Indice

PREFAZIONE

In occasione della mia permanenza alla MB Scambi Culturali come stagista mi è stato affidato l'incarico di migrare il database aziendale dalla piattaforma MS Access 2003 a SQL Server 2005, per poter ottimizzare le prestazioni e per poterlo rendere accessibile dal Web.

Mentre sto scrivendo queste righe, il mio progetto di lavoro è ultimato e posso affermare di aver guadagnato una certa esperienza, non solo nell'affrontare le problematiche legate alla migrazione e alla pubblicazione del database aziendale, ma anche nel risolvere tutti i piccoli problemi con cui ci si scontra ogni giorno, quando si è in molti ad usare gli stessi strumenti (nel mio caso il database), per scopi diversi.

Purtroppo, non mi è possibile elencare tutte le problematiche nate in questo contesto, a causa dell'elevata specificità di alcuni dei problemi incontrati, quindi con il seguente documento mi limiterò a scrivere una traccia che potrà essere utile a chi dovesse scontrarsi con difficoltà simili a quelle da me affrontate in questo stage, concentrandomi su aspetti di carattere generale e approfondendo solo alcuni casi specifici, che hanno suscitato in me un certo interesse.

Voglio approfittare di questo spazio per comunicare quella che a mio avviso è la cosa più importante che ho imparato in questo periodo: *"Se hai un problema basta chiedere"*.
Quando si lavora con strumenti informatici particolarmente complessi, è assurdo pensare di poter risolvere qualsiasi problema con le proprie forze: agendo in questo modo si ottiene soltanto di uno spreco di tempo ed energie; il web è pieno di comunità virtuali, di persone disposte a condividere la propria esperienza, oltre che a fornire **gratuitamente** documentazioni (in molti casi particolarmente curate e precise), grazie alle quali è possibile superare con una certa facilità quelli che sono i problemi più comuni con cui ci si scontra. In questo modo è possibile risparmiare gli sforzi per quelli che sono i problemi più specifici, su cui è più difficile ottenere un aiuto concreto.

Proprio perché ho sempre trovato persone disposte ad aiutarmi gratuitamente, ho deciso di rendere pubblico questo mio documento, nella speranza che possa essere di aiuto a chi dovesse affrontare problemi simili a quelli da me incontrati.

Giovanni Ranzato 6 ottobre 2007

INTRODUZIONE

MS Access è impiegato, nella maggioranza dei casi, in ambito individuale o in applicazioni usate da piccoli gruppi di utenti; in questi contesti risulta essere quasi sempre appropriato. In una piccola percentuale dei casi, tuttavia, è più conveniente costruire un applicazione più sofisticata. Ciò accade quando l'ambito di riferimento cresce e iniziano a sorgere dei bisogni per i quali un'applicazione Access inizia ad essere insufficiente; nasce quindi il bisogno di spostarsi (d'ora in poi diremo migrare) in una piattaforma più potente; una possibile soluzione, che garantisce maggiore scalabilità, affidabilità e sicurezza, è data da **MS SQL Server**.

Quando Migrare?

Come può un'azienda, o un DBA (Data Base Administrator), comprendere quando è giunto il momento di migrare il proprio database?
In questo paragrafo ci occuperemo di rispondere a questa domanda, illustrando

quali sono le motivazioni che possono spingere a prendere la decisione di cambiare il proprio DBMS (data base management System) da Access a SQL Server.

La prima cosa importante da comprendere, è che non tutte le applicazioni Access necessitano di essere migrate. La migrazione comporta dei costi di tempo e di denaro, pertanto bisogna valutare con attenzione la decisione di migrare il proprio database; in genere tutte le applicazioni che non necessitano di attributi come la **scalabilità** e l'**affidabilità** non hanno bisogno di essere migrate.

Scalabilità

La Scalabilità, può essere definita come la capacità di un'applicazione di funzionare in maniera efficiente, anche quando il numero di utenti e processi con cui interagisce cresce. Per comprendere le ragioni che possono portare alla decisione di migrare il database, metteremo in evidenza i limiti imposti da Access, in modo da verificare che non rappresenta una soluzione scalabile.

Sebbene Access non precluda, formalmente, la gestione di database di dimensioni superiori a 100 MB (teoricamente dovrebbe supportare fino a 2 GB di dati), a quei livelli non riesce a garantire prestazioni soddisfacenti, obbligando l'utente a dover attendere lunghe fasi di caricamento durante l'apertura dell'applicazione, dei forms e dei report: ciò è dovuto al database-engine di Access (denominato *Jet*) costruito in un ambito File-Shared (file condivisi), che a differenza di una soluzione client/server come quella di MS SQL Server, non è ottimizzato per gestire grandi quantità di dati.
Un altro limite di Access, riguarda il numero di accessi simultanei che è in grado di gestire: Access può, a livello teorico, accettare fino a 255 connessioni per ogni database. In realtà, il numero di connessioni o utenti che può supportare è dettato da com'è stata progettata e implementata l'applicazione.

