome di un operazione di Request-response o di Solicit-response, di default il nome dell'elemento di input ed output diventa il nome dell'operazione con i rispettivi suffissi "Request/Solicit" e "Response".
E' opportuno ricordare che l'elemento operation non specifica se l'operazione verrà o non verrà utilizzata in un binding di tipo RPC (remote procedure call); quindi quando si utilizza un operazione di questo tipo è necessario rispettare la struttura originale della procedura chiamata.
Per questa ragione, nelle operazioni di tipo Request-response o Olicit-response è necessario specificare la lista dei parametri all'interno dell'attributo parameterOrder (il valore di questo attributo è una lista di parti di messaggi astratti fra quelli definiti in precedenza).

Bindings
Un elemento di tipo binding definisce il formato del messaggio, i dettagli del protocollo per le operazioni ed i messaggi definiti in un certo portType. E' possibile avere più binding per un certo portType.
Un elemento binding segue la seguente struttura all'interno di un documento WSDL, l'attributo name del binding, definisce un nome univoco per tutti gli elementi di tipo binding presenti nel documento WSDL; il binding riferisce il portType attraverso l'attributo type.

```
<wsdl:definitions .... >
        <wsdl:binding name="nmtoken" type="qname"> *
                <-- extensibility element (1) --> *
                <wsdl:operation name="nmtoken"> *
                        <-- extensibility element (2) --> *
                    <wsdl:input name="nmtoken"? > ?
                        <-- extensibility element (3) -->
                    </wsdl:input>
                    <wsdl:output name="nmtoken"? > ?
                        <-- extensibility element (4) --> *
                    </wsdl:output>
                    <wsdl:fault name="nmtoken"> *
                        <-- extensibility element (5) --> *
                    </wsdl:fault>
                </wsdl:operation>
        </wsdl:binding>
</wsdl:definitions>
```

Ports

Un elemento di tipo port definisce un songolo endpoint specificando un singolo indirizzo per un certo binding.

Un elemento port segue la seguente struttura all'interno di un documento WSDL, l'attributo name del port, definisce un nome univoco per tutti gli elementi di tipo port presenti nel documento WSDL; l'attributo binding riferisce all'elemento di tipo binding a cui è associato l'indirizzo.

```
<wsdl:definitions .... >
        <wsdl:service .... > *
                <wsdl:port name="nmtoken" binding="qname"> *
                        <-- extensibility element (1) -->
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

Services

Un elemento di tipo service raggruppa fra loro alcuni port e segue la seguente struttura all'interno di un documento WSDL, l'attributo name del service, definisce un nome univoco per tutti gli elementi di tipo service presenti nel documento WSDL. Nessun port all'interno del service comunica con un altro port, se un service riferisce più port che condividono lo stesso portType ma con un indirizzo differente, allora i port sono da considerarsi alternativi; in fine ogni port segue lo stesso comportamento semantico (con le limitazioni imposte dal proprio binding sul trasporto e sul formato del messaggio).

```
<wsdl:definitions .... >
        <wsdl:service name="nmtoken"> *
                <wsdl:port .... />*
        </wsdl:service>
</wsdl:definitions>
```

5.2 SOAP Simple Object Access Protocol

SOAP (Simple Object Acces Protocol) è un protocollo basato sul linguaggio XML, leggero e semplice, per lo scambio di informazioni strutturate sul Web.

L'obiettivo di progettazione complessivo del SOAP consiste nel garantire il massimo della semplicità e fornire un minimo di funzionalità; a questo scopo il protocollo SOAP definisce una struttura di messaggistica che non contiene alcuna semantica dell'applicazione o del trasporto ed è di conseguenza modulare e molto estensibile.
Viaggiando su protocolli di trasporto standard, SOAP è in grado di sfruttare al meglio l'architettura aperta del Web e può essere facilmente accettato da parte di qualsiasi sistema arbitrario in grado di supportare gli Internet standard più basilari.

L'infrastruttura richiesta per supportare un servizio Web XML compatibile con il protocollo SOAP potrebbe essere considerata piuttosto semplicistica eppure è estremamente potente dal momento che aggiunge relativamente poco all'infrastruttura esistente di Internet e facilita comunque l'accesso universale ai servizi generati con SOAP.

La specifica del protocollo SOAP è composta da quattro parti principali.

La prima parte definisce una Envelope estensibile obbligatoria per l'incapsulamento dei dati, l'Envelope SOAP definisce un messaggio SOAP e rappresenta l'unità di scambio di base tra i processori di messaggi; questa è l'unica parte obbligatoria della specifica.

La seconda parte della specifica del protocollo SOAP definisce le regole di codifica dei dati facoltativi per la rappresentazione di dati definiti dall'applicazione e un modello uniforme per la serializzazione di modelli di dati non sintattici.

La terza parte definisce un modello di scambio dei messaggi in stile RPC (richiesta/risposta).
Ogni messaggio SOAP è una trasmissione unidirezionale e sebbene la base del protocollo SOAP vada ricercata in RPC, non si tratta solo di un meccanismo di richiesta/risposta.
I servizi Web XML spesso combinano i messaggi SOAP per implementare tali modelli, ma il protocollo SOAP non affida un modello di scambio dei messaggi e anche questa parte della specifica è facoltativa.

La quarta parte della specifica definisce un collegamento tra SOAP e HTTP. Tuttavia, anche questa parte è facoltativa. È possibile utilizzare SOAP in combinazione con qualsiasi protocollo o meccanismo di trasporto in grado di trasportare la Envelope SOAP, inclusi SMTP, FTP o persino un disco floppy.

E' importante ricordare, inoltre, che SOAP pur nascendo dalla collaborazione di importanti aziende come la Microsoft DevelopMentor ed IBM è oggetto di standardizzazione ad opera del consorzio W3C; questo fa di SOAP un protocollo standard che permette quindi agli sviluppatori di non rimanere vittime di conflitti aziendali e commerciali. Infatti anche se attualmente alcune aziende propongono differenti soluzioni basate su SOAP, queste devono necessariamente sottostare allo standar W3C così da poter essere il più possibile interoperative.

5.3 HTTP GET/POST

HTTP-GET e HTTP-POST sono protocolli standard che utilizzano verb HTTP (Hypertext Transfer Protocol) per la codifica e il passaggio di parametri come coppie nome/valore, insieme alla semantica della richiesta associata.

Ciascuno di essi è composto da una serie di intestazioni di richieste HTTP che definiscono ciò che il client richiede dal server, che a sua volta risponde (se l'elaborazione avviene correttamente) con una serie di risposte HTTP e con i dati richiesti.
HTTP-GET passa i propri parametri nella forma di testo con codifica URL mediante l'applicazione del tipo MIME/x-www-form-codifica URL, associato all'URL del server che gestisce la richiesta.

La codifica URL è una forma di codifica dei caratteri che garantisce che i parametri passati siano composti da testo conforme, quale ad esempio la codifica di uno spazio come %20. I parametri aggiunti vengono definiti anche come una stringa di query.
Analogamente a HTTP-GET, anche i parametri HTTP-POST hanno la codifica URL; tuttavia, anziché essere passati come parte dell'URL, le coppie nome/valore vengono passate all'interno dell'effettiva richiesta HTTP.

5.4 UDDI Universal Description Discovery and Integration

Come qualsiasi altra risorsa su Internet, sarebbe virtualmente impossibile trovare un determinato servizio Web XML senza disporre di qualche mezzo attraverso il quale eseguire la ricerca.

Gli elenchi di servizi Web XML offrono una posizione centrale in cui i provider di servizi Web possono pubblicare informazioni sui servizi disponibili. Tali elenchi possono essere essi stessi servizi Web, che sono accessibili a livello di codice e forniscono dei risultati delle ricerche in risposta a query eseguite da potenziali client.
Potrebbe essere necessario utilizzare un elenco di servizi Web per individuare un'organizzazione che fornisce un servizio Web XML per un particolare scopo o per determinare cosa offre a una determinata organizzazione.
Le specifiche UDDI (Universal Description, Discovery and Integration) definiscono un metodo standard per pubblicare e individuare informazioni sui servizi Web, gli schemi XML associati alle specifiche UDDI definiscono quattro tipi di informazioni che consentono a uno sviluppatore di utilizzare un servizio Web XML pubblicato, vale a dire: informazioni aziendali, sul servizio, sul collegamento e sulle specifiche dei servizi.

Come componente principale del progetto UDDI, il Registro aziendale UDDI consente alle aziende di individuare a livello di codice le informazioni sui servizi Web XML esposti da altre organizzazioni. Gli sviluppatori possono utilizzare il Registro aziendale UDDI per individuare documenti e descrizioni del servizio.

5.4.1 Utilizzo dell'UDDI e situazione attuale

Come ogni nova tecnologia anche UDDI è ancora e continuamente in fase di sviluppo, per ora molte delle potenzialità di questo strumento restano ancora tali e molti sviluppatori di applicazioni Web sono costretti ad utilizzare UDDI in modo molto simile a un semplice motore di ricerca nel quale poter cercare i servizi Web XML pubblicati per poi contattare personalmente i fornitori e definire con loro un accordo di utilizzo; affidandosi così alle ditte produttrici del servizio anche per quanto riguarda l'affidabilità e la disponibilità dello stesso.

Se utilizzato in questo modo l'UDDI si comporta come una directory, come lo schedario di una biblioteca che fornisce un metodo per individuare tutte le tecnologie di servizi Web XML. Le specifiche UDDI comprendono:

- **Le pagine bianche**: forniscono informazioni sulle aziende fornitrici dei servizi.
- **Le pagine gialle**: organizzano i servizi Web XML in categorie.
- **Le pagine verdi**: forniscono informazioni tecniche dettagliate sui singoli servizi.

UDDI è stato sviluppato però per fornire più che il semplice supporto della fase di progettazione.
L'analogia con i comuni motori di ricerca non tiene conto del fatto che UDDI supporta anche la fase di esecuzione. UDDI gioca un ruolo critico al termine del processo di individuazione, grazie alla possibilità di programmare query all'UDDI stesso in fase di esecuzione; UDDI può inoltre fungere da infrastruttura per realizzare applicazioni di servizi Web affidabili e solide.

Se consideriamo alcune delle più comuni difficoltà o problemi che sorgono dopo l'integrazione di un servizio Web in un'applicazione client, una questione fondamentale è l'incapacità di prevedere e individuare interruzioni di un servizio Web dovute al fornitore (host), nonché di poterne ripristinare il corretto funzionamento.
È in situazioni di questo tipo che UDDI assume un ruolo determinante quale infrastruttura in grado di supportare i servizi Web durante l'esecuzione.

UDDI risponde ai problemi relativi alla "qualità del servizio" definendo una convenzione di chiamata che comporta la memorizzazione nella cache di informazioni di binding come il punto di accesso del servizio Web nonché altri parametri, specifici per implementazione; quando si verificano errori, un client può inviare una query UDDI in fase di esecuzione, aggiornando le informazioni nella cache con quelle più recenti.
Lo schema di tale convenzione è simile al seguente:

- Individuare un servizio Web in UDDI.
  Utilizzare il file WSDL per il servizio Web (rappresentato da un TModel UDDI) e i dettagli di implementazione, quali il punto di accesso e qualsiasi altra informazione di configurazione, contenuti in un bindingTemplate UDDI.

- Preparare un'applicazione client per un particolare servizio Web.
  Nell'applicazione client, memorizzare nella cache la bindingKey univoca del servizio Web in modo che, quando l'applicazione viene utilizzata per la prima volta, possa richiamare tutte le informazioni necessarie da UDDI.

- Quando l'applicazione invoca il servizio Web remoto, utilizzare i dati memorizzati che sono stati ottenuti da un elenco Web UDDI.

- Se l'invocazione non riesce, utilizzare il valore della bindingKey e la chiamata API get_bindingTemplate a un elenco UDDI per ottenere una copia aggiornata delle informazioni di binding.

- Confrontare le nuove informazioni con quelle precedenti: se sono diverse, riprovare ad eseguire la chiamata non riuscita. Se il tentativo ha buon esito, sostituire i dati nella cache con quelli nuovi, memorizzandoli per le chiamate successive. Comunque, se le informazioni di binding ottenute sono identiche a quelle già memorizzate, significa che il fornitore non ha eseguito alcun aggiornamento e l'applicazione dovrebbe generare un errore. Analogamente, se le informazioni di binding sono nuove e la chiamata non riesce, l'applicazione dovrebbe generare un errore.

Per quanto riguarda il fornitore di servizi Web, questi dovrebbe essere consapevole del fatto che è possibile aggiornare la voce UDDI per il servizio Web, se opportuno. Per reindirizzare il traffico a una nuova posizione o sistema di backup il fornitore di un servizio Web deve soltanto attivare la nuova destinazione, quindi modificare il punto di accesso nell'elenco UDDI.

Questo approccio è definito retry on failure (riprova in caso di errore) e offre ai client un meccanismo per il ripristino in caso di errore durante l'esecuzione.

Purtroppo però questo aspetto dell'UDDI non è ancora del tutto supportato dagli UDDI più comuni a causa di molti fattori: la tecnologia UDDI è ancora in sviluppo, pochi XML Web Service sono disponibili, i dati forniti dai fornitori non sono ancora del tutto standardizzati e la cultura del servizio web non è ancora entrata nel senso comune degli sviluppatori; inoltre i vari UDDI non hanno ancora definito una modalità comune per la presentazione e la gestione dei dati rendendo molto problematico (e per nulla standardizzato) l'accesso ai dati da parte di progettisti Web.

5.5 Creazione e Utilizzo di Web Service tramite .NET

5.5.1 Parametri Input/Output del Web Service

Prima di poter entrare nel dettaglio dello sviluppo di un XML Web Service è importante sapere che per garantire prestazioni veloci e leggerezza di codice, nonché una standardizzazione del servizio stesso non tutti i tipi di dati sono supportati da un XML Web Service come parametri di input o output.

In generale possono essere usati i seguenti tipi di dato:

Tipi di Dato	Descrizione
Primitive Types	Tipi primitive Standard. String, Char, Byte, Boolean, Int16, Int32, Int64, UInt16, UInt32, UInt64, Single, Double, Guid, Decimal, DateTime (as XML's timeInstant), DateTime (as XML's date), DateTime (as XML's time), and XmlQualifiedName (as XML's QName).
Enum Types	Enumeration types come ad esempio "public enum color { red=1, blue=2 }".
Arrays of Primitives, Enums	Array di tipi primitivi, come string[ ] and int[ ].
Classes and Structs	Classi e struct con attribute o proprietà pubbliche.
Arrays of Classes (Structs)	Arrays delle Classi o Struct.
DataSet	ADO.NET DataSet Types.
Arrays of DataSet	Arrays di tipi DataSet.
XmlNode	XmlNode.
Arrays of XmlNode	Arrays di tipi XmlNode.

Quando si richiama un XML Web Service usando SOAP o HTTP GET/POST tutti i tipi esposti sopra sono supportati come valori di ritorno.
Quando si usa SOAP come protocollo tutti i parametri sono supportati sia come input che come output; ma se si usa HTTP GET/POST solo un limitato numero di tipi sono supportati:

Tipi di Dato	Descrizzione
Primitive Types (limitati)	Int32, String, Int16, Int64, Boolean, Single, Double, Decimal, DateTime, UInt16, UInt32, UInt64, and Currency. Dalla prospettiva del Client tutti questi dati sono convertiti in stringhe.
Enum Types	Dalla prospettiva del Client tutti questi dati sono convertiti in classi con costanti statiche definite come stringhe per ogni valore.
Arrays of Primitives, Enums	Arrays di Tipi Primitivi come string[ ] and int[ ].

5.5.2 Introduzione a un XML Web Service in C# per .NET

@WebServe Directive
La prima parte di un file .asmx è **@WebServe Directive**, simile a @Page Directive per una pagina Web Form, è usata per identificare gli attributi specifici del file XML Web Service.
La @WebService Directive è il solo codice che è strettamente necessario per un file .asmx; possiamo creare un XML Web Service sia definendo una classe nel file .asmx che sarà compilata dal compilatore JIT (Just In Time) o anche definendo una classe nell'assembly; in questo caso nessun altra riga di codice è richiesta dopo la @ WebService directive, tutto il codice per l'XML Web Service sarà messo nel file della classe.

Sono previsti alcuni attributi per la WebService Directive:

- **Language**: Questo attributo specifica il linguaggio .NET usato per scrivere l'XML Web Service.

- **Class**: Specifica il nome della classe che può essere definita nello stesso file .asmx o in un assembly che risederà nella directory \bin dell'applicazione Web.

Esempio:

```
<%@WebService Language="C#" Class="MioService" %>
```

WebService Attribute
Il WebService Attribute (Classe **System.Web.Services.WebServiceAttribute**) è un attributo opzionale che può essere aggiunto alla classe pubblica specificata nell'XML Web Service per applicare specifici valori descrittivi del servizio.

- **Name**: Specifica il nome a cui il Web Service si riferisce.

- **Description**: Serve per scrivere commenti che aiutino a rendere il codice più user-friendly.

- **Namespace**: Specifica il namespace di default per l'XML Web Service.

Esempio:

```
[WebService(Name="MioService", Description="Descrizione del Servizio.",
Namespace="http://www.dotnetjunkies.com/")]
public class MyWebServiceClass
{
  //...
}
```

WebMethod Attribute
Il WebMethod Attribute non è opzionale, qualsiasi metodo che voglia essere pubblicato come metodo per un XML Web Service deve avere un WebMethodAttribute e deve essere dichiarato pubblico.
Il WebMethodAttribute indica al Framwork .NET che il metodo deve essere reso accessibile dai protocolli standard per internet, implementando le seguenti caratteristiche:

- **BufferResponse**: Specifica se il risultato di questo metodo deve essere bufferizzato, di default è impostato su true.

- **CacheDuration**: Specifica il numero di secondi durante i quali il risultato del metodo rimane in Cache.

- **Description**: Serve per rendere più user-friendly il codice.

- **EnableSession**: Specifica se lo stato di Sessione è attivato per il metodo, di default è impostato si false.

- **MessageName**: Specifica il nome usato per il Web Method.

- **TransactionOption**: Specifica l'Enterprise Service transaction support per il metodo.

Esempio:

```
[WebMethod(Description="Descrizione del Metodo.")]
public String MioWebMethod()
{
 return "FooBar";
}
```

Uso di un XML Web Service da parte di un'Applicazione Client
Per utilizzare l'XML Web Service creato si deve usare il tool **WSDL.exe** (messo a disposizione dal Framework .NET) per creare un Proxy Class; dopo di che si compila il proprio codice applicativo con inclusa la Proxy Class.
Una volta che la Poxy Class esiste si possono creare oggetti basati su di essa, ogni metodo chiamato andrà sul Web all'URI dell'XML Web Service (in genere tramite richieste SOAP).
Esempio:

Il comando riportato di seguito consente di creare un file WSDL per il servizio Web XML disponibile all'URL specificato nonché una classe proxy client in linguaggio C# per il servizio Web XML.

```
wsdl http://hostServer/WebserviceRoot/WebServiceName.asmx?WSDL
```

Il comando riportato di seguito consente di creare una classe proxy client in linguaggio C# per un servizio Web XML disponibile all'URL specificato. La classe proxy client viene salvata nel file myProxyClass.cs.