Un'applicazione professionale può arrivare a supportare fino a 20 utenti concorrenti, che aggiornano il database, mentre per database usati principalmente per operazioni di sola lettura, può accettare fino a 100 connessioni simultanee. Il problema è che la maggior parte delle applicazioni sviluppate in Access non sono progettate da professionisti, ma da utenti che non hanno l'esperienza e le conoscenze necessarie a costruire applicazioni professionali. Il risultato è che, in molti casi, le Applicazioni create sono in grado di gestire solo poche connessioni simultanee.

Abbiamo appena visto come, con i limiti imposti dal suo database-engine, Access non rappresenta certo una soluzione scalabile. Pertanto, possiamo comprendere come una delle motivazioni principali che spingono alla migrazione sia proprio quella di garantire maggior scalabilità alla propria applicazione.

Affidabilità

L'affidabilità rappresenta, per molte applicazioni, un attributo di vitale importanza (basta pensare a tutte le applicazioni che contengono dati critici) e deve essere garantita al 100%. Proprio questo risulta essere uno dei motivi chiave che spesso spinge a migrare: Access, infatti, per diverse ragioni non può essere considerata una soluzione affidabile.

Quando, durante l'esecuzione di un'applicazione Access, viene generato un errore di connessione, può succedere che il database venga corrotto; quando ciò accade, tutti gli utenti vengono tagliati fuori, e può verificarsi la perdita dei dati. Ciò rappresenta un serio problema, in quanto Access è molto esposto a questo rischio, soprattutto se l'applicazione è definita in ottica di multiutenza, nella quale tutti gli utenti sono simultaneamente connessi ai dati. Sebbene Access disponga di un apposito Tool per risolvere il problema, questo non sempre è sufficiente per riparare il database, quindi in alcuni casi si è costretti a rivolgersi a servizi offerti

da terzi, che comportano però di dover pagare un costo in denaro e di dover attendere che il database venga restaurato.
Per diminuire il rischio di corruzione, un database deve essere costantemente compattato. Questa operazione, però, richiede che l'applicazione venga chiusa, cosa che in alcuni casi non è possibile (quando il database deve rimanere attivo 24/24 7/7). In queste situazioni quindi risulta necessaria la migrazione a SQL Server.

L'architettura di Access soffre di un altro difetto: non esiste alcun meccanismo che consenta di avere una copia di backup del database, finché tutti gli utenti non si sono disconnessi. In un contesto di multiutenza, è difficile (o quantomeno scomodo e macchinoso) assicurarsi che tutti gli utenti siano disconnessi, prima di effettuare il backup. Un esempio tipico è quando gli utenti lasciano il computer accesso, durante la pausa pranzo: se anche un solo utente lascia il database connesso il software di backup non sarà in grado di eseguire una copia affidabile del database. Access inoltre non è in grado di riconoscere in automatico sprechi di spazio in memoria. Per ottimizzare indici e query è necessario *compattare* il database, ma questo come già detto richiede che l'applicazione venga chiusa. L'utilizzo di SQL Server, invece, consente di eseguire backup dinamici, incrementali o completi, del database in uso. Non è pertanto necessario uscire dal database (operazione tra altro impossibile da schedulare su un sito web in hosting) per eseguire il backup. In questo modo, il database può rimanere in esecuzione 24 ore su 24, sette giorni su sette.

Le motivazioni che possono spingere un azienda o il DBA al passaggio da Access a SQL Server, non sono tuttavia legate unicamente alla scalabilità o all'affidabilità; in alcuni casi possono nascere esigenze particolari, a cui Access non può far fronte semplicemente perché non è stato progettato appositamente per questo.
Oltre che per quanto detto finora, la migrazione rappresenta una soluzione ottimale quando ad esempio si inizia a parlare di protezione, recupero oppure di

pubblicare il database nel web.

Recupero immediato

Se si verifica un errore di sistema, come un arresto del sistema operativo o un'interruzione dell'alimentazione, Microsoft SQL Server dispone di un meccanismo di recupero automatico in grado di recuperare un database, allo stato in cui si trovava, in pochi minuti.

Protezione ottimale

Microsoft SQL Server usa un sistema di protezione attraverso degli schemi, che permettono una gestione ottimale dei permessi; può inoltre integrarsi con la protezione del sistema operativo Windows per fornire un accesso singolo alla rete e al database.
Ciò facilita notevolmente l'amministrazione di schemi di protezione complessi.

Accesso al Web

Una delle più comuni ragioni per migrare da Access a SQL Server è la necessità di rendere disponibili i dati da un browser attraverso una connessione a internet.
Sebbene Access non precluda la possibilità di essere impiegato per applicazioni web, a causa del limitato numero di connessioni simultanee che è in grado di supportare non risulta essere la scelta migliore.
SQL Server, invece, oltre a godere dei numerosi vantaggi elencati sopra, offre la possibilità di accettare un numero teoricamente infinito di connessioni, garantendo così alte performance in applicazioni Web.