```
wsdl /out:myProxyClass.cs http://hostServer/WebserviceRoot/WebServiceName.asmx
```

5.5.3 Creazione di un Semplice Web Service

5.5.3.1 XML Web Service

Come prima cosa è necessario scrivere il codice dell'XML Web Service che verrà salvato su di un file denominato ciao.asmx.

```
<%@ WebService Language="c#" Class="MioService" %>
using System;
using System.Web.Services;

[WebService()]
public class MioService : WebService
{
[WebMethod()]
 public String DiCiao( String TuoNome )
        {
        return "Ciao, " + TuoNome;
        }
}
```

Assieme alla Proxy Class viene generato anche il codice in linguaggio WSDL che descrive l'XML Web Service e che funge da piattaforma comune per tutte le applicazioni che vorranno dialogare col nostro XML Web Service.
E' importante ricordare che l'XML Web Service userà XML per la gestione dei dati, XSD Schema per il controllo sulla sintassi e della forma dei dati e il Framework prepara automaticamente l'XML Web Service per essere interrogato tramite SOAP e HTTP.

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions
        xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:s="http://www.w3.org/2001/XMLSchema"
        xmlns:s0="http://tempuri.org/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
        xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
        targetNamespace=http://tempuri.org/
        xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="DiCiao">
```

```
<s:complexType>
<s:sequence>
   <s:element minOccurs="0" maxOccurs="1" name="TuoNome" type="s:string"/>
   </s:sequence>
   </s:complexType>
   </s:element>
<s:element name="DiCiaoResponse">
 <s:complexType>
 <s:sequence>
   <s:element minOccurs="0" maxOccurs="1" name="DiCiaoResult" type="s:string"/>
   </s:sequence>
   </s:complexType>
   </s:element>
   <s:element name="string" nillable="true" type="s:string"/>
   </s:schema>
   </types>

 <message name="DiCiaoSoapIn">
   <part name="parameters" element="s0:DiCiao"/>
</message>

 <message name="DiCiaoSoapOut">
   <part name="parameters" element="s0:DiCiaoResponse"/>
</message>

 <message name="DiCiaoHttpGetIn">
   <part name="TuoNome" type="s:string"/>
</message>

<message name="DiCiaoHttpGetOut">
   <part name="Body" element="s0:string"/>
</message>

<message name="DiCiaoHttpPostIn">
   <part name="TuoNome" type="s:string"/>
</message>

<message name="DiCiaoHttpPostOut">
   <part name="Body" element="s0:string"/>
</message>

<portType name="MioServiceSoap">
<operation name="DiCiao">
   <input message="s0:DiCiaoSoapIn"/>
   <output message="s0:DiCiaoSoapOut"/>
</operation>
</portType>
```

```
<portType name="MioServiceHttpGet">
<operation name="DiCiao">
  <input message="s0:DiCiaoHttpGetIn"/>
  <output message="s0:DiCiaoHttpGetOut"/>
</operation>
</portType>

<portType name="MioServiceHttpPost">
<operation name="DiCiao">
  <input message="s0:DiCiaoHttpPostIn"/>
  <output message="s0:DiCiaoHttpPostOut"/>
</operation>
</portType>

<binding name="MioServiceSoap" type="s0:MioServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="DiCiao">
  <soap:operation soapAction="http://tempuri.org/DiCiao" style="document"/>
<input>
  <soap:body use="literal"/>
</input>
<output>
  <soap:body use="literal"/>
</output>
</operation>
</binding>

 <binding name="MioServiceHttpGet" type="s0:MioServiceHttpGet">
  <http:binding verb="GET"/>
<operation name="DiCiao">
  <http:operation location="/DiCiao"/>
<input>
  <http:urlEncoded />
  </input>
<output>
  <mime:mimeXml part="Body"/>
</output>
</operation>
</binding>

<binding name="MioServiceHttpPost" type="s0:MioServiceHttpPost">
  <http:binding verb="POST"/>
<operation name="DiCiao">
  <http:operation location="/DiCiao"/>
<input>
  <mime:content type="application/x-www-form-urlencoded"/>
```

```
</input>
<output>
  <mime:mimeXml part="Body"/>
</output>
</operation>
</binding>

<service name="MioService">
<port name="MioServiceSoap" binding="s0:MioServiceSoap">
  <soap:address location="http://localhost/temp/ciao.asmx"/>
</port>
<port name="MioServiceHttpGet" binding="s0:MioServiceHttpGet">
  <http:address location="http://localhost/temp/ciao.asmx"/>
</port>
<port name="MioServiceHttpPost" binding="s0:MioServiceHttpPost">
  <http:address location="http://localhost/temp/ciao.asmx"/>
</port>
</service>

</definitions>
```

5.5.3.2 Creazione della Proxy Class

Una volta creato il file contenente l'XML Web Service ci si posiziona nella directory che lo contiene e si lancia il tool per la creazione della Proxy Class messo a disposizione dal Framework .NET:

```
wsdl /out:miaProxyClass.cs http://localhost/temp/ciao.asmx
```

Questo comando lancia il tool WSDL.exe che crea una Poxy Class chiamata miaProxyClass.cs per il servizio che si trova all'URL specificato.

Proxy Class autogenerata dal Tool WSDL

```
//------------------------------------------------------------------------------
// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version: 1.0.3705.288
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </autogenerated>
//------------------------------------------------------------------------------
```

```
// Codice sorgente generato automaticamente da wsdl, versione=1.0.3705.288.

using System.Diagnostics;
using System.Xml.Serialization;
using System;
using System.Web.Services.Protocols;
using System.ComponentModel;
using System.Web.Services;

/// <remarks/>
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Web.Services.WebServiceBindingAttribute(Name="MioServiceSoap",
Namespace="http://tempuri.org/")]
public class MioService : System.Web.Services.Protocols.SoapHttpClientProtocol {

    /// <remarks/>
    public MioService() {
        this.Url = "http://localhost/temp/ciao.asmx";
    }

    /// <remarks/>

[System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://tempuri.org/DiCiao"
, RequestNamespace="http://tempuri.org/", ResponseNamespace="http://tempuri.org/",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
    public string DiCiao(string TuoNome) {
        object[] results = this.Invoke("DiCiao", new object[] {
                    TuoNome});
        return ((string)(results[0]));
    }

    /// <remarks/>
    public System.IAsyncResult BeginDiCiao(string TuoNome, System.AsyncCallback
callback, object asyncState) {
        return this.BeginInvoke("DiCiao", new object[] {
                    TuoNome}, callback, asyncState);
    }

    /// <remarks/>
    public string EndDiCiao(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return ((string)(results[0]));
    }
}
```

5.5.3.3 Applicazione Client che usa l'XML Web Service

Come ultima cosa è stata realizzata una semplice pagina ASP .NET che utilizza l'XML Web Service; si è scelto il protocollo HTTP GET per interrogare e connettersi al servizio per la sua semplicità di utilizzo.

```
<html>
<head>
<link rel="alternate" type="text/xml" href="ciao.asmx?disco"/>
<style type="text/css">
BODY { color: #000000; background-color: white; font-family: Verdana; margin-left: 0px;
margin-top: 0px; }
        ……
        Istruzioni di formattazione
        ……
        .frmInput { font-family: Verdana; font-size: 1em; }
        .intro { margin-left: -15px; }
</style>
<title>MioService Servizio Web</title>
</head>
<body>
<div id="content">
<p class="heading1">MioService</p><br>
<span>
<h2>DiCiao</h2>
<p class="intro"></p>
<h3>Test</h3>
Per eseguire il test dell'operazione utilizzando il protocollo HTTP GET, fare clic sul pulsante
'Richiama'.
<form target="_blank" action='http://localhost/temp/ciao.asmx/DiCiao' method="GET">
<table cellspacing="0" cellpadding="4" frame="box" style="border-collapse: collapse;">
<tr>
<td class="frmHeader" background="#dcdcdc" style="border-right: 2px solidwhite;">
Parametro</td>
<td class="frmHeader" background="#dcdcdc">Valore</td>
</tr>
<tr>
          <td class="frmText" style="color: #000000; font-weight:normal;">TuoNome:</td>
          <td><input class="frmInput" type="text" size="50" name="TuoNome"></td>
</tr>
<tr>
           <td></td>
           <td align="right"> <input type="submit" value="Richiama" class="button"></td>
</tr>
</table>
</form>
</body>
</html>
```

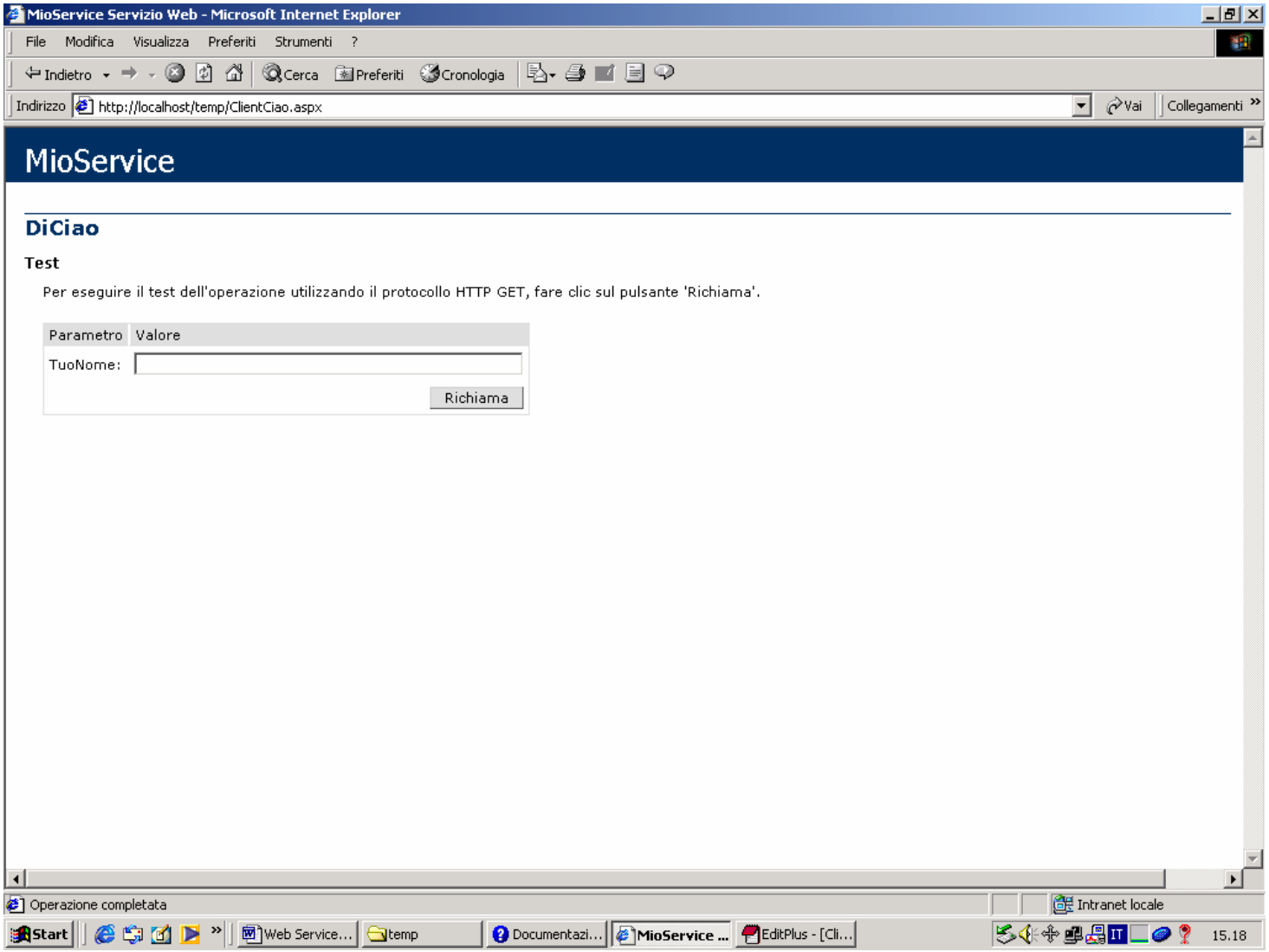
5.5.3.4 Output del Web Service

Il Risultato del XML Web Service è un file XML disponibile per l'applicazione Client che ha invocato il servizio.

Esempio:

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://tempuri.org/">Ciao, Yuri</string>
```

Mentre per l'utente la pagina ASP.NET visualizza il file XML restituito dall'XML Web Service fornendo questa videata:



Capitolo 5, Figura 9: Output della Pagina ASP .NET che Visualizza l'XML Web Service

5.5.4 Uso di un Semplice Web Service esterno tramite UDDI

5.5.4.1 Recupero del XML Web Service

La Prima operazione da farsi è individuare un XML Web Service in un UDDI (Universal Description Discovery and Integration) che corrisponda alle proprie esigenze.

In questo semplice esempio si è scelto come "motore di ricerca" il sito UDDI XMethods (www.xmetohods.com), nella lista degli XML Web Service pubblicati in questo sito è stato scelto un servizio di criptazione e decriptazione di una stringa di caratteri usando l'algoritmo TripleDesCbc.

Il servizio, realizzato in MS .NET, prende in input un testo piano e una chiave che verrà usata per il criptaggio e per il decriptaggio.

5.5.4.2 Utilizzo delle Specifiche date dall'UDDI

Una volta scelto l'UDDI e l'XML Web Service si devono utilizzare le risorse rese disponibili dall'UDDI stesso: il file WSDL per il servizio Web e i dettagli di implementazione, quali il punto di accesso e qualsiasi altra informazione di configurazione.
Nel nostro caso l'UDDI XMethods metteva a disposizione dei suoi utenti le seguenti informazioni:

- **WSDL**: http://test.mapfrepr.net/Encryption/Encryption.asmx?WSDL
  Si rende noto l'indirizzo da cui scaricare il file WSDL che servirà per costruire la propira Proxy Class.

- **Key**: A4BCB1BB-091A-E39E-966B-69C327B7FA01
  Chiave messa a disposizione dal produttore per accedere gratuitamente all'XML Web Service.

- **Owner:** arios
  Produttore dell'XML Web Service.

- **For More Info:** http://www.mapfrepr.com
  Sito della ditta produttrice del servizio, da dove attingere informazioni aggiuntive sul servizio stesso.

5.5.4.3 Usare il file WSDL

Dalle specifiche precedentemente ottenute è possibile scaricato il seguente file WSDL che descrive il servizio:

```
<?xml version="1.0" encoding="utf-8" ?>

<definitions
        xmlns:http=http://schemas.xmlsoap.org/wsdl/http/
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:s="http://www.w3.org/2001/XMLSchema"
        xmlns:s0="http://test.mapfrepr.net/"
        xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
        xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
        targetNamespace="http://test.mapfrepr.net/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>

<s:schema elementFormDefault="qualified" targetNamespace="http://test.mapfrepr.net/">

<s:element name="EncryptText">
<s:complexType>
<s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="text" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="privateKey" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>

<s:element name="EncryptTextResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="EncryptTextResult" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>

<s:element name="DecryptText">
<s:complexType>
<s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="encText" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1" name="privateKey" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
```

```
<s:element name="DecryptTextResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="DecryptTextResult" type="s:string" />
</s:sequence>
</s:complexType>
</s:element>
<s:element name="string" nillable="true" type="s:string" />
</s:schema></types>

<message name="EncryptTextSoapIn">
        <part name="parameters" element="s0:EncryptText" />
</message>

<message name="EncryptTextSoapOut">
        <part name="parameters" element="s0:EncryptTextResponse" />
</message>

<message name="DecryptTextSoapIn">
        <part name="parameters" element="s0:DecryptText" />
</message>

<message name="DecryptTextSoapOut">
        <part name="parameters" element="s0:DecryptTextResponse" />
</message>

<message name="EncryptTextHttpGetIn">
        <part name="text" type="s:string" />
        <part name="privateKey" type="s:string" />
</message>

<message name="EncryptTextHttpGetOut">
        <part name="Body" element="s0:string" />
</message>

<message name="DecryptTextHttpGetIn">
        <part name="encText" type="s:string" />
        <part name="privateKey" type="s:string" />
</message>

<message name="DecryptTextHttpGetOut">
        <part name="Body" element="s0:string" />
</message>

<message name="EncryptTextHttpPostIn">
        <part name="text" type="s:string" />
        <part name="privateKey" type="s:string" />
</message>
```

```
<message name="EncryptTextHttpPostOut">
        <part name="Body" element="s0:string" />
</message>

<message name="DecryptTextHttpPostIn">
        <part name="encText" type="s:string" />
        <part name="privateKey" type="s:string" />
</message>

<message name="DecryptTextHttpPostOut">
        <part name="Body" element="s0:string" />
</message>

<portType name="EncryptionWSSoap">
        <operation name="EncryptText">
                <input message="s0:EncryptTextSoapIn" />
                <output message="s0:EncryptTextSoapOut" />
        </operation>
        <operation name="DecryptText">
                <input message="s0:DecryptTextSoapIn" />
                <output message="s0:DecryptTextSoapOut" />
        </operation>
</portType>

<portType name="EncryptionWSHttpGet">
        <operation name="EncryptText">
                <input message="s0:EncryptTextHttpGetIn" />
                <output message="s0:EncryptTextHttpGetOut" />
        </operation>
        <operation name="DecryptText">
                <input message="s0:DecryptTextHttpGetIn" />
                <output message="s0:DecryptTextHttpGetOut" />
        </operation>
</portType>

<portType name="EncryptionWSHttpPost">
        <operation name="EncryptText">
                <input message="s0:EncryptTextHttpPostIn" />
                <output message="s0:EncryptTextHttpPostOut" />
        </operation>
        <operation name="DecryptText">
                <input message="s0:DecryptTextHttpPostIn" />
                <output message="s0:DecryptTextHttpPostOut" />
        </operation>
</portType>
```

```
<binding name="EncryptionWSSoap" type="s0:EncryptionWSSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />

<operation name="EncryptText">
<soap:operation soapAction="http://test.mapfrepr.net/EncryptText" style="document" />
<input>
        <soap:body use="literal" />
</input>
<output>
        <soap:body use="literal" />
</output>
</operation>

<operation name="DecryptText">
<soap:operation soapAction="http://test.mapfrepr.net/DecryptText" style="document" />
<input>
        <soap:body use="literal" />
</input>
<output>
        <soap:body use="literal" />
</output>
</operation>
</binding>

<binding name="EncryptionWSHttpGet" type="s0:EncryptionWSHttpGet">
<http:binding verb="GET" />
<operation name="EncryptText">
<http:operation location="/EncryptText" />
<input>
        <http:urlEncoded />
</input>
<output>
        <mime:mimeXml part="Body" />
</output>
</operation>

<operation name="DecryptText">
<http:operation location="/DecryptText" />
<input>
        <http:urlEncoded />
</input>
<output>
        <mime:mimeXml part="Body" />
</output>
</operation>
</binding>
```

```
<binding name="EncryptionWSHttpPost" type="s0:EncryptionWSHttpPost">
<http:binding verb="POST" />
<operation name="EncryptText">
<http:operation location="/EncryptText" />
<input>
        <mime:content type="application/x-www-form-urlencoded" />
</input>
<output>
        <mime:mimeXml part="Body" />
</output>
</operation>
<operation name="DecryptText">
<http:operation location="/DecryptText" />
<input>
        <mime:content type="application/x-www-form-urlencoded" />
</input>
<output>
        <mime:mimeXml part="Body" />
</output>
</operation>
</binding>