Sito Web statico o Dinamico?

Nel paragrafo precedente abbiamo visto come una delle motivazioni che può spingere un'azienda a migrare da Access a SQL Server sia quella di usare il database in un'applicazione web: così facendo, possiamo generare il contenuto delle pagine in modo dinamico. Ma cosa significa generare pagine in modo dinamico? Che vantaggi comporta l'utilizzo di un database in un applicazione web? In questo paragrafo proveremo a rispondere a queste domande.

Navigare un sito web statico significa semplicemente interrogare un server tramite un browser (o, più genericamente, un client web) e visualizzare sul proprio computer le informazioni che il server dà come risposta, solitamente in formato HTML. Il server ospita i file HTML, le immagini e tutto ciò che compone ogni singola pagina; il navigatore accede a tali informazioni banalmente sfogliando i file nei quali esse sono contenute.

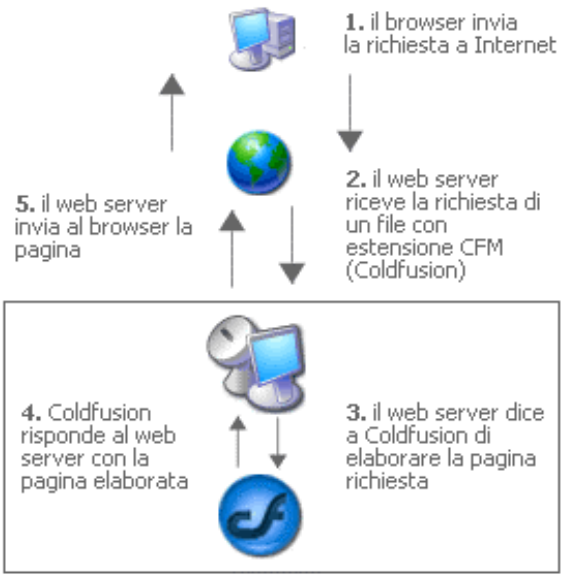
Per i siti web dinamici lo scopo finale è lo stesso: fornire informazioni in formato interpretabile da un browser. Cambia però il processo che restituisce queste informazioni. Il contenuto di un sito web dinamico, infatti, non è più quello che "fisicamente" risiede sul server (ad esempio, in una pagina HTML) ma esso viene generato in base ad una serie di parametri, comandi e condizioni che il server si preoccupa di valutare ad ogni richiesta che arriva dal browser. Una volta impacchettata in un formato interpretabile dal client (come l'HTML, appunto), la pagina generata può essere visualizzata tranquillamente dal visitatore.

Si parla di "generazione" di pagina "dinamica" perché un software installato sul server è in grado di creare, dinamicamente e a seconda delle richieste del

browser, pagine con contenuto variabile. Questo software si chiama **application server**, mentre web server è quello adibito ad inviare al client la pagina generata.
Il contenuto può risiedere in semplici file, ma anche in database o altre fonti di dati. Può essere il risultato di una elaborazione matematica, oppure una serie di informazioni provenienti da server remoti. In questa maniera è possibile separare il contenuto dalla sua visualizzazione, lasciando all'application server il lavoro di unire le informazioni per renderle accessibili al client.

La programmazione lato server si occupa di "istruire" (bisogna specificare dove e come visualizzare quali informazioni) l'application server sulla generazione delle pagine grazie ad un linguaggio specifico.
Nel nostro caso è stato usato ColdFusion, un application server sviluppato da Macromedia, per ColdFusion, ma anche per gli altri application server, come ASP o PHP. Vale lo schema rappresentato in **figura 1.1**, che esemplifica il processo di generazione di una pagina dinamica e i ruoli che assumono web server e application server.

figura 1.1: funzionamento application server



All'inizio del paragrafo ci siamo chiesti che vantaggi potesse recare la creazione di un sito dinamico tramite l'utilizzo di un database.

Un sito dinamico può essere più interattivo: potrà ad esempio consentire all'utente di "autenticarsi", lasciare commenti e personalizzare il sito, inviare e ricevere e-mail, accedere a fonti dati e molto altro ancora, cose che sarebbero impossibili da fare con il semplice linguaggio HTML; utilizzando un database per generare i contenuti di una pagina, godiamo di grande libertà di visualizzazione, manutenzione e aggiornamento.

A questo possiamo sommare la possibilità di realizzare motori di ricerca, o mantenere coerenza nell'interfaccia grafica: quando decidiamo di dare un aspetto alle informazioni contenute in un record del database, esso sarà identico a quello di tutti gli altri record.