<service name="EncryptionWS">
<port name="EncryptionWSSoap" binding="s0:EncryptionWSSoap">
<soap:address location="http://test.mapfrepr.net/Encryption/Encryption.asmx" />
</port>
<port name="EncryptionWSHttpGet" binding="s0:EncryptionWSHttpGet">
<http:address location="http://test.mapfrepr.net/Encryption/Encryption.asmx" />
</port>
<port name="EncryptionWSHttpPost" binding="s0:EncryptionWSHttpPost">
<http:address location="http://test.mapfrepr.net/Encryption/Encryption.asmx" />
</port>
</service>

</definitions>
```

Tramite il tool **WSDL.exe** lanciato dalla direcotry che conterrà i file del mio codice, ottengo una Proxy Class da usare per la mia applicazione.

wsdl /out:CriptProxyClass.cs http://test.mapfrepr.net/Encryption/Encryption.asmx?WSDL

Questo commando genera il seguente file chiamato CriptProxyClass.cs

```
//------------------------------------------------------------------------------
// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version: 1.0.3705.288
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </autogenerated>
//------------------------------------------------------------------------------
//
// Codice sorgente generato automaticamente da wsdl, versione=1.0.3705.288.
//
using System.Diagnostics;
using System.Xml.Serialization;
using System;
using System.Web.Services.Protocols;
using System.ComponentModel;
using System.Web.Services;

/// <remarks/>
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Web.Services.WebServiceBindingAttribute(Name="EncryptionWSSoap",
Namespace="http://test.mapfrepr.net/")]
public class EncryptionWS : System.Web.Services.Protocols.SoapHttpClientProtocol {

    /// <remarks/>
    public EncryptionWS() {
        this.Url = "http://test.mapfrepr.net/Encryption/Encryption.asmx";
    }

    /// <remarks/>

[System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://test.mapfrepr.net/En
cryptText",                                    RequestNamespace="http://test.mapfrepr.net/",
ResponseNamespace="http://test.mapfrepr.net/",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
    public string EncryptText(string text, string privateKey) {
        object[] results = this.Invoke("EncryptText", new object[] {
                    text,
```

```
                privateKey});
        return ((string)(results[0]));
    }

    /// <remarks/>
    public System.IAsyncResult BeginEncryptText(string text, string privateKey,
System.AsyncCallback callback, object asyncState) {
        return this.BeginInvoke("EncryptText", new object[] {
                text,
                privateKey}, callback, asyncState);
    }

    /// <remarks/>
    public string EndEncryptText(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return ((string)(results[0]));
    }

    /// <remarks/>

[System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://test.mapfrepr.net/De
cryptText", RequestNamespace="http://test.mapfrepr.net/",
ResponseNamespace="http://test.mapfrepr.net/",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
    public string DecryptText(string encText, string privateKey) {
        object[] results = this.Invoke("DecryptText", new object[] {
                encText,
                privateKey});
        return ((string)(results[0]));
    }

    /// <remarks/>
    public System.IAsyncResult BeginDecryptText(string encText, string privateKey,
System.AsyncCallback callback, object asyncState) {
        return this.BeginInvoke("DecryptText", new object[] {
                encText,
                privateKey}, callback, asyncState);
    }

    /// <remarks/>
    public string EndDecryptText(System.IAsyncResult asyncResult) {
        object[] results = this.EndInvoke(asyncResult);
        return ((string)(results[0]));
    }
}
```

5.5.4.4 Scrivere il codice della Classe che utilizzerà il web service

A questo punto si dovrà scrivere la Classe (C#) che utilizza il Web Service, per farlo sarà necessario importare i seguenti namespace che contengono gli oggetti forniti dal Framework .NET per lavorare direttamente con il codice sorgente dell'UDDI:

```
using Microsoft.Uddi;
using Microsoft.Uddi.Binding;
```

Questi namespace sono presenti in: Microsoft.Uddi.SDK.dll scaricabile dal sito della Microsoft.

```
using System;
using System.Web.UI.WebControls;
using System.Web.UI;
using System.Data;
using System.IO;
using Microsoft.Uddi;
using Microsoft.Uddi.Binding;
using WebCript;

namespace WebCript
{
public class Cript : Page
{
//Access Point dato dall'UDDI dal quale si accede al servizio
  protected string accessPoint =
"http://test.mapfrepr.net/Encryption/Encryption.asmx?WSDL";
  protected BindingTemplate bt;
  protected string risultatocript;
  protected string risultatodecript;
  protected TextBox textBox1 = new TextBox();
  protected Button button1 = new Button ();
  protected Label label1 = new Label ();
  protected TextBox textBox2 = new TextBox();
  protected Label label2 = new Label ();
  protected bool flag = false;
  protected TextBox textBox3 = new TextBox();
  protected TextBox textBox4 = new TextBox();

//Funzione per invocare il servizio Web:

    protected bool InvokeWebService()
    {
      // Creo un oggetto del tipo del mio service
```

```
        EncryptionWS sr = new EncryptionWS();
      //imposta il punto di accesso per la classe proxy
        sr.Url = accessPoint;
      try
      {
        //Invocazione del Primo Metodo del Web Service
           risultatocript = sr.EncryptText( textBox3.Text,textBox4.Text);
        label1.Text = "Risultato del Criptagio:   " +risultatocript;
        textBox1.Text += "XML Web Service Invocata con Successo!!!! WOW!!";
        flag = true;
      }
      catch
      {
        textBox1.Text += "Errorone! nell'Invocare il Primo Servizio dell'XML Web Service";
        flag = false;
      }

        try
      {
        //Invocazione del Secondo Metodo del Web Service
           risultatodecript = sr.DecryptText( risultatocript,textBox4.Text);
        label2.Text = "Risultato del Decriptagio:   " +risultatodecript;
        textBox2.Text += "XML Web Service Invocata con Successo!!!! WOW!!";
        flag = true;
      }
      catch
      {
        textBox2.Text += "Errorone! nell'Invocare il Secondo Servizio dell'XML Web
Service";
        flag = false;
      }
        return flag;
    }
//Infine, quando l'utente fa clic sul pulsante, l'applicazione tenta di invocare il servizio Web.

protected void button1_Click(object sender, System.EventArgs e)
{
//prova ad invocare il servizio Web
bool WebServiceSuccess = InvokeWebService();
//in caso negativo, visualizzo un errore
if ( WebServiceSuccess == false )
        {
        textBox1.Text += "Chiamata all'XML Web Service Fallita.";
        }
}
}//Fine Classe Cript
}// Fine Namespace
```

5.5.4.5 Compilazione dei Componenti

Una volta che è stata scritta la Classe Client ed è stata generata la Proxy Class i due file (.cs) devono essere compilati insieme tramite un file bat (nel nostro esempio webcript.bat).

```
csc      /t:library /debug+ /debug:full /r:"c:/Programmi/Microsoft UDDI
SDK/Microsoft.Uddi.Sdk.dll"
/out:bin\webcript.dll Cript.cs CriptProxyClass.cs
```

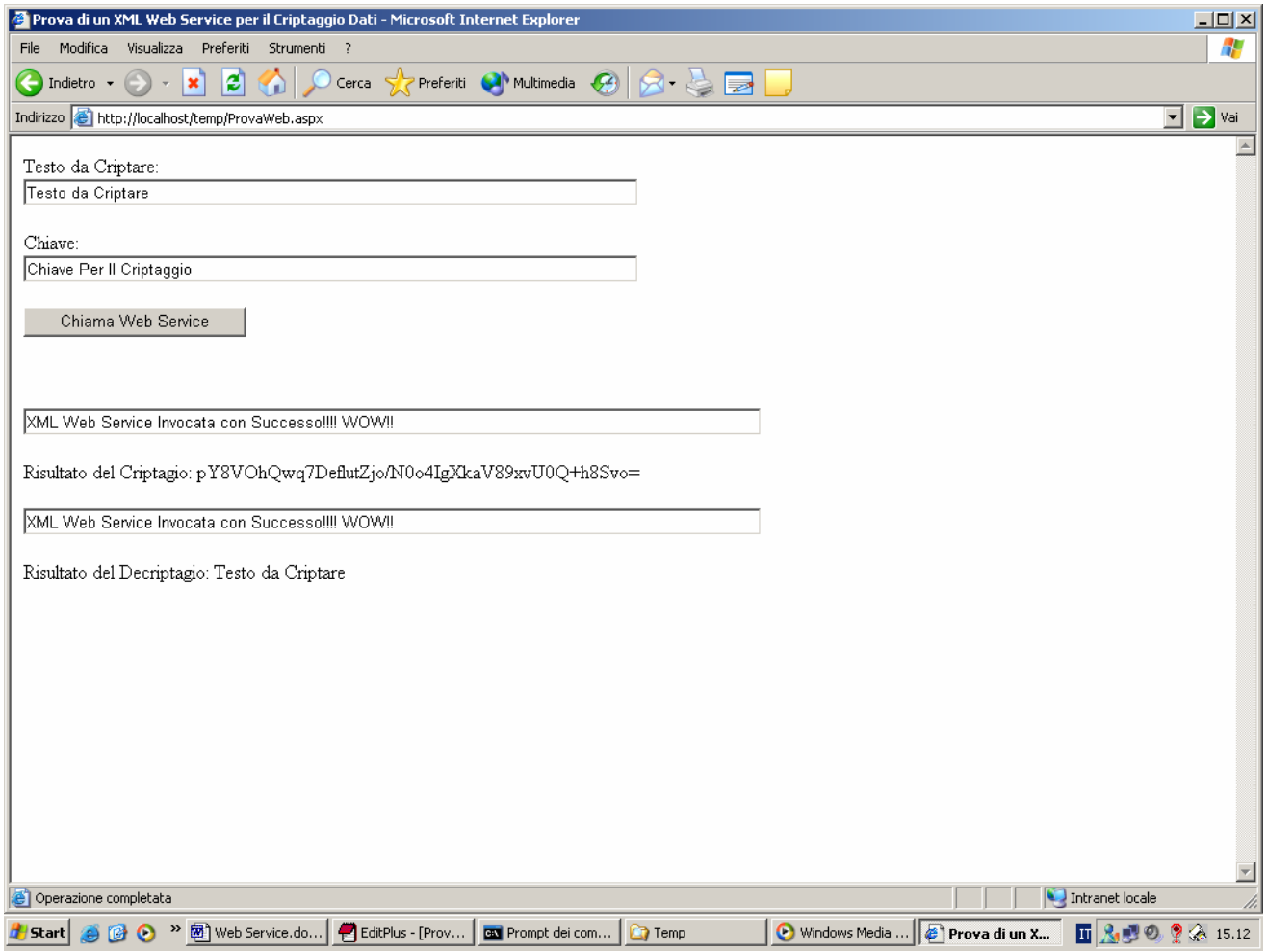
5.5.4.6 Interfaccia Utente

Essendo un XML Web Service fondamentalmente un servizio per altre applicazioni perché un utente umano possa vedere gli effetti del servizio dovrà essere scritta una semplice interfaccia utente, per farlo useremo una pagina ASP .NET (nel nostro esempio ProvaWeb.aspx).

La Pagina deve essere scritta in una directory virtuale impostata tramite l'IIS (Internet Information Service) in modo da poter essere raggiunta dal Browser attraverso il localhost (http://localhost/temp/ProvaWeb.aspx).

```
<%@Page language="C#" inherits="WebCript.Cript" Debug="true"%>
<HTML>
<HEAD>
<TITLE> Prova di un XML Web Service per il Criptaggio Dati</TITLE>
</HEAD>

<BODY>
<form runat="server">
Testo da Criptare:  <br>
<asp:TextBox ID="textBox3" runat="server" width=500/><br><br>
Chiave: <br>
<asp:TextBox ID="textBox4" runat="server" width=500/><br><br>
<asp:Button ID="button1" runat="server" text="Chiama Web Service"
            onclick="button1_Click"/><br><br><br><br>
<asp:TextBox ID="textBox1" runat="server" width=600/><br><br>
<asp:Label ID="label1" runat="server"/><br><br>
<asp:TextBox ID="textBox2" runat="server" width=600/><br><br>
<asp:Label ID="label2" runat="server"/>
</form>
</BODY>