Situazione in Azienda

Prima di analizzare le fasi del procedimento di migrazione, e in seguito di pubblicazione nel web, del database, è opportuno soffermarci ad analizzare la situazione in azienda, in modo da poter comprendere con maggior chiarezza innanzitutto i motivi che hanno reso necessaria la migrazione, poi il punto di partenza dal quale è stato avviato il progetto di migrazione, infine il motivo delle tante operazioni che sono state necessarie per completare il progetto di lavoro.

MB Scambi culturali è un azienda che da 10 anni lavora nel settore delle vacanze

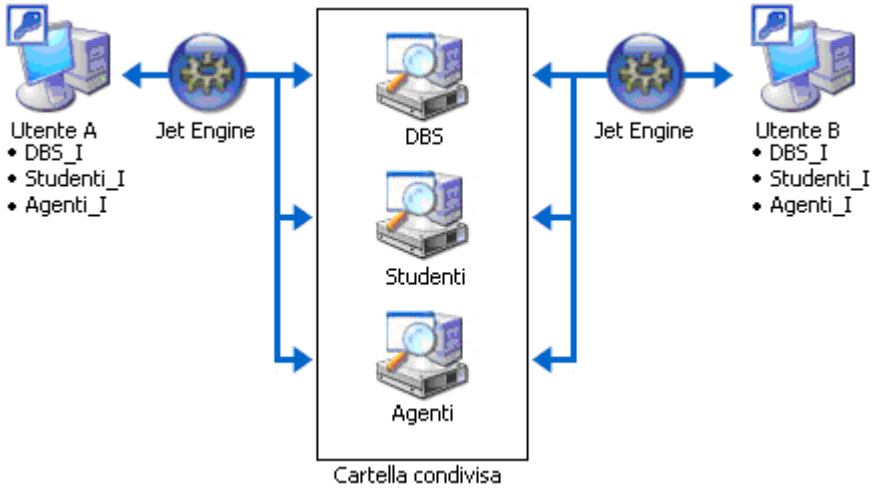
studio all'estero, offrendo programmi adatti a tutte le fasce di età e a tutti i livelli, che consentono di lavorare o studiare all'estero in qualsiasi periodo dell'anno, permettendo così all'utente di migliorare le proprie conoscenze, senza precludere la possibilità di vivere un esperienza gratificante e in molti casi divertente.
Il ruolo del database in agenzia è pressoché fondamentale: questo deve gestire l'anagrafica e i profili di tutti i contatti e deve memorizzare tutte le informazioni sui corsi, sulle scuole e sugli insegnati a cui si appoggi l'azienda per fruire i propri servizi; pertanto, deve essere sempre efficiente e disponibile, in quanto un malfunzionamento rischia di compromettere il lavoro dell'operatore.
Data l'importanza del ruolo del database nel lavoro dell'agenzia, è normale che tutti gli operatori debbano poter accedere al database in modo concorrente; per rendere ciò possibile, è stata studiata una topologia multiutente.

In un ambito multiutente, l'applicazione Access deve essere divisa in due parti: una parte contenente le tabelle e tutti i dati, detta **Back-End** e un'altra parte detta **Front-End**, contenente tutte le maschere, le query e i report che vengono eseguiti sul database, che funge da interfaccia per accedere ai dati nel Back-End.
L'applicazione Back-End viene inserita all'interno di una cartella condivisa nel Server aziendale, mentre ogni utente è in possesso di una copia dell'interfaccia, che è in grado di accedere ai dati.

In realtà in azienda non è presente un solo database in grado di gestire tutto il business, ma esistono 3 diversi database, ognuno con una propria interfaccia: è presente un database *Studenti* contenente i dati (anagrafica e profili) dei contatti e la relativa interfaccia *Studenti_I*, un database *DBS* contente le informazioni sui corsi e le scuole usate, con la relativa interfaccia *DBS_I,* e un database *Agenti* contente i dati degli insegnanti di lingua, con l'interfaccia *Agenti_I*. La situazione appena descritta (illustrata in **figura 1.2)**, è il risultato di un evoluzione che il sistema informativo aziendale (non possiamo più parlare di database) ha subito a causa della caratteristica delle applicazioni Access di poter essere sviluppate

giorno per giorno, aggiungendo di volta in volta il componente di cui si ha bisogno. Inutile dire che questa caratteristica comporta anche un rischio: agendo in questo modo si giunge a una crescita irregolare, non strutturata, della propria applicazione, che con il tempo rischia di divenire troppo pesante e inefficiente a causa di ridondanze e/o parti inutilizzate che si accumulano.

figura 1.2 Architettura del database aziendale multiutente



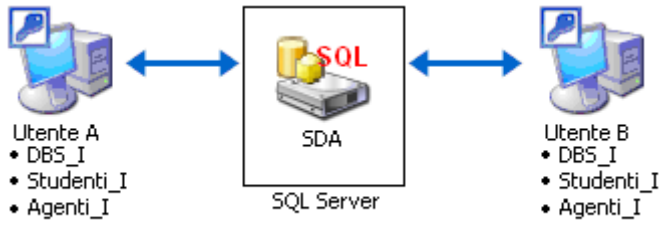
Il problema di un architettura di questo tipo è che non è in grado di gestire grandi quantità di dati: le query in Access vengono eseguite localmente, quindi c'è bisogno che tutti i dati necessari alla query siano inviati attraverso la rete, generando una mole di traffico non indifferente e costringendo l'operatore ad attendere noiosi caricamenti. Uno dei vantaggi di SQL Server consiste nel fatto che le query vengono eseguite direttamente dal server, e vengono inviati nella rete solamente il risultati, limitando il traffico di rete e i tempi di attesa dell'utente.