</HTML>
```

5.5.4.7 Output dell'XML Web Service



Capitolo 5, Figura 10: Output della Pagina ASP .NET che Esegue un XML Web Service su UDDI

5.6 Conclusioni su XML Web Service

Questa tecnologia ha i suoi punti di forza nella standardizzazione e nell'elevata interoperabilità da essa derivata; l'XML Web Service rende le applicazioni Web in grado di mettere i propri dati a disposizione dell'intera rete in modo che vi si possa accedere con i più comuni protocolli di comunicazione.

Gli accordi fra fornitore e cliente del servizio basati su punti di accesso ben definiti e sull'uso di chiavi per controllare l'accesso garantiscono un elevata sicurezza, e i protocolli che stanno alla base dei servizi (XML) sono perfettamente supportati dalla emergente tecnologia del Framework .NET che permette l'utilizzo e la creazione di XML Web Service anche a chi non conosce nello specifico la loro struttura interna (WSDL).

Considerato l'attuale livello di sviluppo (che è comunque in continua evoluzione e in continuo miglioramento) l'XML Web service ha ancora problemi di affidabilità dovuti alla sua dislocazione nel Web e alla non ancora completamente standardizzata e sviluppata tecnologia degli UDDI che forniscono i servizi.

Infatti se si decide di non utilizzare a pieno le potenzialità di un UDDI, mettendosi personalmente d'accordo con la ditta fornitrice del servizio e considerato che un XML Web Service è in genere utilizzato da altre applicazioni (e non da un utente umano), si rischia di prendersi carico dei problemi del fornitore.

Si rischia cioè, di subire passivamente qualsiasi difficoltà subita dal server del fornitore che ospita il servizio da noi usato: se ad esempio la nostra applicazione Web sta attendendo una risposta da un XML Web Service e nel frattempo il server che la ospita cade o rallenta per qualche problema interno, la nostra applicazione rimarrà in attesa tanto tempo quanto ne servirà perché un timeout non blocchi la richiesta; inoltre ogni ritardo del fornitore nell'avvertire il cliente su eventuali spostamenti del servizio o sue modifiche provocherebbe continue interruzioni dei servizio stesso che, proprio perché automatizzato in modo statico all'interno di altre applicazioni, ne provocherebbe ritardi e blocchi.

Gli sviluppatori dovranno quindi attendere, prima di poter utilizzare a pieno questo servizio, che lo sviluppo degli UDDI raggiunga presto un livello sufficientemente alto da poter supportare le applicazioni anche nella fase di esecuzione, dando così informazioni dinamiche in tempo reale alle applicazioni Web che possono così gestire qualsiasi problema o rallentamento del servizio.

6. Conclusioni

Il progetto che intendevo realizzare è stato portato a termine e può dirsi concluso: è stato approfondito l'uso di SQL Server 2000 (soprattutto per quanto riguarda XML), è stata esaminata affondo un alternativa a questo RDBMS data da MySQL 4.0.12, permettendo così anche ad applicazioni Web complesse di poter disporre di piattaforme differenti per l'archiviazione e la gestione dei dati. A questo scopo sono stati rilevati tutti gli accorgimenti e le metodologie per passare da una piattaforma SQL Server 2000 a una MySQL tramite nuova la tecnologia data dal Framework .NET.

E' stato studiato a fondo lo strumento dei documenti XML come metodo per trasferire dati garantendo l'interoperabilità fra applicazioni e RDBMS, è stata esaminata e valutata l'ipotesi di utilizzare l'XML come "database" a se stante e come base per la creazione e l'utilizzo di XML Web Service.

L'esperienza di stage mi ha quindi permesso di conoscere un RDBMS completamente differente dai più comuni standard Microsoft, mi ha dato la possibilità di apprendere l'uso di un nuovo linguaggio di programmazione (C#) per ambienti Web e Windows nonché la possibilità (con ASP .NET) di studiare un evoluzione del linguaggio ASP appreso durante gli studi.

E' stato notevolmente interessante anche lo studio di una tecnologia così potente, versatile e pur relativamente semplice quale l'XML (e relativi strumenti di controllo e interrogazione quali DTD, XSL, XSD Schema e XPath) che sta guadagnando sempre più credito nella progettazione e nello sviluppo di applicazioni Web sempre più in grado di comunicare liberamente tra loro.

Seppur ancora all'inizio del loro sviluppo, anche gli XML Web Service sono risultati essere un interessante strumento per creare applicazioni Web decentralizzate e accessibili che in futuro garantiranno un notevole passo avanti delle tecnologie legate allo scambio di dati sul Web.

Concludendo, il progetto svolto è stato molto interessante e particolarmente stimolante in quanto mi ha offerto la possibilità di lavorare con strumenti e argomenti del tutto nuovi, di affrontare problematiche differenti da quelle affrontate nel corso di sudi, di poter valutare personalmente tecnologie e strumenti che più si adattavano al mio modo di lavorare e alle esigenze del progetto stesso.

Info:
Contemporaneamente alla fine della stesura di questa tesi la MySQL AB ha rilasciato gratuitamente la versione di prova di MySQL Server 4.1.0 Alpha che implementa anche le SELECT innestate e le Stored Procedure, nonché altri miglioramenti e accorgimenti che rendono più sicure e soprattutto più user firendly questo Open Source Database che (pur non avendo informazioni a sufficienza su questa ultima versione) sempre più si sta rivelando una valida alternativa ai più pesanti e costosi RDBMS commerciali.

7. Glossario

Termine o Acronimo	Spiegazione
. NET Framework	Il .NET Framework è una piattaforma per la realizzazione della nuova generazione di applicazioni e servizi Web XML distribuiti. Espone un modello di programmazione coerente, indipendente dal linguaggio e condiviso da tutti gli strati di un applicazione e consente di interoperare in modo trasparente e di migrare facilmente a partire dalle tecnologie esistenti. Il .NET Framework è costituito da tre componenti fondamentali: il Common Language Runtime, le Classi unificate ed ASP .NET.
ADO .NET	La tecnologia del .NET Framework per l'accesso ai dati.
API Web	API che consentono l'integrazione di un servizio Web XML all'intenro del .NET Framework.
ASP .NET	La tecnologia Active Server Pages per il .NET Framework.
Assembly	L'unità per il deployment ed il controllo dei conflitti di versione del .NET Framework. Definisce gli spazi dei nomi per soddisfare le richieste e determina quali risorse esporre esternamente e quali rendere, invece, accessibili esclusivamente dall'interno dell'assembly. Un assembly include un manifesto che ne descrive i contenuti.
BDB	BerkeleyDB, tabella fornita da MySQL 4.0.12 transaction safe che si basa su tecnologia XML.
C#	Il primo linguaggio orientato alle componenti della famiglia C/C++. E' stato sottoposto all'ECMA per la standardizzazione.
CGI	Common Gateway Interface, il primo protocollo Internet utilizzato per generare contenuti interattivi sul Web.
CLR	Common Language Runtime. I sistemi dei tipi dei metadati e di esecuzione forniti dal .NET Framework, che forniscono codice gestito e dati con servizi quali l'integrazione cross-language, la sicurezza per l'accesso al codice, la gestione del tempo di vita degli oggetti ed il supporto per il debugging ed il profilino.
Codice nativo	Codice compilato in codice macchina specifico per il processore.
DOM	Document Object Model, è una rappresentazione in memoria di un documento XML; grazie ad essa è possibile leggere e modificare un documento XML a livello di programmazione.
DTD	Document Type Definition, un file che indica le regole per la definizione e la correlazione logica tra elementi, gli attributi e gli altri dati di un documento XML.
ECMA	Ente europeo per gli standard nato nel 1961.

Termine o Acronimo	Spiegazione
Form Web	Le form Web sono una tecnologia ASP .NET utilizzabile per realizzare pagine Web programmabili. Le Form Web possono visualizzare informazioni all'utente, utilizzando un qualsiasi linguaggio di marcatura, in qualsiasi browser ed utilizzare il codice server per implementare la logica applicativa.
GPL	General Public Licenses, licenza che garantisce l'acquisizione e la libera circolazione di Software libero utilizzabile e modificabile di chiunque senza scopo di lucro.
HEAP	Tabella fornita da MySQL 4.0.12 non transaction safe e residente completamente in memoria, molto veloce ma estremamente poco affidabile.
HTML	Hyper Text Markup Language, nasce come lingua franca per pubblicare informazioni ipertestuali sul World Wide Web. L'HTML è un formato, non proprietario, basato su SGML (Standard Generalized Markup Language).
HTTP	Hyper Text Transfer Protocol, protocollo Internet standard per trasferire informazioni tra Client e Server, Server e Server e così via.
IDL	Interface Definition Language, un linguaggio utilizzato dall'applicazione per specificare le interfacce che intende esporre ad altre applicazioni.
IL	Intermediate Language, è un linguaggio utilizzato come input per numerosi compilatori e come input per un comolatore JIT. L'IL definisce un'architettura di esecuzione astratta e basata sullo stack. Il CLR può includere diversi compilatori JIT per convertire l'IL in codice nativo.
InnoDB	Tabella Fornita da MySQL 4.0.12 transaction safe ottimizzata per grandi quantità di dati.
ISAM	Tabella fornita da MySQL 4.0.12 non transaction safe.
JIT	Just In Time, una fase che descrive un'azione che viene eseguita solo quando necessario, come la compilazione Just in Time o l'attivazione dell'oggetto Just IN Time. Per convenzione il termine JIT viene utilizzato per riferirsi al compilatore JIT.
LGPL	Lesser General Public Licenses, versione più restrittiva della licenza GPL, questa licenza che tratta sempre di software libero ha più clausole specifiche per la gestione di librerie e funzioni software.
Manifesto	Metadati che descrivono i moduli ed i file di risorse che fanno parte di un particolare assembly.
MERGE	Tabella fornita da MySQL 4.0.12 che implementa un set di tabelle di tipo MyISAM.
Metadati	Informazioni sui dati. I metadati vengono memorizzati assieme a questi all'interno dei file eseguibili ed utilizzati da compilatori, strumenti e dal runtime per fornire un ricco insieme di servizi.
MyISAM	Tabella di default di MySQL 4.0.12 non transaction safe, ottenuta migliorando il tipo ISAM.

Termine o Acronimo	Spiegazione
Proxy Class	Classe C# autogenerata dal tool wsdl.exe di .NET Framework che consente l'accesso ai dati forniti da un XML Web Service.
SOAP	Simple Object Access Protocol, standard W3C. Un protocollo leggero per lo scambio di informazioni in un ambiente decentralizzato e distribuito. E' un protocollo basato su XML costituito da tre parti: uno strato esterno che definisce una struttura per la descrizione e l'elaborazione del contenuto di un messaggio, un insieme di regole di codifica per esprimere istanze di tipo dato ed una convenzione per la rappresentazione di invocazioni a procedure remote relativamente esposte.
UDDI	Le specifiche Universal Description Discovery and Integration, un'iniziativa che crea una struttura aperta, globale ed indipendente dalla piattaforma che consente ai servizi aziendali di individuarsi gli uni rispetto gli altri, definire le modalità di interazione su Internet e condividere informazioni in un registry globale.
WebMethod	Parola chiave del .NET Framework che consente di accedere agli oggetti da Internet.
WSDL	Web Service Description Language, una grammatica XML utilizzabile da sviluppatori e strumenti di sviluppo per rappresentare le funzionalità di un XML Web Service.
XML	Extendible Markup Language, uno standard W3C per la formattazione di documenti e dati strutturati sul Web.
XML Web Service	Un servizio Web XML è un applicazione che espone le proprie funzionalità da codice su Internet utilizzando protocolli standard come XML, HTTP, SOAP e MIME.
XPath	XML Path Language, è il risultato di uno sforzo per fornire di una sintassi comune tra XSLT e XPointer; lo scopo primario di XPath è quello di indirizzare parti di un documento XML e fornire funzioni di base per la manipolazione di numeri, stringhe e boolean.
XSD Schema	Standard W3C che codifica un unico documento di riferimento che funge da prototipo. In questo modo possono essere definite una volta per tutte sia la struttura che la grammatica degli elementi che compongono tutti i documenti XML scambiati.
XSL	Extensible Styleseet Language, serie di tag che possono essere usati per applicare delle regole di formattazione a ciascuno degli elementi interni ad un documento XML.

8. Bibliografia

Link Internet

.NET Home Page:
www.microsoft.com/net

Visual C# .NET:
http://msdn.microsoft.com/vcsharp/

Microsoft UDDI Home Page:
http://uddi.microsoft.com/

Tutto su .NET
(Esempi, Help, Articoli, ecc..):
www.gotdotnet.com

World Wide Web Consortium:
www.w3c.org

MySQL Home Page:
http://www.mysql.com/

UDDI XMethods Home Page:
http://www.xmethods.com/

UDDI Organization Home Page:
http://www.uddi.org/

DataBase Group Dipartimento di Ingegneria dell'Informazione Facoltà di Ingegneria di Modena:
http://dbgroup.unimo.it/

Libri

[1] Titolo: Progetto di Basi di dati Relazionali “lezioni ed esercizi”
Autore/i: Domenico Beneventano, Sonia Bergamaschi, Maurizio Vincini
Editore: Pitagora Editrice

[2] Titolo: Microsoft Visual C# .NET “Passo per Passo”
Autore/i: Jhon Sharp, Jon Jagger
Editore: Mondadori Informatica

[3] Titolo: XML Pocket Reference “Extensible Markup Language”
Autore/i: Robert Eckstein
Editore: Hops Libri

9. Indice delle Figure

Appendice "A"

A.1 Confronto dettagliato fra SQL Server 2000 e MySQL 4.0.12

Sono di seguito riportate una serie di tabelle e grafici che confrontano in modo sintetico e completo le due piattaforme SQL Server 2000 e MySQL 4.0.12 sviluppate secondo ideologie completamente opposte e concorrenti sul mercato.

DEFINIZIONE DATI

Tipo di Dato	SQL92	MySQL 4.0.12	SQL Server 2000
BIGINT	NO	SI	SI
BIT	SI	SI	SI
BIT VAR	SI	NO	NO
BLOB	NO	SI	NO
BOOL	NO	SI	SI
CHAR	SI	SI	SI
DATE	SI	SI	NO
DATETIME	NO	SI	SI
DECIMAL	SI	SI	SI
DOUBLE PRECISION	SI	SI	SI
ENUM	SI	SI	NO
FLOAT	SI	SI	SI
IMAGE	NO	NO	SI
INT o INTEGER	SI	SI	SI
INTERVAL	SI	NO	NO
MEDIUMINT	NO	SI	NO
MONEY	NO	NO	SI
NCHAR	SI	SI	SI
NUMERIC	SI	SI	SI
NVCHAR	SI	SI	SI
REAL	SI	SI	SI
SET	SI	SI	NO
SMALLDATETIME	NO	NO	SI
SMALLINT	SI	SI	SI
SMALLMONEY	NO	NO	SI
SQL_VARIANT	NO	NO	SI
SYSNAME	NO	NO	SI
TEXT	NO	SI	SI
TIME	SI	SI	NO
TIMESTAMP	SI	SI	SI

TINYINT	NO	SI	Si
UNIQUEIDENTIFIRE	NO	NO	SI
VARCHAR	SI	SI	SI
YEAR	NO	SI	NO

OPERATORI RELAZIONALI

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
=	SI	SI	SI
<=>	NO	SI	NO
<>	SI	SI	SI
>	SI	SI	SI
>=	SI	SI	SI
<	SI	SI	SI
<=	SI	SI	SI

FUNZIONI AGGREGATE

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
AVG	SI	SI	SI
COUNT	SI	SI	SI
MAX	SI	SI	SI
MIN	SI	SI	SI
SUM	SI	SI	SI

VINCOLI DI TABELLA

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
AUTOINCREMENT	NO	SI	SI
CHECK	SI	NO	SI
FOREIGN KEY	SI	SI **	SI
NOT NULL	SI	SI	SI
ON DELETE CASCADE	SI	SI	SI
ON DELETE NO ACTION	SI	SI	SI
ON DELETE SET DEFAULT	SI	NO	NO
ON DELETE SET NULL	SI	SI	NO
ON UPDATE CASCADE	SI	SI	NO
ON UPDATE NO ACTION	SI	SI	SI
ON UPDATE SET DEFAULT	SI	NO	NO
ON UPDATE SET NULL	SI	SI	NO
PRIMARY KEY	SI	SI *	SI
UNIQUE	SI	SI	SI
ZEROFILL	NO	SI	SI

* MySQL prevede un unica chiave primaria costituita da una unica colonna, nel caso si vogliano implementare chiavi primarie su più attributi si devono creare esplicitamente tramite un indice, che contenga tutte le chiavi, definito PRIMARY

** MySQL supporta da questa versione anche le FOREIGN KEY, per utilizzarle la procedura è molto più complessa di quella di SQL Server. Per prima cosa si possono usare le chiavi

esterne solo su i tipi di tabella BDB e InnoDB, dopo di ché ogni chiave deve essere impostata esplicitamente dall'utente ed è molto difficile modificare le tabelle così ottenute dopo la creazione.

MySQL vuole che le chiavi primarie e le loro chiavi importate siano identiche e non controlla chiavi con valori NULL.

Procedura di utilizzo:
Creare le tabelle Padre in cui si specifica la chiave primaria (che sarà ereditata).
Creare le tabelle Figli con i campi ereditati identici a quelli del padre, definire le chiavi primarie, crear un indice per ogni chiave eterna del padre e poi aggiungere le FOREIGN KEY.

CREATE e DROP

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
CREATE ASSERTION	SI	NO	SI
CREATE DOMAIN	SI	NO	NO
CREATE INDEX	SI	SI	SI
CREATE SCHEMA	SI	NO	SI
CREATE TABLE FROM SELECT	NO	SI	NO
CREATE TABLE IF NOT EXIST	NO	SI	NO
CREATE VIEW	SI	NO	NO
DROP DOMAIN	SI	NO	NO
DROP MANY TABLES	NO	SI	NO
DROP SCHEMA	SI	NO	SI
DROP TABLE	SI	SI	SI
DROP TABLE IF EXIST	NO	SI	NO
DROP VIEW	SI	NO	SI
TMPORARY TABLE	SI	SI (HEAP, MyISAM,ISAM)	SI
TRUNCATE TABLE	NO	SI	NO

ALTER TABLE

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
ALTER COLUMN DROP FOREIGN KEY	NO	SI (con drop foreign key)	SI (con drop constraint)
ALTER DOMAIN	SI	NO	NO
ALTER TABLE ADD COLUMN	SI	SI	SI
ALTER TABLE ADD CONSTRAINT	SI	SI	SI
ALTER TABLE ADD FOREIGN KEY	SI	SI	SI
ALTER TABLE ADD MANY COLUMN	NO	SI	SI
ALTER TABLE ALTER COLUMN DEFAULT	SI	SI	NO
ALTER TABLE CHANGE COLUMN	NO	SI	NO
ALTER TABLE DROP COLUMN	SI	SI	SI
ALTER TABLE DROP CONSTRAINT	NO	NO	SI
ALTER TABLE MODIFY COLUMN	NO	SI	NO
ALTER TABLE RENAME TABLE	NO	SI	NO
ALTER TRIGGER	NO	NO	SI
ALTER VIEW	SI	NO	SI

SELECT

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
ALL	SI	SI	SI
COLUMN ALIAS	SI	SI	SI
COMPUTE	NO	NO	SI
DISTINCT	SI	SI	SI
INSERT INTO.. SELECT	NO	SI	SI
SELECT Tab_name.*	SI	SI	SI
SELECT WITHOUT FROM	NO	SI	SI
SUBQUERY	SI	NO *	SI
TABLE ALIAS	SI	SI	SI

* MySQL non prevede l'uso di interrogazioni innestate in questa versione, per ottenere gli stessi risultati è necessario fare ampio uso di JOIN e di tabelle temporanee.

FUNCTION in WHERE

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
ALL	SI	NO	SI
ANY	SI	NO	SI
BETWEEN	SI	SI	SI
EXIST	SI	NO	SI
IN (NUMBERS)	SI	SI	SI
IS NOT NULL	SI	SI	SI
IS NULL	SI	SI	SI
LIKE	SI	SI	SI
LIKE ESCAPE	SI	SI	SI
NOT BETWEEN	SI	SI	SI
NOT EXIST	SI	NO	SI
NOT LIKE	SI	SI	SI
SOME	SI	NO	SI

GROUP BY

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
GROUP BY	SI	SI	SI
GROUP BY ALIAS	SI	SI	NO
GROUP BY POSITION	NO	SI	NO
GROUP ON COLUMN WITH NULL VALUES	SI	SI	SI
GROUP ON UNUSED COLUMN	SI	SI	SI

HAVING

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
HAVING	SI	SI	SI
HAVING ON ALIAS	NO	SI	SI
HAVING WITH GROUP FUNCTION	SI	SI	SI

OREDER BY

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
ORDER BY	SI	SI	SI
ORDER BY ALIAS	SI	SI	SI
ORDER BY FUNCTION	SI	SI	SI
ORDER BY POSITION	SI	SI	SI
ORDER BY UNUSED COLUMN	SI	SI	SI

SET

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
EXCEPT	SI	NO	NO
EXCEPT ALL	SI	NO	NO
INTERSECT	SI	NO	NO
INTERSECT ALL	SI	NO	NO
MINUS	NO	NO	SI
UNION	SI	SI	SI
UNION (INOCMPATIBLE LIST)	SI	SI	SI
UNION ALL	SI	SI	SI

INSERT, DELETE, UPDATE

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
DELETE	SI	SI	SI
INSERT WITH SET SYNTAX	SI	SI	SI
INSERT WITH VALUES LIST	SI	SI	SI
UPDATE	SI	SI	SI

TIPI DI JOIN

Istruzione	SQL92	MySQL 4.0.12	SQL Server 2000
CROSS JOIN (from a,b)	SI	SI	SI
DELETE FROM tables1,tebles2,..	SI	SI	NO**
FULL OUTER JOIN	SI	NO	SI
INNER JOIN	SI	SI	SI
LEFT OUTER JOIN	SI	SI	SI
NATURAL JOIN	SI	SI	NO
NATURAL LEFT OUTER JOIN	NO	SI	NO
RECURSIVE SUB QUERIES*	SI	NO	40
RIGHT OUTER JOIN	SI	SI	SI
SIMPLE JOINS	SI	SI	SI
SUBQUERIES	SI	NO	SI
TABLES IN JOIN	--	31	+64
UPDATE from many tables	SI	SI	NO**
UPDATE with SUBSELECT	SI	NO	SI

* Livello di sotto linee
** Prevede un Delete da un Join di Tabelle
*** Prevede un Update da un Join di Tabelle

STRING HANDLING

Operazione	MySQL 4.0.12	SQL Server 2000
Allow ‘ and “ as string markers	SI	NO
Case insensitive compare	SI	SI
Constant string size in SELECT	1048565	16777207
Constant string size in where	1048565	8000
Double “ as ‘ in string	SI	SI
Hex strings (x’1ace’)	SI	NO
Ignore end space in compare	SI	SI
Insert empty string	SI	SI
Multiple line string	SI	NO
Remembers space in varchar()	NO	SI
Select Constans	SI	SI

QUOTING

Identificatore	SQL92	MySQL 4.0.12	SQL Server 2000
“ as identifier quote (ANSI SQL)	SI	NO	SI
[ ] as identifier quote	SI	NO	SI
‘ as identifier quote	SI	SI	NO

COMMENTI

Tipo	SQL92	MySQL 4.0.12	SQL Server 2000
# AS COMMENT	NO	SI	NO
-- AS COMMENT	NO	SI	SI
/* */ AS COMMENT	NO	SI	SI

FUNZIONI SQL E FUNZIONI PROPRIETARIE

Funzione	SQL92	MySQL 4.0.12	SQL Server 2000
& (bitwise and)	SI	SI	SI
| (Bitwise or)	SI	SI	SI
+ , - , * , /	SI	SI	SI
<< and >> (Bitwise Shift)	NO	SI	NO
ABS	NO	SI	SI
ACOS	NO	SI	SI
ADDDATE o DATE_ADD	NO	SI	NO
AES_ENCRYPT and DECRYPT	NO	SI	NO
AND as ‘&&’	SI	SI	NO
ASCII	NO	SI	SI
ASIN	NO	SI	SI
ATAN	NO	SI	SI
ATN2	NO	NO	SI
Automatic num->string convert	NO	SI	NO
Automatic string->num convert	NO	SI	SI
BIT_AND	NO	SI	NO
BIT_LENGTH	NO	SI	NO
BIT_OR	NO	SI	NO
CAST o CONVERT	SI	SI	SI
CEILING	NO	SI	SI
CHARINDEX	NO	NO	SI
COALESCE	SI	SI	SI
COL_LENGHT	NO	NO	SI
COL_NAME	NO	NO	SI
COLUMNPROPERTY	NO	NO	SI
CONCAT(list)	SI	SI	NO
CONTAINS	NO	NO	SI
CONTAINSTABLE	NO	NO	SI
COS	NO	SI	SI
COT	NO	SI	SI

CURRENT_DATE	NO	SI	NO
CURRENT_TIME	NO	SI	NO
CURRENT_TIMESTAMP	NO	SI	SI
CURRENT_USER	NO	SI	SI
CURSOR_STATUS	NO	NO	SI
DATABASE	NO	SI	NO
DATABASEPROPERTY	NO	NO	SI
DATE_FORMAT	NO	SI	NO
DATEADD	NO	NO	SI
DATEDIFF	NO	NO	SI
DATENAME	NO	NO	SI
DATEPART	NO	NO	SI
DAY	NO	NO	SI
DB_NAME	NO	NO	SI
DEALLOCATE	NO	NO	SI
DECLARE	NO	NO	SI
DECLARE CURSOR	NO	NO	SI
DECODE and ENCODE	NO	SI	SI
DEGREES	NO	SI	SI
DENY	NO	NO	SI
DIFFERENCE	NO	NO	SI
ELT	NO	SI	NO
ENCRYPT	NO	SI	SI
EXP	NO	SI	SI
EXPORT_SET	NO	SI	NO
EXTRACT	NO	SI	NO
FIELD	NO	SI	NO
FILE_ID	NO	NO	SI
FILE_NAME	NO	NO	SI
FILEGROUP_ID	NO	NO	SI
FILEGROUP_NAME	NO	NO	SI
FILEGROUPPROPERTY	NO	NO	SI
FILEPROPERTY	NO	NO	SI
FIND_IN_SET	NO	SI	NO
FLOOR	NO	SI	SI
FORMAT	NO	SI	NO
FROM_DAYS	NO	SI	NO
FROM_UNIXTIME	NO	SI	NO
Function concatenation with +	NO	NO	SI
GET_LOCK	NO	SI	NO
GETDATE	NO	NO	SI
GO and GOTO	SI	NO	SI
GREATTEST	NO	SI	NO
HEX	NO	SI	NO
HOUR	NO	SI	NO
IDENTITY	NO	NO	SI

IF	NO	SI	SI
IN on numbers in SELECT	SI	SI	SI
IN on string SELECT	SI	SI	SI
INET_ATON and INET_NTOA	NO	SI	NO
INTERVAL	NO	SI	NO
LAST_INSERT_ID	NO	SI	NO
LEAST	NO	SI	NO
LEN	NO	NO	SI
LENGTH	NO	SI	NO
LIKE ESCAPE in SELECT	NO	SI	NO
LIKE in SELECT	NO	SI	SI
LN	NO	SI	NO
LOAD_FILE	NO	SI	NO
LOCALTIME	NO	SI	NO
LOCALTIMESTAMP	NO	SI	NO
LOCATE as INSTR	NO	SI	NO
LOG, LOG2, LOG10	NO	SI	SI
LOWER	NO	SI	SI
LPAD	SI	SI	NO
LTRIM	SI	SI	SI
MID	NO	SI	NO
MINUTE	NO	SI	NO
MOD as %	SI	SI	SI
MONTH	NO	SI	SI
MONTHNAME	NO	SI	NO
NOT as “!” in SELECT	NO	SI	NO
NULLIF WITH NUMBERS	NO	SI	SI
NULLIF WITH STRING	NO	SI	SI
OBJECT_ID	NO	NO	SI
OBJECT_NAME	NO	NO	SI
OBJECTPROPERTY	NO	NO	SI
OCTET_LENGTH	NO	SI	NO
OR as ‘||’	NO	SI	NO
PASSWORD	NO	SI	NO
PATINDEX	NO	NO	SI
PERIOD_ADD	NO	SI	NO
PERIOD_DIFF	NO	SI	NO
PI	NO	SI	SI
POSITION	SI	SI	NO
POW	NO	SI	SI
QUARTER	NO	SI	NO
RADIANS	NO	SI	SI
RAND	NO	SI	SI
RELEASE_LOCK	NO	SI	NO
REPEAT	NO	SI	NO
REPLACE	NO	SI	SI

REPLICATE	NO	NO	SI
REVERSE	NO	SI	SI
RIGHT	NO	SI	SI
ROUND	NO	SI	SI
RPAD	NO	SI	NO
RTRIM	NO	SI	SI
SAPDB compatible TRIM(1 ARG)	NO	SI	NO
Searched CASE	NO	SI	SI
SEC_TO_TIME	NO	SI	NO
SECOND	NO	SI	NO
SESSION_USER	NO	SI	SI
Simple CASE	NO	SI	SI
SPACE	NO	SI	SI
SQRT	NO	SI	SI
STD	NO	SI	SI
STR	NO	NO	SI
STRCMP	NO	SI	NO
STUFF	NO	NO	SI
SUBDATE o DATE_SUB	NO	SI	NO
SUBSTRING	NO	SI	SI
SYSDATE	NO	SI	NO
SYSTEM_USER	NO	SI	SI
TAN	NO	SI	SI
TIME_TO_SEC	NO	SI	NO
TO_DAYS	NO	SI	NO
TRIM	SI	SI	NO
TRUNCATE	NO	SI	NO
UPPER	NO	SI	SI
USER	NO	SI	SI
VERSION	NO	SI	NO
WEEKDAY	NO	SI	NO
YEAR	NO	SI	NO

ESPRESSIONI

Tipo	MySQL 4.0.12	SQL Server 2000
Binary Numbers (0b1001)	NO	SI
Hex Numbers (0x41)	SI	SI
TRUE and FALSE	NO	SI

LIMITI NEI NOMI

Nome	MySQL 4.0.12	SQL Server 2000
Case independent field name	SI	NO
Column name length	64	128
Different namespace for index	SI	SI
Index name length	64	128
Rename Table	SI	NO
Select alias name length	+512	128
Table name length	+512	128

LIMITI NEGLI INDICI

Operazione	MySQL 4.0.12	SQL Server 2000
ALTER TABLE ADD PRIMARY KEY	With Constraint	With Constraint
ALTER TABLE ADD UNIQUE	SI	SI
ALTER TABLE DROP PRIMARY KEY	SI (Con drop primary key)	NO
ALTER TABLE DROP UNIQUE	SI (Con drop key)	SI (Con constraint)
CREATE INDEX	SI	SI
DROP INDEX	SI	SI
INDEX IN COLUMN PART	SI	NO
INDEX IN CREATE TABLE	SI	NO
INDEX LENGTH	500	900
INDEX PARTS	16	16
INDEX VARCHAR PART LENGTH	255	900
MAX INDEX	32	+64
MAX INDEX PART LENGTH	255	900
NULL IN INDEX	SI	SI
UNIQUE INDEXES	32	+64
INDEXING	B-Tree*	B-Tree

* Tutti gli indici MySQL (PRIMARY, UNIQUE e INDEX) sono implementati con B-Tree

LIMITI NEI TIPI

Tipo	MySQL 4.0.12	SQL Server 2000
Supports 9999-12-31 dates	SI	NO
Supports YY-MM-DD 2000 compiliant dates	SI	NO
Supports 0000-00-00 dates	SI	NO
No need to cast from integer to float	SI	SI
Mixing Integer and Float in expression	SI	SI
Max char() size	1048543	8000
Max text or Blob size	1048543	+8000000
Max varchar() size	1048543	8000
Storage Float values	Round	Round
Supports 0001-01-01 dates	SI	NO

ALTRI LIMITI

Tipo	MySQL 4.0.12	SQL Server 2000
Columns in table	2819	1024
Query Size	104857	16777216
Simultanus connectios (Installation default)	101*	1000

* Impostato di default durante l'istallazione può essere cambiato dal Superutente tramite: SET GLOBAL max_connections=valore_voluto; il limite è dato dalla RAM e dal sistema operativo.

NOTE VARIE

	MySQL 4.0.12	SQL Server 2000
ACID Compliant	NO (SI con InnoDB, BDB)	SI
Allows end ';'	SI	SI
Atomic Updates	NO (SI con InnoDB, BDB)	SI
Atomic Updates with ROLLBACK	NO (SI con InnoDB, BDB)	SI
Criptazione della Connessione Client-Server	SSL 4.0	SSL 4.2
Installazione su XP	Facile (68,2 MB)	Media (270 MB)
License	GPL, LGPL, Commercial	Commercial
Lock Table	SI	NO
Right Assignment Level	HIGH	MEDIUM
Stored Procedure	NO	SI
Transactions	SI	SI
Trigger	NO	SI
Views	NO	SI

LIVELLI DI ISOLAMENTO

	MySQL 4.0.12	SQL Server 2000
READ-COMMITTED	SI	SI
READ-UNCOMMITTED	SI	SI
REPEATABLE-READ	SI	SI
SERIALIZABLE	SI	SI

PIATTAFORME SUPPORTATE

	MySQL 4.0.12	SQL Server 2000
AIX	SI	NO
BSDI	SI	NO
COMPAQ	SI	NO
DEC	SI	NO
FreeBSD	SI	NO
HP-UX	SI	NO
IRIX	SI	NO
LINUX	SI	NO
MacOS	SI	NO
NetBSD	SI	NO
OpenBSD	SI	NO
OS/2	SI	NO
SCO	SI	NO
SOLARIS	SI	NO
Tru64	SI	NO
WINDOWS 2000	SI	SI
WINDOWS 95	SI	NO
WINDOWS 98	SI	NO
WINDOWS NT	SI	SI
WINDOWS XP	SI	SI

INTERFACCE A PROGRAMMI

	MySQL 4.0.12	SQL Server 2000
C/C++/C#	SI	SI
DELPHI	SI	SI
JDBC	SI	SI
ODBC	SI	SI
OLEDB	NO*	SI
PERL	SI	SI
PHP	SI	SI
PYTHON	SI	SI
VISUAL BASIC	SI	SI
XML	SI (Tramite Client)	SI

* Lo sviluppo è in corso ci si può riuscire con alcuni trucchi

A.2 Test di velocità fra SQL Server 2000 e MySQL 4.0.12

Una delle caratteristiche più importanti da esaminare in un DBMS è la sua velocità nell'eseguire query SQL lavorando con differenti volumi di dati.
E' stato quindi eseguito uin confronto sulle differenti velocità di SQL Server 2000 e MySQL 4.0.12, per farlo si è deciso di utilizzare una pagina ASP .NET che dopo essersi connessa al database esegue una query SQL.
In questo modo sarà possibilite valutare le prestazioni dei due DBMS all'interno di applicazioni webdatabase di ultima generazione e farsì così un idea delle loro potenzialità.

Piattaforma utilizzata per il confronto
Processore: Intel Pentium 4 CPU 1.70 GHz.
RAM: 224 Mb.
Cache: 512 Kb.
Sistema Operativo: Windows XP Professional.
Disco Fisso:
Maxtor 4D04H2 da 40 Gb con Velocità di accesso 6000 rpm e Formattazione: NTFS.

Struttura del Database di prova

Tabella1 (id, Nome)

Colonna	Tipo	Note
Id	Int (4)	Chiave Primaria
Nome	Varchar (30)	

Tabella2 (id, Cognome)

Colonna	Tipo	Note
Id	Int (4)	Chiave Primaria
Cognome	Varchar (30)	

Tabella3 (id, id_Nome, id_Cognome)
AK: id_Nome
AK: id_Cognome
FK: id_Nome **REFERENCES** Tabella1
FK: id_Cognome **REFERENCES** Tabella2

Colonna	Tipo	Note
id	Int (4)	Chiave Primaria
Id_Nome	Int (4)	Indice
id_Cognome	Int (4)	Indice

Creazione del Database con MySQL 4.0.12
La creazione del database "Prova" viene eseguita tramite un Client e vengono create le tabelle secondo la sintassi MySQL 4.0.12 per le InnoDB, scelte per una maggiore aderenza agli Standard di controllo e sicurezza di SQL Server 2000.

Viene di seguito indicata la sintassi da utilizzare secondo MySQL 4.0.12 per utilizzare tabelle transaction safe relazionate fra loro; si ricorda che per poter definire chiavi esterne si devono creare indici espliciti sui campi che saranno poi referenziati.

Creazione delle tabelle