Se a questo aggiungiamo le dimensioni dei 3 database, che presi singolarmente iniziano a superare la soglia dei 100 Mb, oltre la quale Access non è in grado di garantire prestazioni accettabili, l'esigenza di una maggiore scalabilità e, come vedremo tra breve, la necessità di dover pubblicare il database nel Web, possiamo comprendere quali sono state le motivazioni che hanno spinto l'azienda a

compiere la scelta di migrare i database. Tale scelta, peraltro, veniva coltivata già da tempo. Infatti, un precedente progetto di migrazione da Access 97 a Access 2003 aveva preparato la strada alla migrazione a SQL Server.

Quando si parla di migrazione da Access a SQL Server, si aprono generalmente diverse possibilità. La scelta più comune consiste nel migrare solamente le tabelle e i dati (Back-End) e lasciare in Access tutti i componenti dell'interfaccia (vedi **figura 1.3**); un'altra possibilità, invece, consiste nella completa ristrutturazione dei dati e nella riprogettazione dell'applicazione Front-End. Scegliendo questa strada si abbandona la piattaforma Access in modo definitivo, aprendo la porte a nuove tecnologie più potenti, come ad esempio .NET. Inutile dire che la seconda possibilità, sebbene porti ad un elevato incremento delle prestazioni del sistema informativo, comporta costi elevati sia in termini di tempo sia in termini strettamente economici, mentre la prima soluzione rappresenta un ottimo compromesso tra costi e benefici.

figura 1.3 Architettura del database aziendale con SQL Server



Nel nostro caso è stata scelta proprio la prima strada. Lo schema di **figura 1.3** mostra come è stata ridefinita l'architettura del sistema informativo aziendale: in particolare, si può notare che è stato possibile unire i 3 database in un singolo database SQL Server (in SQL Server infatti non abbiamo problemi di dimensioni), lasciando però inalterate le interfacce.

Tale scelta è stata operata senza escludere, in futuro, di riprogettare da zero

l'applicazione Front-End, creando un interfaccia Web che renda possibile agli operatori di lavorare con il database indipendentemente da dove si trovino.

Precedentemente abbiamo accennato a come l'esigenza di spostare il database sul web sia stato uno dei motivi principali a determinare la scelta di migrazione. Se analizziamo la struttura del sito web (Statico) aziendale è facile comprenderne il motivo.

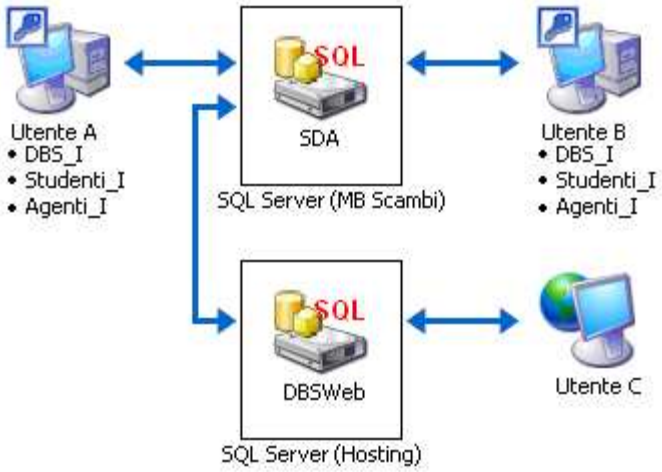
MB scambi culturali è un azienda che fa della sua trasparenza informativa un mezzo per differenziarsi dai concorrenti. Nel sito web, infatti, sono state pubblicate tutte le schede dei corsi offerti. Il problema consiste nel fatto che il numero di questi corsi si aggira intorno a 100, il che significa che esistono altrettante pagine statiche (in formato pdf) caricate nel sito.

Gestire una mole tanto vasta di informazioni risulta essere quantomeno scomodo, e richiede un certo impiego di lavoro. Infatti, ogni volta che un corso viene aggiunto, modificato o cancellato, è necessario riportare le modifiche nelle pagine statiche.

Le schede presenti nel web, però, non sono altro che il risultato dei report sui corsi del database DBS, quindi è stato semplice pensare di poter pubblicare il database DBS, in modo da poter generare le pagine dei corsi in modo dinamico.

MB Scambi culturali, infatti, aveva già iniziato un progetto per rendere dinamico il sito web nella parte relativa ai corsi. Il progetto però fu sospeso quando era ancora incompleto.

figura 1.4 Architettura del database aziendale con SQL Server con accesso dal Web



Lo schema riassume quella che è la struttura del sistema informativo aziendale al termine del progetto di migrazione e di pubblicazione della parte del database aziendale relativa ai corsi (DBSWeb).

Gli operatori si collegano al database aziendale (SDA), memorizzato all'interno del server aziendale, attraverso le interfacce, usando una connessione ODBC.

Il database DBSWeb, salvato su un server remoto(lo stesso che ospita il sito web), non è altro che il risultato dell'esportazione delle tabelle di SDA relative ai corsi, con l'aggiunta di alcune viste, usate dall'applicazione web. In questo modo, un utente interessato ai corsi può visualizzarne le informazioni delle schede generate dinamicamente tramite browser.

STRUMENTI UTILIZZATI

In questo capitolo non vogliamo certo fornire una guida completa agli strumenti utilizzati. A questo scopo infatti esistono interi manuali che approfondiscono l'argomento in ogni sfaccettatura. Noi desideriamo solamente fornire alcuni cenni che possano introdurre tali strumenti al lettore che non ne è a conoscenza. Approfitteremo di questo capitolo, inoltre, per descrivere la creazione dell'ambiente di sviluppo in cui sono state effettuate le prime prove di migrazione, i test per verificare il corretto funzionamento delle interfacce dopo la migrazione e infine l'ambiente di sviluppo per la programmazione del sito web.

MS Access 2003 e SQL Server 2005

MS Access 2003

Microsoft Access 2003 è il primo strumento con cui abbiamo avuto a che fare nel corso del nostro progetto; non si tratta di un semplice motore di database; può essere visto piuttosto come uno strumento di sviluppo che, oltre a consentire la gestione di un database, permette di progettare query, creare maschere e report

(per l'accesso ai dati) e di scrivere macro e script in Visual Basic per rendere automatico il funzionamento dell'applicazione.

Nel capitolo precedente abbiamo già visto come Access, per eseguire le operazioni sui dati, usa di default il suo database-engine (Jet), che risulta però limitato dalla sua architettura File-Share. Uno dei vantaggi di Access, però, consiste nella possibilità di potersi collegare a un database esterno (tramite connessione ODBC); in questo modo, una volta eseguita la migrazione, si può sostituire il Jet Engine con il motore SQL di SQL Server continuando ad usare le maschere, i report, le macro e il codice già progettati in Access.

Access in aggiunta, consente di caricare dati da altri programmi come Excel o Outlook, e dispone di diversi strumenti, che semplificano l'amministrazione del database, come il gestore delle tabelle collegate, la procedura di Compattazione/Ripristino e l'**Upsize Guidato**; quest'ultimo si è rivelato fondamentale per portare a termine il progetto di migrazione, in quanto permette di esportare la struttura del database, oltre che i dati, in un altro DBMS (nel nostro caso SQL Server), in maniera quasi automatica e indolore.

MS SQL Server 2005

Microsoft SQL Server rappresenta una soluzione per la gestione dei dati end-to-end completa e integrata che fornisce una piattaforma più sicura, affidabile e produttiva per i dati aziendali; grazie a un insieme completo di funzioni, all'interoperabilità con i sistemi esistenti (permette di condividere i dati tra più piattaforme, applicazioni e dispositivi per agevolare le connessioni tra sistemi interni ed esterni), fornisce una soluzione completa per i dati delle aziende di tutte le dimensioni, garantendo maggiore affidabilità e scalabilità ai dati aziendali e alle applicazioni analitiche; SQL Server include inoltre un'ampia gamma di funzionalità di protezione altamente precise e configurabili che consentono agli amministratori di implementare difese approfondite e ottimizzate.

SQL Server è integrato da numerosi componenti per la gestione di database

relazionali, per l'analisi dei dati e data mining, per la creazione di report, per la gestione e lo sviluppo del database, per lo sviluppo del sistema di Business Intelligence e molti altri.
Nel nostro progetto non abbiamo visto che una minima parte degli strumenti che caratterizzano SQL Server. Tra questi quello usato principalmente è senza dubbio **SQL Server Management Studio**, un programma (Client) che connettendosi a SQL Server consente all'amministratore di poter gestire il database. Usare SQL Server Management Studio è fondamentale se si vuole accedere al database: una volta avviata l'applicazione, bisognerà specificare il nome del Server SQL al quale si desidera connettersi e l'autenticazione (di Windows se è supportata o di SQL); il vantaggio di questo programma è che tramite connessione TCP/IP ci si può connettere a qualsiasi SQL Server, purché si disponga dell'indirizzo e si abbiano i permessi necessari.

Creazione Ambiente di Sviluppo

Quando si devono compiere operazioni delicate sul database, come nel nostro caso, in cui è richiesto di migrare il database, è opportuno creare un ambiente di verifica in cui eseguire prima alcune prove, in modo da poter preservare i dati da eventuali errori.