```
CREATE TABLE `Tabella1` (`id` INT (3) UNSIGNED DEFAULT '0' NOT NULL,
                         `Nome` VARCHAR (30) DEFAULT '0',
                         PRIMARY KEY(`id`))  TYPE = InnoDB

CREATE TABLE `Tabella2` (`id` INT (3) UNSIGNED DEFAULT '0' NOT NULL,
                         `Cognome` VARCHAR (30) DEFAULT '0',
                         PRIMARY KEY(`id`))  TYPE = InnoDB

CREATE TABLE `Tabella3` (`id` INT (3) UNSIGNED DEFAULT '0' NOT NULL,
                         `id_Nome` INT (3) UNSIGNED DEFAULT '0',
                         `id_Cognome` INT (3) UNSIGNED DEFAULT '0',
                         PRIMARY KEY(`id`))  TYPE = InnoDB
```

Creazione degli indici espliciti

```
ALTER TABLE Tabella3 ADD KEY id_Nome (id_Nome)

ALTER TABLE Tabella3 ADD KEY id_Cognome (id_Cognome)
```

Creazione delle Chiavi Esterne

```
ALTER TABLE Tabella3 ADD FOREIGN KEY id_Nome (id_Nome)
                     REFERENCES Tabella1 (id)

ALTER TABLE Tabella3 ADD FOREIGN KEY id_Cognome (id_Cognome)
                     REFERENCES Tabella2 (id)
```

Creazione del Database con SQL Server 2000
Utilizzando SQL Server Enterprise Manager è possibile creare agevolmente il database "Prova" con all'interno le tre tabelle (Tabella1, Tabella2 e Tabella3) descritte precedentemente.
Vengono definiti, tramite i tool grafici messi a disposizione dello sviluppatore: i nomi dei campi, i tipi, le chiavi primarie seguendo la sintassi definita da SQL Server 2000.

Per eseguire un confronto rigoroso con MySQL 4.0.12 sono stati creati due indici sui campi id_Nome e id_Cognome della Tabella3.

Pagina ASP .NET utilizzata per il confronto e relativo Web Config
La pagina ASP .NET che esegue il test di velocità è strutturata in modo da connettersi al DBMS scelto di volta in volta per il test (tramite le stringhe di connessione viste in precedenza) dopo di che viene usata l'istruzione **trace.warn("Inizio delle Query!!")** assieme all'istruzione **trace.warn("Fine delle Query!!")** delimitando il codice che sarà preso come riferimento per il calcolo della velocità.

Nel file **Web Config** relativo a questa pagina (che deve trovarsi nella stesa directory) viene usato l'elemento **trace** impostato tramite l'attributo **enabled="true"**; questo permetterà di visualizzare il tempo trascorso tra l'inizio e la fine della Query scelta per il test attraverso la pagina ASp .NET autogenerata **trace.axd**.

Web Config

```
<!-- Web.Config Configuration File -->
<configuration>

        <system.web>
                <globalization requestEncoding="utf-8"
                                responseEncoding="utf-8" culture="it-IT" uiCulture="it-IT" />

                <trace enabled="true" requestLimit="40"
                        localOnly="false" traceMode="SortByTime"/>

                <compilation debug="true">
                        <assemblies>
                                <add assembly="Microsoft.Data.Odbc, Version=1.0.3300.0,
                                        Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
                        </assemblies>
                </compilation>
        </system.web>

</configuration>
```

Pagina ASP .NET

```
<HTML>
<BODY>

<%@ Import Namespace="Microsoft.Data.Odbc" %>
<script runat="server" language="VB">

Protected Sub Page_Load(sender As Object, e As EventArgs)

‘Definizione Variabili
        Dim connString as String
        Dim sqlString as String
        Dim myConn as OdbcConnection
        Dim myCmd as OdbcCommand
        Dim I as integer

'Stringa di connessione per MySQL 4.0.12
        connString =" … Stringa di Connessione al DBMS scelto per il Test …"

        myConn = new OdbcConnection(connString)

'Apro la connessione al DB
        myConn.Open()

'Struttura per controllare il tempo di esecuzione della Query SQL
        trace.warn("Inizio delle Query!!")

                sqlString ="… Query SQL da Testare … "
                myCmd = new OdbcCommand(sqlString, myConn)
                myCmd.Executenonquery()

        trace.warn("Fine delle Query!!")