Per creare l'ambiente di prova della migrazione è stato sufficiente copiare il database aziendale e l'applicazione Front-End nel computer destinato allo sviluppo e collegare le tabelle dell'interfaccia in modo che si connettano alla copia del database aziendale e non all'originale. Per collegare delle tabelle a una nuova fonte è necessario aprire il gestore delle tabelle collegate (click su strumenti, gestione tabelle collegate), selezionare le tabelle che si intende collegare alla nuova fonte, abilitare l'opzione "Richiedi sempre nuovo percorso", fare click su OK e selezionare nella nuova finestra la copia del nostro database.

Nel prossimo capitolo, vedremo che una volta effettuata la migrazione, sarà necessario controllare che le interfacce Access funzionino correttamente con il database SQL Server.

Per far ciò sarà necessario interrogare il database dalle interfacce, eseguendo delle prove di inserimento, modifica e cancellazione. Per evitare di sporcare il database sarà anche in questo caso necessario costruire un ambiente sicuro, in cui potremo lavorare liberamente. Per prima cosa è necessaria una copia del database SQL Server: possiamo crearne una esportando tutte le tabelle del database in un nuovo database, oppure possiamo usare il database creato in occasiono di una prova di migrazione. L'importante è che siano presenti tutte le tabelle e relazioni; in secondo luogo sarà necessario collegare le tabelle dell'interfaccia al database SQL Server di prova.

Macromedia ColdFusion e Dreamweaver MX

Macromedia Coldfusion MX

Coldfusion MX è un application server sviluppato da Macromedia che consente di generare pagine web dal contenuto dinamico con estrema semplicità; il punto di forza di Coldfusion si basa proprio sulla facilità di apprendimento: in poco tempo si possono imparare le basi del CFML (il linguaggio di Coldfusion) e iniziare a programmare un'applicazione web dinamica, mentre usando altri linguaggi verrebbe richiesto molto più tempo nello studio del linguaggio e nella fase di sviluppo.

Coldfusion non hai limiti nel creare applicazioni dinamiche. Con la nuova versione MX può vantare l'integrazione con Flash, il supporto dell'XML ampliato e la gestione dei Web Service , se a questo aggiungiamo la presenza di numerose e attive comunità virtuali in rete, i costi di sviluppo praticamente nulli (la versione Developer permette di disporre gratuitamente di un ambiente Coldfusion completo sul proprio computer di lavoro) e la completa compatibilità con i sistemi operativi e i database relazionali più diffusi, possiamo affermare che ha veramente raggiunto un livello da far invidia a qualsiasi altro application server disponibile sul mercato.

Macromedia Dreamweaver MX

Dreamweaver è uno strumento utile alla progettazione di applicazioni web, leader nel settore. Esso consente la progettazione visiva delle pagine web, permettendo di manipolare con precisione gli oggetti grafici che compongono la pagina, in modo da accelerare i tempi di sviluppo; consente inoltre un assoluto controllo del codice, permettendo agli sviluppatori di lavorare nel modo più adatto, in base alle proprie esigenze e abitudini. Una delle caratteristiche più importanti di Dreamweaver è il supporto a numerosi linguaggi lato server come ASP, JSP, PHP e ovviamente CFML; sarà possibile, una volta configurato il server di prova, creare componenti dinamici (come ad esempio select box) con estrema semplicità, grazie all’ausilio di procedure guidate.

Oltre che ad aiutare il programmatore nella creazione delle pagine, Dreamweaver permette di organizzare i file in maniera efficiente tramite la creazione di un sito locale, che consente di utilizzare tutte le funzionalità di Dreamweaver garantendo un incremento della produttività.

Creazione Ambiente di Sviluppo

Per iniziare a programmare in Coldfusion è necessario aver installato, oltre Coldfusion (abbiamo usato la versione Developer), un editor HTML (nel nostro

caso è stato usato Dreamweaver), per poter visualizzare e modificare il sorgente delle pagine.
Non è necessario installare alcun Server Web, in quanto Coldfusion dispone un piccolo Server Web appositamente inserito per essere usato in ambiente di sviluppo.

INSTALLAZIONE COLDFUSION

Per installare la versione Developer di Coldfusion, eseguire il file **coldfusion-60-win-en.exe** (può essere reperito gratuitamente nel sito di Adobe), premere Next > e accettare i termini di licenza. Verrà richiesto il nome dell'utente, possiamo compilare il modulo come preferiamo: non serve inserire alcun **Serial Number**, per cui si possono lasciare vuoti gli ultimi due campi. Cliccando ancora su Next > verrà aperta una nuova finestra in cui bisognerà selezionare il web server che si intende usare per testare, in fase di sviluppo, il funzionamento delle pagine. Coldfusion dà la possibilità di utilizzare un web server "proprio", senza aver bisogno di appoggiarsi ad altro software; di conseguenza possiamo selezionare il server Standalone di Coldfusion e ciccare su **Next >**.
Nella finestra successiva vengono richieste due informazioni: la prima riguarda la posizione sul disco rigido in cui vogliamo installare i file di Coldfusion, la seconda dove vogliamo che risiedano i file dei nostri siti dinamici. Installare tutto sul disco C nelle posizioni indicate risulta essere la scelta migliore.