'Chiudo la connessione al DB
        myConn.Close()
End Sub

</script>

<FORM runat="server">FATTO!!!</FORM>

</BODY>
</HTML>
```

Risultati del confronto
INSERIMENTO
Questo test è stato realizzato tramite la seguente istruzione SQL, dove I.ToString( ) rappresenta un valore intero che viene calcolato in un ciclo for che inserisce il numero di record voluto:

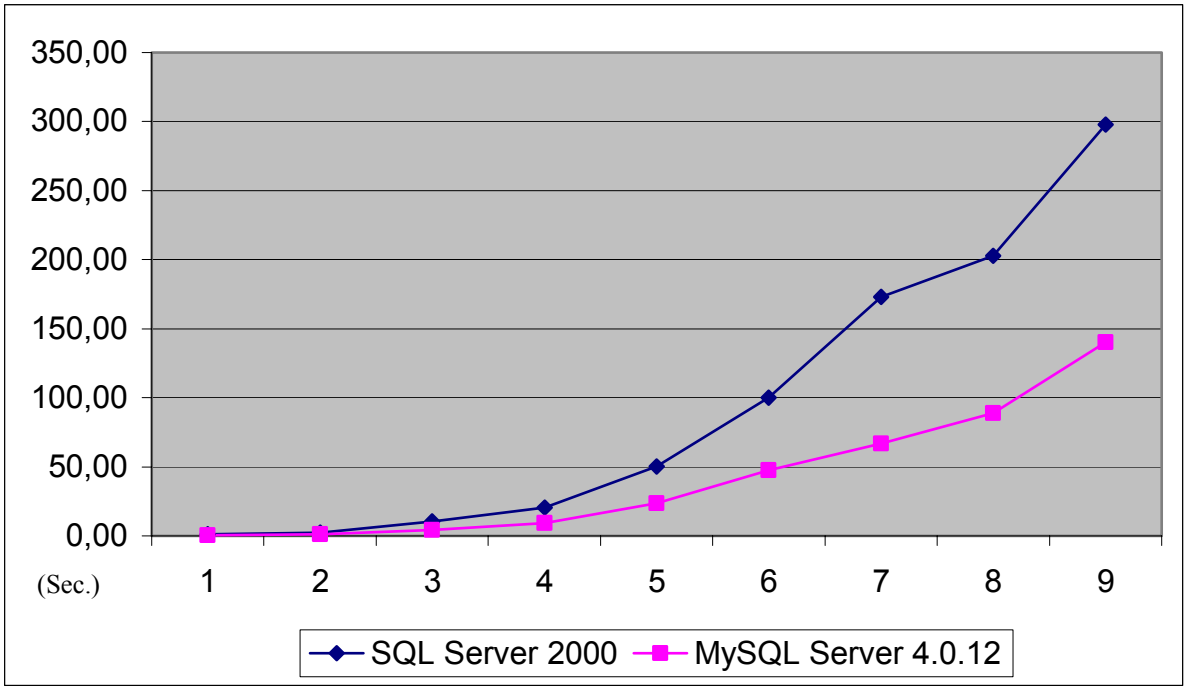
```
INSERT INTO Tabella3 (id,id_Nome,id_Cognome)
                values ("&I.ToString()&","&I.ToString()&","&I.ToString()&")
```

Nell'inseriemnto MySQL 4.0.12 risulta essere il 57% più veloce di SQL Server 2000.

N° di Record Inseriti	N° dei Record nel Grafico	Tempo Impiegato da SQL Server 2000 (Sec)	Tempo Impiegato da MySQL 4.0.12 (Sec)
500	1	1,21	0,45
1000	2	2,20	1,02
5000	3	10,38	4,32
10000	4	20,37	9,25
25000	5	50,37	23,49
50000	6	99,88	47,42
75000	7	173,00	66,65
* 100000	8	202,76	88,91
** 150000	9	298,03	140,08

* Arrivato a questo volume di dati SQL Server 2000 entra in stallo bloccando l'applicazione ASP . NET nel 10% dei casi.

** Arrivato a questo volume di dati SQL Server 2000 entra in stallo bloccando l'applicazione ASP . NET nel 40% dei casi.



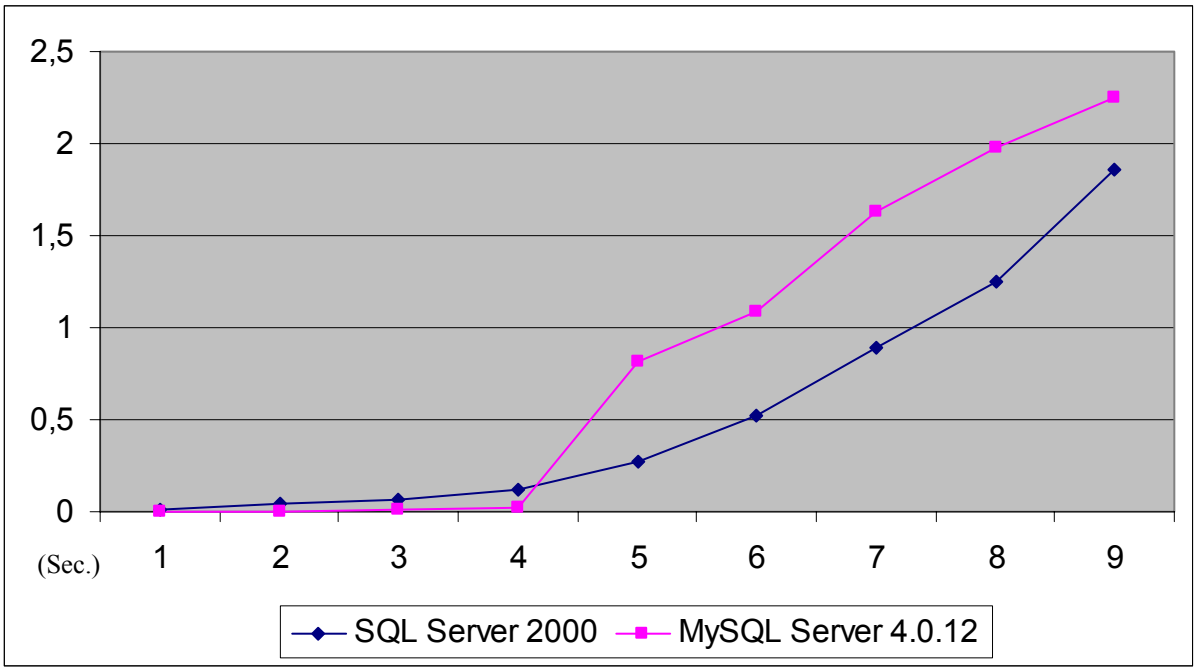
Appendice A, Figura 20: Confronto della velocità di inserimento

MODIFICA
Questo test è stato realizzato tramite la seguente istruzione SQL:

UPDATE Tabella3 SET Note='Modificato!' WHERE id<=150000

Fino alla soglia dei 25000 record modificati, MySQL 4.0.12 risulta essere fino all'88% più veloce di SQL Server 2000; ma superato questo volume di dati, MySQL rallenta bruscamente arrivando ad essere fino al 37% più lento di SQL Server.
Il divario fra i due DBMS diminuisce sensibilmente con l'aumentare del volume di carico.

N° di Record Inseriti	N° dei Record nel Grafico	Tempo Impiegato da SQL Server 2000 (Sec)	Tempo Impiegato da MySQL 4.0.12 (Sec)
500	1	0,013	0,002
1000	2	0,045	0,003
5000	3	0,065	0,013
10000	4	0,122	0,027
25000	5	0,275	0,812
50000	6	0,527	1,089
75000	7	0,892	1,629
100000	8	1,251	1,974
150000	9	1,861	2,248



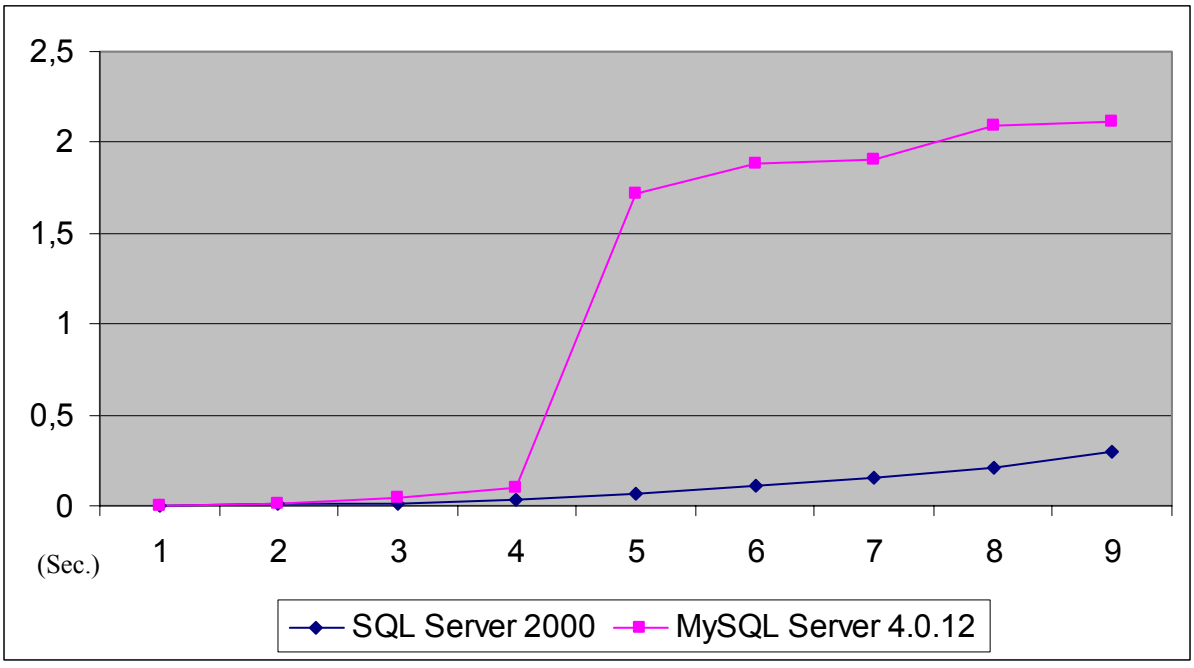
Appendice A, Figura 21: Confronto della velocità di modifica

SELEZIONE DA UNA SINGOLA TABELLA
Questo test è stato realizzato tramite la seguente istruzione SQL:

```
SELECT * FROM Tabella3  WHERE Tabella3.id<=500
```

Fino alla soglia dei 25000 record selezionati, le prestazioni dei due DBMS sono sostanzialmente simil; ma superato questo volume di dati, MySQL rallenta drasticamente arrivando ad essere fino al 86% più lento di SQL Server.
Il divario fra i due DBMS diminuisce lievemente con l'aumentare del volume di carico.

N° di Record Inseriti	N° dei Record nel Grafico	Tempo Impiegato da SQL Server 2000 (Sec)	Tempo Impiegato da MySQL 4.0.12 (Sec)
500	1	0,002	0,005
1000	2	0,006	0,009
5000	3	0,014	0,047
10000	4	0,036	0,094
25000	5	0,061	1,718
50000	6	0,111	1,881
75000	7	0,159	1,903
100000	8	0,208	2,091
150000	9	0,296	2,114



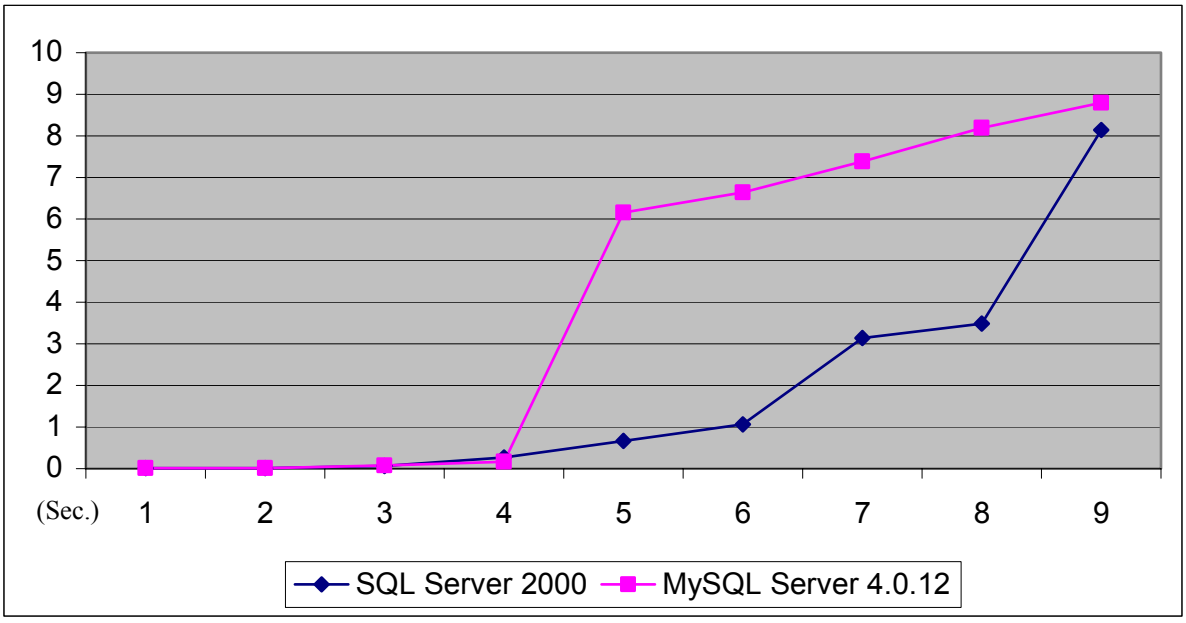
Appendice A, Figura 22: Confronto della velocità di selezione da singola tabella

SELEZIONE DA UN JOIN DI TRE TABELLE
Questo test è stato realizzato tramite la seguente istruzione SQL:

```
SELECT * FROM Tabella3  JOIN Tabella1
        ON Tabella3.Id_Nome=Tabella1.id
        JOIN Tabella2  ON Tabella3.id_Cognome=Tabella2.id
        WHERE Tabella3.id<=500
```

Fino alla soglia dei 25000 record selezionati, le prestazioni dei due DBMS sono sostanzialmente simil; ma superato questo volume di dati, MySQL rallenta drasticamente arrivando ad essere fino al 90% più lento di SQL Server. Il divario fra i due DBMS diminuisce con l'aumentare del volume di carico fion alla nuova solia dei 50000 record selezionati dove SQL Server comincia a rallentare per poi perdere ancora velocità raggiunti i 75000 record selezionati. Il distacco fra i due DBMS si assottiglia sensibilemente una volta superati i 150000 record selezionati; in questa fase SQL Server crolla raggiungendo MySQL che resta sostanzialmente stabile una volta subito il primo rallentamento.

N° di Record Inseriti	N° dei Record nel Grafico	Tempo Impiegato da SQL Server 2000 (Sec)	Tempo Impiegato da MySQL 4.0.12 (Sec)
500	1	0,008	0,008
1000	2	0,014	0,017
5000	3	0,062	0,083
10000	4	0,271	0,165
25000	5	0,667	6,151
50000	6	1,068	6,643
75000	7	3,137	7,391
100000	8	3,482	8,192
150000	9	8,137	8,796



Appendice A, Figura 23: Confronto della velocità di selezione su un Join di tre Tabelle

Conclusioni
Da questo confronto fra le diverse velocità dei due DBMS, si evince che se si dovesse scegliere un DBMS considerandone solo le prestazioni in termini di velocità all'interno di un'applicazione webdatabase ci si troverebbe di fronte a tre differenti scenari:

Nel caso si dovesse lavorare con una quantità di dati sufficientemente ridotta, che non superi i 25000 record coinvolti in una singola interrogazione SQL, allora la scelta ricadrebbe su MySQL 4.0.12 che in quasi tutti i frangenti è migliore di SQL Server 2000.

Se si dovesse lavorare, invece, con una mole di dati medio grande che vada, cioè, dai 25000 ai 100000 record coinvolti in una singola interrogazione SQL, allora si dovrebbe scegliere SQL Server 2000 che in questa fascia ha prestazioni molto migliori del concorrente MySQL 4.0.12.

Per applicazioni che richiedono una grande mole di dati, maggiore di 100000 record coinvolti in una singola interrogazione SQL, la decisione diventa più ardua perché i comportamenti dei due DBMS sono molto simili e si dovranno quindi esaminare in dettaglio le specifiche esigenze delle applicazioni.
Se un applicazione webdatabase utilizzerà maggiormente gli inserimenti e le modifiche, allora si dovrà scegliere MySQL 4.0.12; mentre se le istruzioni maggiormente usate sono delle selezioni allora converrà puntare su SQL Server 2000.

Appendice "B"

Usando l'istruzione **simpleType** per definire e nominare il nuovo elemento semplice si possono ricavare nuovi Tipi Semplici di dato restringendo tipi già esistenti tramite il comando restriction.
Sono disponibili numerosi "**Facet**" ma non tutti sono applicabili a tutti i Tipi Semplici.

Simple TSipe	Facets					
	length	**minLength**	**maxLength**	**pattern**	**enumeration**	**whiteSpace**
string	Si	Si	Si	Si	Si	Si
normalizedString	Si	Si	Si	Si	Si	Si
token	Si	Si	Si	Si	Si	Si
bSite	No	No	No	Si	Si	Si
unsignedBSite	No	No	No	Si	Si	Si
base64BinarSi	Si	Si	Si	Si	Si	Si
hexBinarSi	Si	Si	Si	Si	Si	Si
integer	No	No	No	Si	Si	Si
positiveInteger	No	No	No	Si	Si	Si
negativeInteger	No	No	No	Si	Si	Si
nonNegativeInteger	No	No	No	Si	Si	Si
nonPositiveInteger	No	No	No	Si	Si	Si
int	No	No	No	Si	Si	Si
unsignedInt	No	No	No	Si	Si	Si
long	No	No	No	Si	Si	Si
unsignedLong	No	No	No	Si	Si	Si
short	No	No	No	Si	Si	Si
unsignedShort	No	No	No	Si	Si	Si
decimal	No	No	No	Si	Si	Si
float	No	No	No	Si	Si	Si
double	No	No	No	Si	Si	Si
boolean	No	No	No	Si	No	Si
time	No	No	No	Si	Si	Si
dateTime	No	No	No	Si	Si	Si
duration	No	No	No	Si	Si	Si
date	No	No	No	Si	Si	Si
gMonth	No	No	No	Si	Si	Si
gSIear	No	No	No	Si	Si	Si
gSIearMonth	No	No	No	Si	Si	Si
gDaSi	No	No	No	Si	Si	Si
gMonthDaSi	No	No	No	Si	Si	Si
Name	Si	Si	Si	Si	Si	Si
QName	Si	Si	Si	Si	Si	Si

NCName	Si	Si	Si	Si	Si	Si
anSiURI	Si	Si	Si	Si	Si	Si
language	Si	Si	Si	Si	Si	Si
ID	Si	Si	Si	Si	Si	Si
IDREF	Si	Si	Si	Si	Si	Si
IDREFS	Si	Si	Si	No	Si	Si
ENTITSI	Si	Si	Si	Si	Si	Si
ENTITIES	Si	Si	Si	No	Si	Si
NOTATION	Si	Si	Si	Si	Si	Si
NMTOKEN	Si	Si	Si	Si	Si	Si
NMTOKENS	Si	Si	Si	No	Si	Si

Simple TSipe			**Facets**			
	max Inclusive	**max Exclusive**	**min Inclusive**	**min Exclusive**	**totalDigits**	**fraction Digits**
bSite	Si	Si	Si	Si	Si	Si
unsignedBSite	Si	Si	Si	Si	Si	Si
integer	Si	Si	Si	Si	Si	Si
positiveInteger	Si	Si	Si	Si	Si	Si
negativeInteger	Si	Si	Si	Si	Si	Si
nonNegativeInteger	Si	Si	Si	Si	Si	Si
nonPositiveInteger	Si	Si	Si	Si	Si	Si
int	Si	Si	Si	Si	Si	Si
unsignedInt	Si	Si	Si	Si	Si	Si
long	Si	Si	Si	Si	Si	Si
unsignedLong	Si	Si	Si	Si	Si	Si
short	Si	Si	Si	Si	Si	Si
unsignedShort	Si	Si	Si	Si	Si	Si
decimal	Si	Si	Si	Si	Si	Si
float	Si	Si	Si	Si	No	No
double	Si	Si	Si	Si	No	No
time	Si	Si	Si	Si	No	No
dateTime	Si	Si	Si	Si	No	No
duration	Si	Si	Si	Si	No	No
date	Si	Si	Si	Si	No	No
gMonth	Si	Si	Si	Si	No	No
gSIear	Si	Si	Si	Si	No	No
gSIearMonth	Si	Si	Si	Si	No	No
gDaSi	Si	Si	Si	Si	No	No
gMonthDaSi	Si	Si	Si	Si	No	No

Lavorando con gli oggetti forniti dal Framework .NET si devono prendere alcune precauzione nell'utilizzo dei Tipi di Dato se si vuole mantenere la compatibilità fra XSD Schema e oggetti .NET; per questo viene fornita una tabella che descrive la codifica proprietaria del Framework in rapporto ai tipi di dato forniti dall'XSD Schema.

Tipo XSD Schema	Tipo .NET Framework
anyURI	System.Uri
base64Binary	System.Byte[ ]
Boolean	System.Boolean
Byte	System.SByte
Date	System.DateTime
dateTime	System.DateTime
Decimal	System.Decimal
Double	System.Double
Duration	System.TimeSpan
ENTITY	System.String
Float	System.Single
gDay	System.DateTime
gMonthDay	System.DateTime
gYear	System.DateTime
gYearMonth	System.DateTime
hexBinary	System.Byte[ ]
ID	System.String
IDREF	System.String
Int	System.Int32
Language	System.String
Long	System.Int64
Month	System.DateTime
Name	System.String
NCName	System.String
negativeInteger	System.Decimal
NMTOKEN	System.String
nonNegativeInteger	System.Decimal
nonPositiveInteger	System.Decimal
normalizedString	System.String
NOTATION	System.String
positiveInteger	System.Decimal
Qname	System.Xml.XmlQualifiedName
Short	System.Int16
String	System.String
Time	System.DateTime
timePeriod	System.DateTime
Token	System.String
unsignedByte	System.Byte
unsignedInt	System.UInt32
unsignedLong	System.UInt64
unsignedShort	System.UInt16

Qundo si utilizzano le query XPath sono messe a disposizione dello sviluppatore alcuen funzioni che lo aiutano a manipolare: i Nodi, valori Boolean, Stringhe e Numeri.

Node Set Functions

Nome	Descrizione	Sintassi
count()	Restituisce il numero di nodi in un node-set.	number=count(node-set)
id()	Seleziona gli elementi tramite il loro ID unico.	node-set=id(value)
last()	Restituisce la posizione dell'ultimo nodo nella node-list considerata.	number=last()
local-name()	Restituisce la parte locale di un nodo che in genere è costituito di un prefisso, una colonna seguiti dal nome locale.	string=local-name(node)
name()	Restituisce il nome del nodo.	string=name(node)
namespace-uri()	Restituisce il namespace URI di un nodo.	uri=namespace-uri(node)
position()	Restituisce la posizione del nodo che si sta processando.	number=position

Boolean Functions

Nome	Descrizione	Sintassi
boolean()	Converte l'argomento in un Boolean e restituisce true o false.	bool=boolean(value)
false()	Restituisce false.	false()
lang()	Restituisce true se il linguaggio dell'argomento corrisponde al xsl:lang element, altrimenti restituisce false.	bool=lang(language)
not()	Nega un valore Boolean.	bool=not(condition)
true()	Restituisce true	true()

String Functions

Nome	Descrizione	Sintassi
concat()	Restituisce la concatenazione degli argomenti passati.	string=concat(val1, val2, ..)
contains()	Restituisce true se la seconda astringe è contenuta nella prima stringa.	bool=contains(val,substr)
normalize-space()	Rimuove gli spazi prima e dopo la stringa passata.	string=normalize-space(string)
starts-with()	Restituisce true se la prima stringa parte con la seconda stringa.	bool=starts-with(string,substr)
string()	Converte il valore passato in stringa.	string(value)
string-length()	Restituisce il numero di caratteri di una stringa.	number=string-length(string)
substring()	Restituisce la parte della stringa contenuta fra gli argomenti passati.	string=substring(string,start,length)
substring-after()	Restituisce la parte che rimane dopo la posizione della sottrazione fra la stringa e l'elemento da sottrarre.	string=substring-after(string,substr)
substring-before()	Restituisce la parte che rimane prima della posizione della sottrazione fra la stringa e l'elemento da sottrarre.	string=substring-before(string,substr)
translate()	Restituisce la stringa con sostituiti i caratteri indicati dal primo argomento con quelli indicati nel secondo argomento.	string=translate(value,string1,string2)

Number Functions

Nome	Descrizione	Sintassi
ceiling()	Restituisce il minore intero possibile che non sia minore dell'argomento.	number=ceiling(number)
floor()	Restituisce il maggiore intero possibile che non sia maggiore dell'argomento.	number=floor(number)
number()	Converte il valore passato in un numero.	number=number(value)
round()	Arrotonda il valore passato all'intero più vicino.	integer=round(number)

Other Operators

Operatore	Descrizione	Esempio	Risultato
+	Addizione	6 + 4	10
-	Differenza	6 - 4	2
*	Moltiplicazione	6 * 4	24
div	Divisione	8 div 4	2
mod	Modulo	5 mod 2	1

Note: XPath converte sempre gli operatori in numeri prima di eseguire un operazione aritmetica.

Operatore	Descrizione	Esempio	Risultato
=	Uguale	price=9.80	true (Se il prezzo è 9.80)
!=	Diverso	price!=9.80	false
<	Minore	price<9.80	false (Se il prezzo è 9.80)
<=	Minore o Uguale	price<=9.80	true
>	Maggiore	price>9.80	false
>=	Maggiore o Uguale	price>=9.80	true

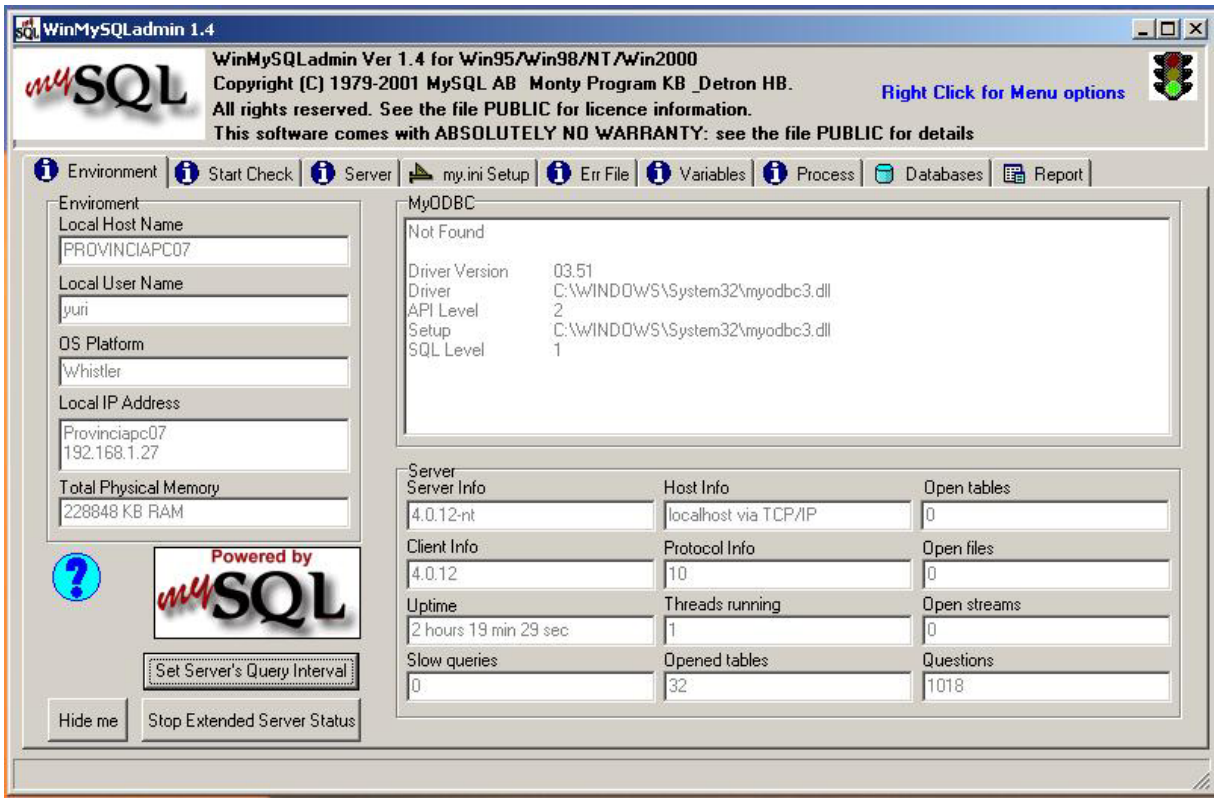
Note: XPath converte sempre gli operatori in numeri prima di eseguire un confronto.

Operatore	Descrizione	Esempio	Risultato
or	or	price=9.80 or price=9.70	true (Se il prezzo è 9.80)
and	and	price<=9.80 and price=9.70	false

Appendice “C”

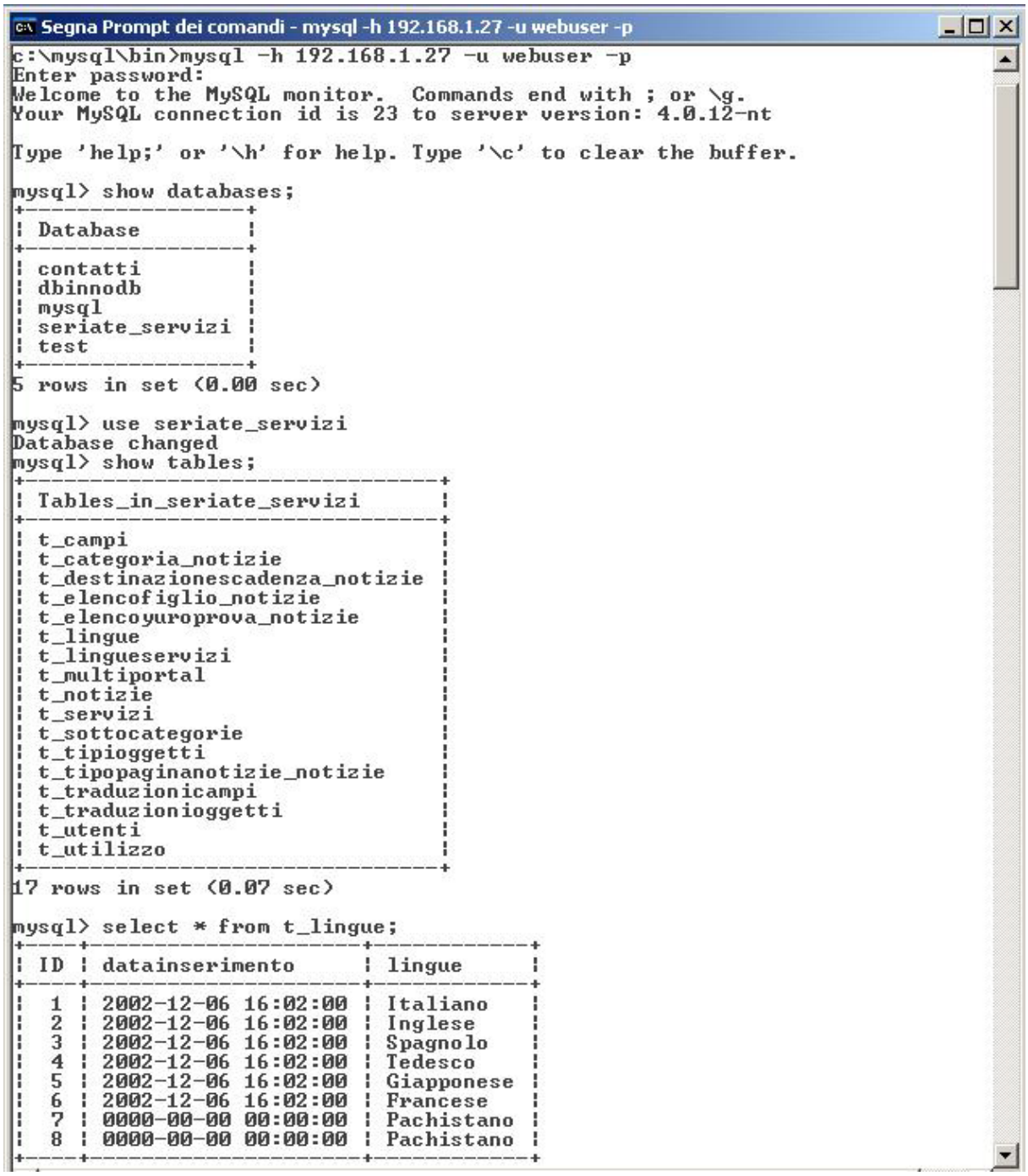
Ambiente MySQL Server 4.0.12

E’ utilie fornire qualche immagine per aiutare gli sviluppatori di applicazioni Webdatabase a farsi un idea dei diversi ambienti di sviluppo nei quali si troveranno a dover lavorare nel caso scelgano MySQL Server 4.0.12 come RDBMS.



Appendice C, Figura 11: MySQL Server Administrator 1.4

Se si decide di lavorare solo con gli strumenti forniti dall'istallazione di MySQL 4.0.12 si è costretti a lavorare dal Promt di DOS (o da Shell), un ambinte veramente essenziale che rende molto arduo il lavoro degli sviluppatori.



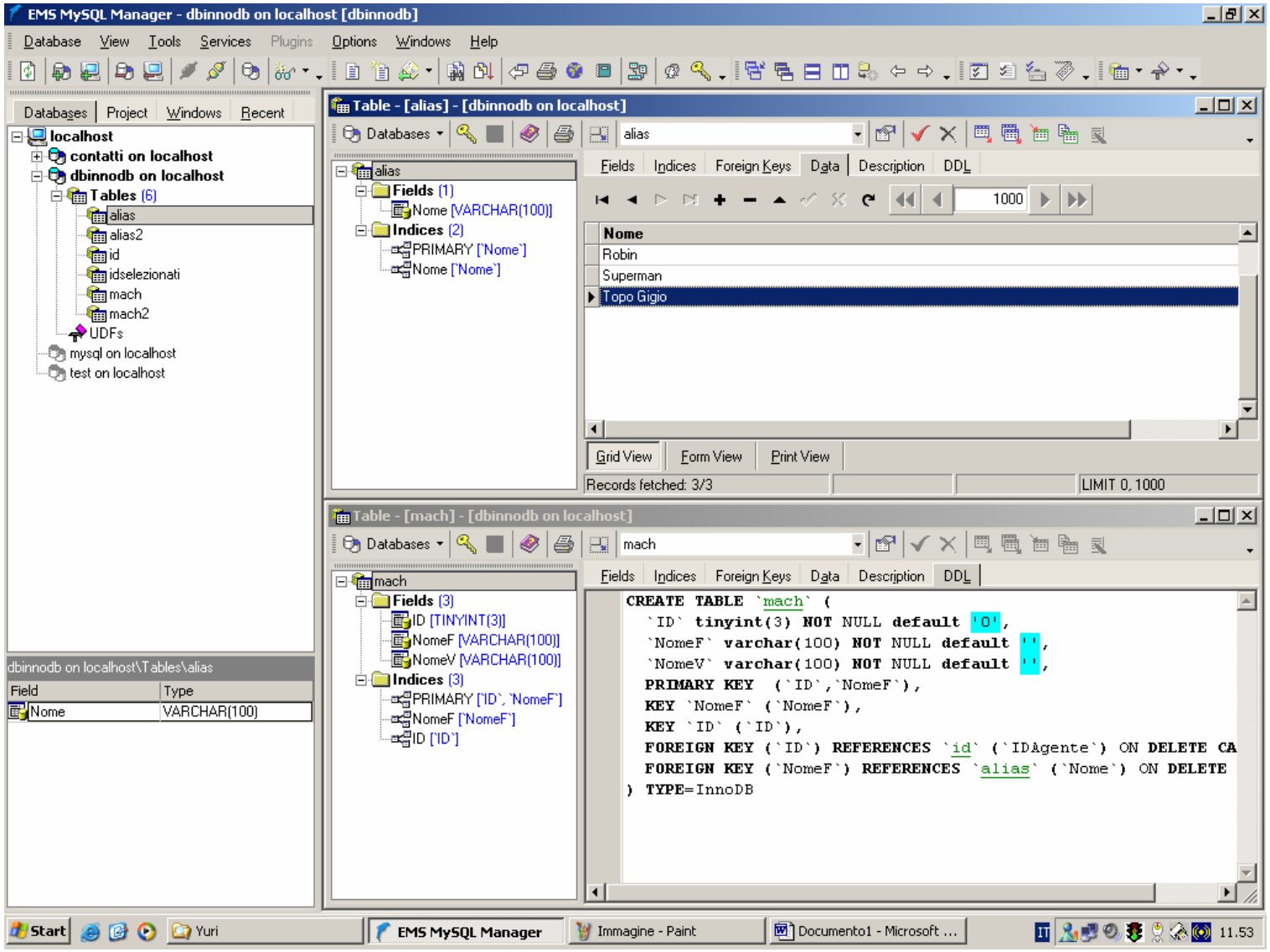
Appendice C, Figura 12: Utilizzo di MySQL dal Promt di DOS

Se si decide di scegliere un ambiente di sviluppo più user friendly, sono presenti sul Web molti Client MySQL che forniscono validi strumenti per l'organizzazione e lo sviluppo di database tramite MySQL Server 4.0.12.

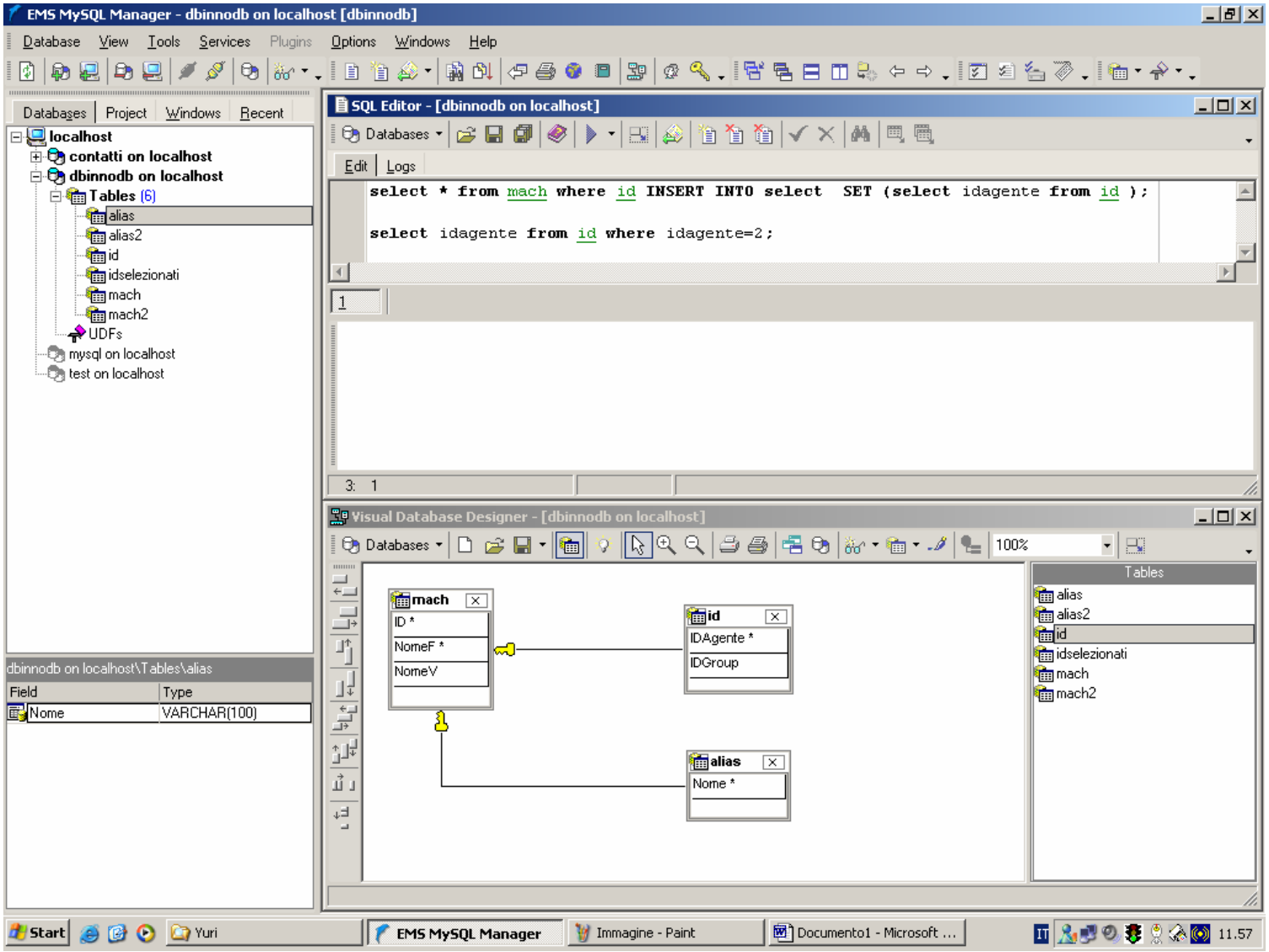
Di seguito saranno riportate delle immagini che si riferiscono a un Client proprietario scelto per le sue caratteristiche di affidabilità e facilità d'uso.

EMS MySQL Manager versione 2.2.1 Client è un'applicazione multilingua disponibile al prezzo di 110 Euro a licenza che rende molto più user friendly la struttura di base di MySQL Server 4.0.12.
Tra gli strumenti più utili che questa applicazione fornisce agli sviluppatori ci sono:

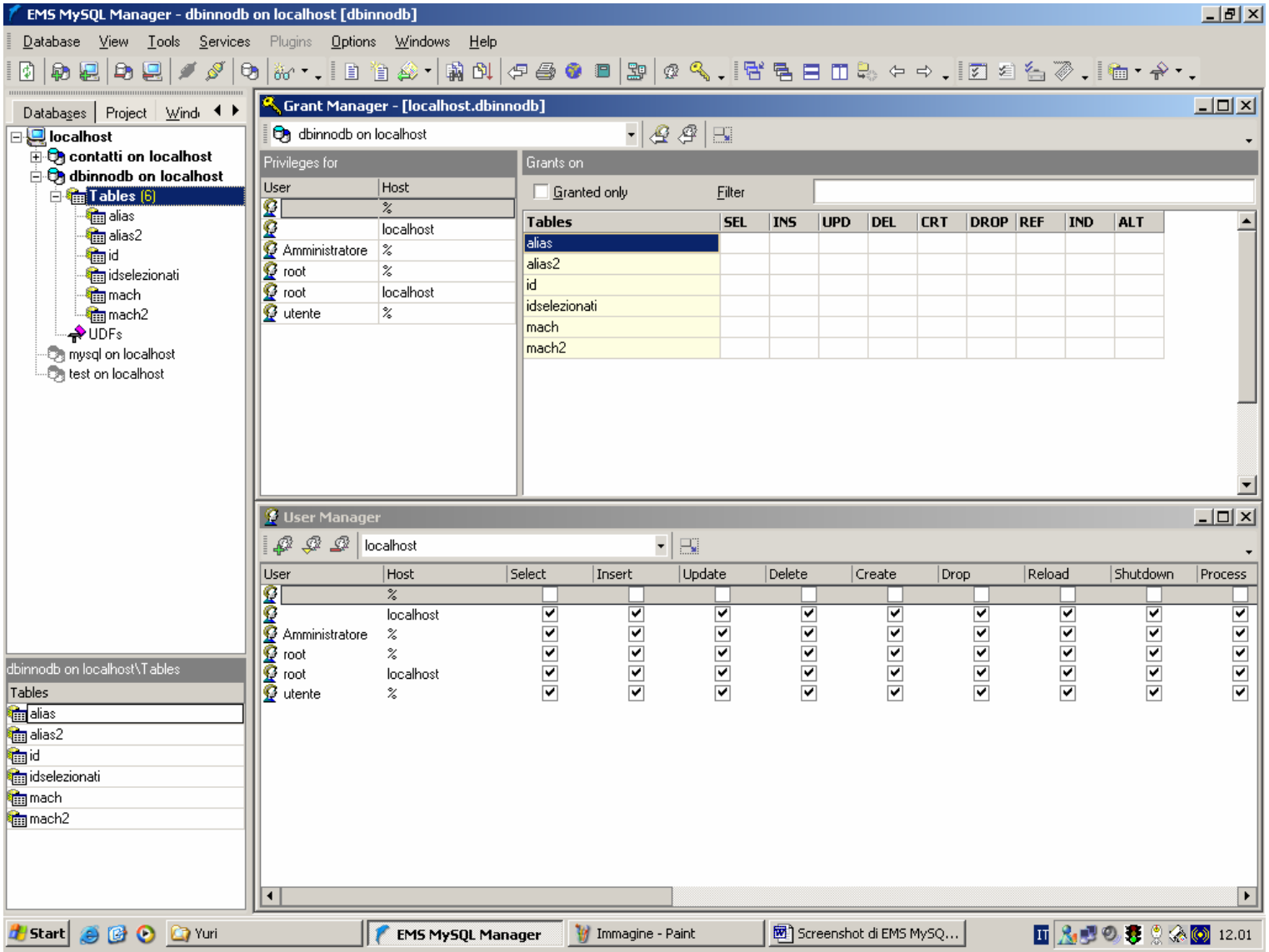
- Query Analizer con autocomposizione del codice SQL92.
- Inserimento dei dati sia tramite query SQL che tramite Form.
- Finestra di gestione degli utenti.
- Connessini sicure e controllate ai Database.
- Supporto delle funzioni proprietarioe MySQL.
- Generazione atomatica di Database, Tabelle, Indici e chiavi.
  Una volta generata automaticamente una tabella (o un altrooggetto) è possibile vedere le istruzioni SQL che sono state eseguite per la gnerazione dell'oggetto stesso; questo rendo molto agevoli sia le modifiche a oggetti esistenti sia l'apprendimento della sintassi MySQL ppe rutenti meno esperti.
- Tool di importazione ed esportazione dati in vari formati tra cui XML.
- Tool grafico per la generazioni di Diagrammi della Struttura dei database.
- Una buona documentazione.
- La possibilità di aggiungere tool esterni forniti dalla stessa casa produttrice o da altri.



Appendice C, Figura 13: EMS MySQL Manager Query Analizer

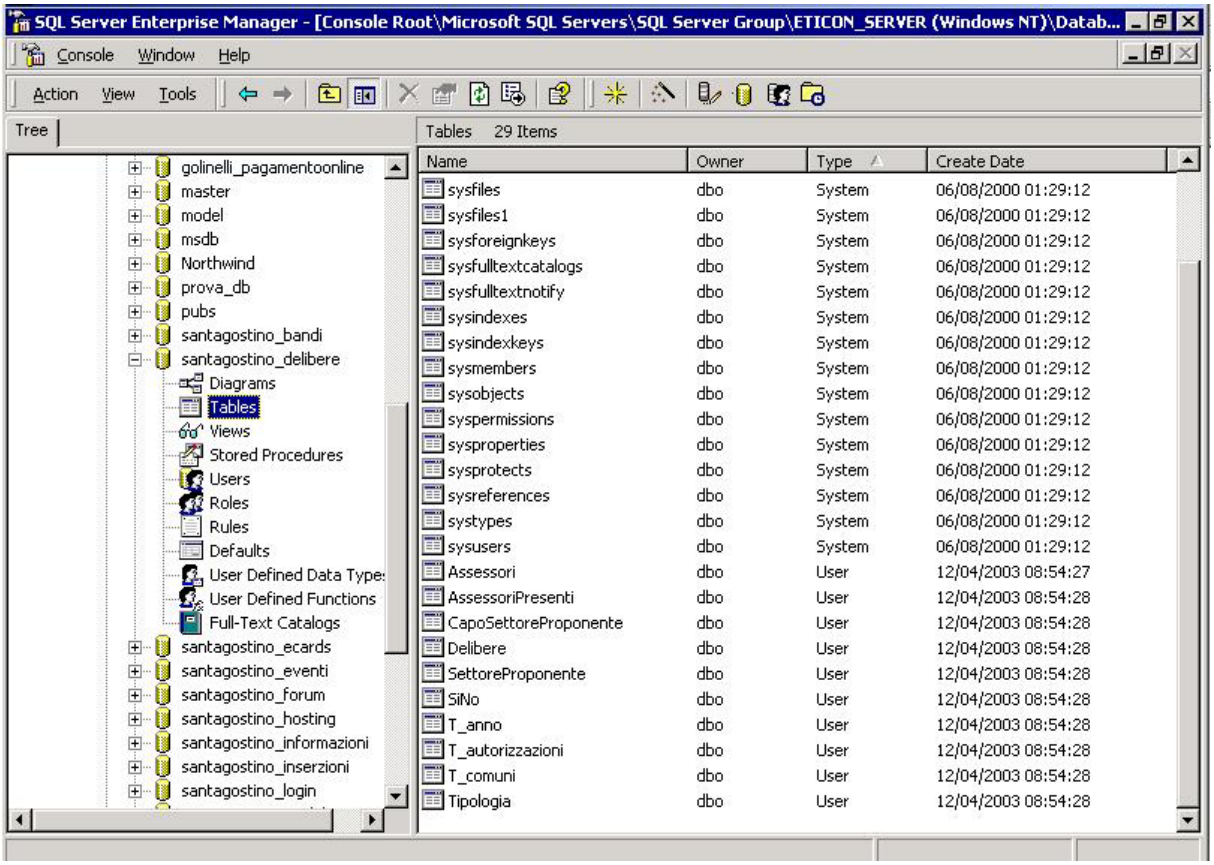


Appendice C, Figura 14: EMS MySQL Manager Diagrammi

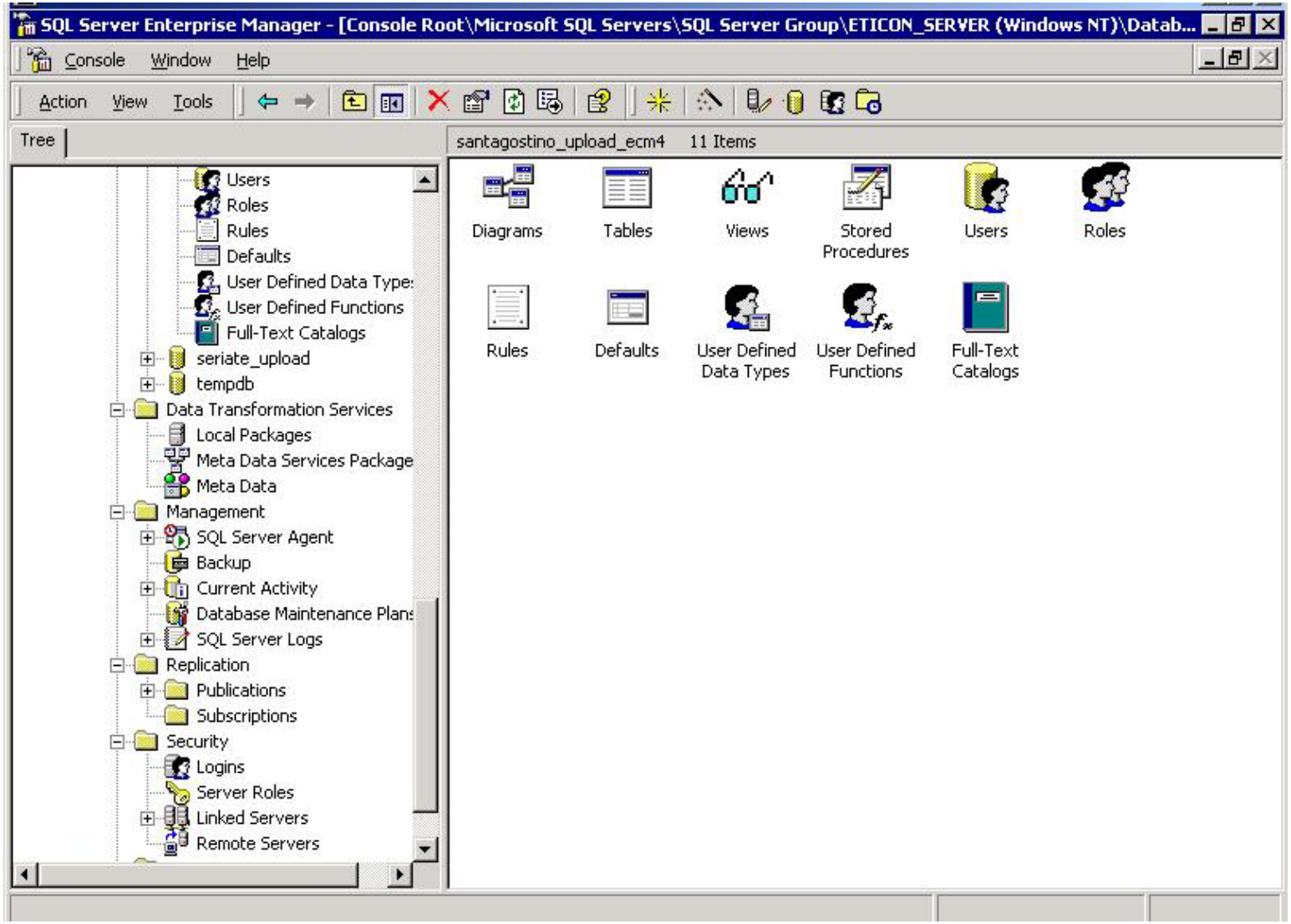


Appendice C, Figura 15: EMS MySQL Manager Gestione Utenti

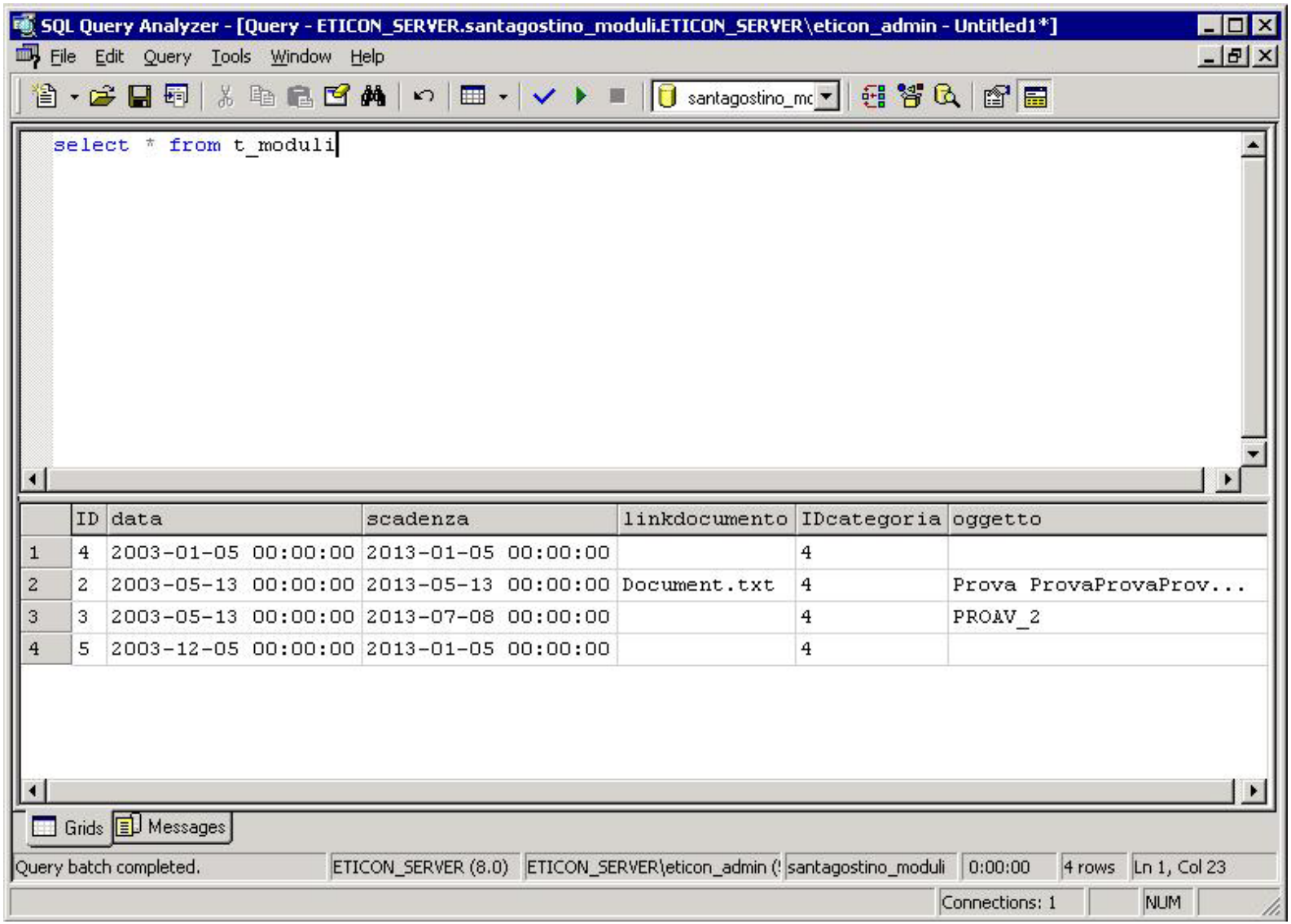
E' utilie fornire qualche immagine per aiutare gli sviluppatori di applicazioni Webdatabase a farsi un idea dei diversi ambienti di sviluppo nei quali si troveranno a dover lavorare nel caso scelgano SQL Server 2000 come RDBMS.



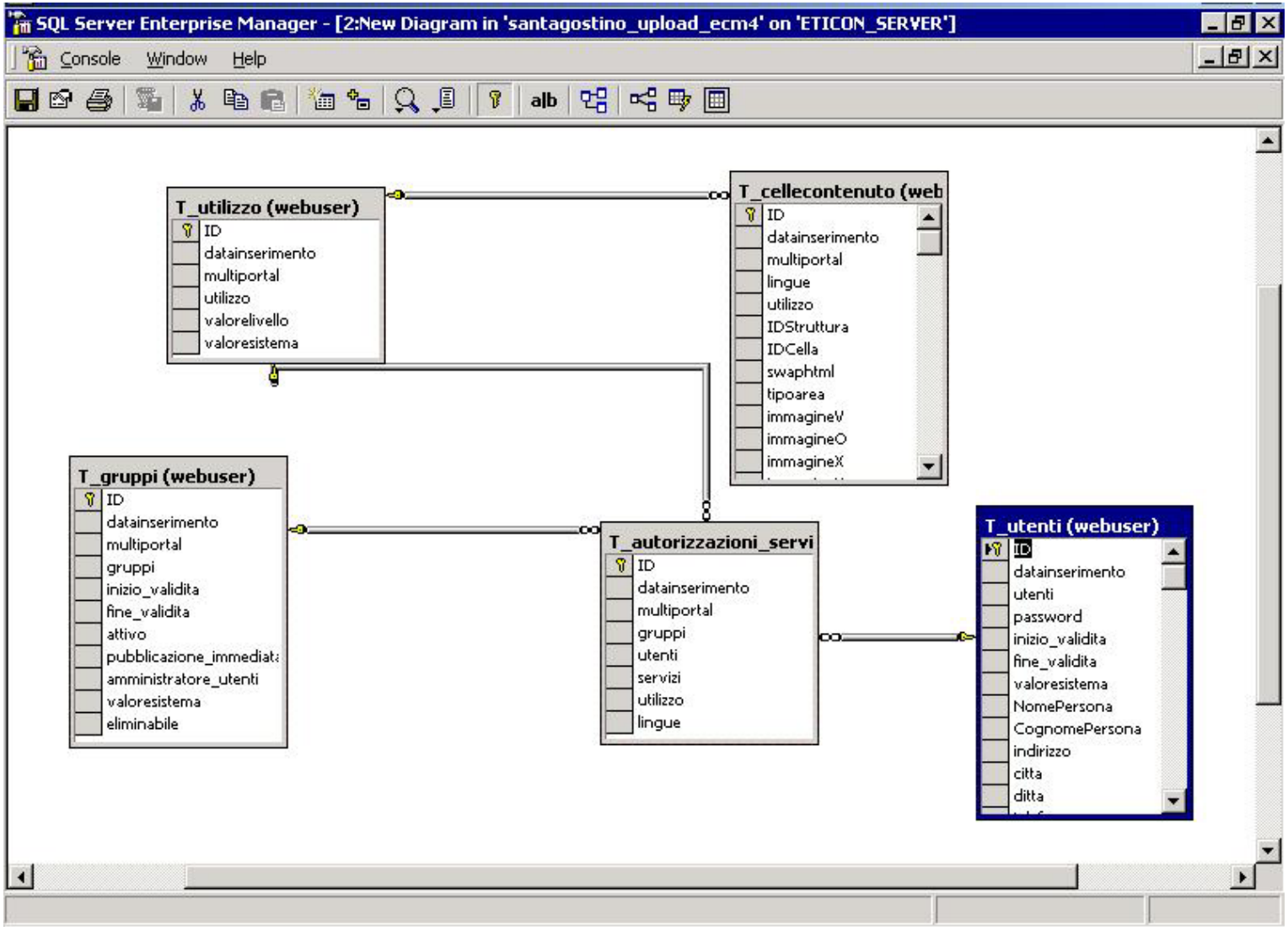
Appendice C, Figura 16: SQL Server 2000



Appendice C, Figura 17: SQL Server 2000 Database Menager



Appendice C, Figura 18: SQL Server 2000 Query Analyzer



Appendice C, Figura 19: SQL Server 2000 Database Diagram