A questo punto vengono richieste le opzioni di installazione (si consiglia di installare sia "Coldfusion MX Application" sia "Documentation and Simple Apps"), mentre successivamente vengono richieste delle password: la prima serve per accedere a Coldfusion Administrator, l'area che permette di configurare Coldfusion Server (quest'opzione è fondamentale per la sicurezza dei server in produzione); la seconda riguarda il Coldfusion RDS (Remote Development Services), che è utilizzato solo dagli sviluppatori che devono collegarsi al server remoto.

A questo punto apparirà una schermata che riassume le impostazioni di installazione: scegliere Install e una volta terminata l'installazione cliccare su Finish e riavviare il computer. All'avvio successivo si aprirà automaticamente una finestra di "congratulazioni", e dietro questa finestra del browser ci sarà la maschera di accesso a Coldfusion Administrator.
Per Poter accedere a Coldfusion Administrator (in un secondo momento) sarà necessario aprire una finestra del browser e inserire il seguente indirizzo: http://localhost:8500/CFIDE/administrator/index.cfm

Notiamo che l'indirizzo è diverso da quelli a cui siamo abituati: "localhost" significa che stiamo visitando il server locale, cioè quello **standalone** installato da Coldfusion. 8500 è il port che il server utilizza per far passare le informazioni, come ci era stato segnalato durante il setup. /CFIDE/administrator/index.cfm è la pagina che dà accesso a Coldfusion Administrator. L'estensione **CFM** significa che è una pagina scritta in CFML (così come HTM indica una pagina in HTML). A questo punto abbiamo la certezza che il nostro server è in grado di interpretare le pagine .cfm che andremo a provare.

Ultimata l'installazione di Coldfusion saremo pronti a programmare in CFML, l'ultima cosa importante da tenere in considerazione è che per visualizzare una pagina .cfm, dovremmo inserirla nella cartella **C:\CFusionMX\wwwroot** e digitare dal browser l'indirizzo: **http://localhost:8500/*nomepagina*.cfm**

ERRORI

Al termine dell'installazione quando si cerca di avviare Coldfusion Administrator può essere visualizzato il seguente messaggio di errore:

Error Occurred While Processing Request
Please Try The Following:
Check the CFML Reference Manual to verify that you are using the correct syntax.
Search the Knowledge Base to find a solution to your problem.
Browser Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Remote Address 127.0.0.1
Referer
Date/Time 10-apr-06 09:40 AM
Stack Trace (click to expand)
java.lang.ExceptionInInitializerError: java.lang.SecurityException: Cannot set up certs for trusted CAs
at javax.crypto.b.([DashoPro-V1.2-120198])
…

Coldfusion MX 6.0 include la versione *1.3.1_03-b03* di *Sun Java Runtime Environment* e la versione 1.2.1 di *Sun Java Cryptography Extensions*. Il certificato usato per la JCE 1.2.1 si è estinto il 28 luglio 2005. Dopo questa data, quando Coldfusion MX 6.0 compila una pagina ColdFusion cifrata verrà visualizzato il messaggio di errore.

Soluzione: Gli utenti di Coldfusion MX 6.0 dovranno installare il seguente aggiornamento: ColdFusion MX 6.1 Upgrade.

Workaround: Se un sistema Coldfusion MX 6.0 non può essere immediatamente aggiornato alla versione 6.1, è possibile aggirare il problema finché l'aggiornamento non può essere effettuato:

- Fermare Coldfusion MX.
- Scaricare cfmx60jce.zip.(191K).
- Estrarre cfmx60jce.zip nella cartella *cf_root/lib,* sovrascrivendo il file esistente.
- Far ripartire Coldfusion MX.

Fonte: http://kb.adobe.com/selfservice/viewContent.do?externalId=98b88caa&sliceId=2

PROCEDIMENTO MIGRAZIONE

In questo capitolo illustreremo tutti i passaggi necessari ad eseguire la migrazione dei database aziendali; per non risultare ridondanti ci limiteremo ad illustrare le fasi per il processo di migrazione di un unico database, sebbene il processo necessiti di essere ripetuto per ben ognuno dei 3 database.
Come già detto nel capitolo precedente, per eseguire la migrazione è stato usato un tool di Access, che prende il nome di **Upsize Guidato** (d'ora in avanti Upsize), che permette la migrazione completa della struttura e dei dati del database. Il processo di Upsize però è esposto a diversi problemi, dovuti alla non perfetta compatibilità tra il Jet Engine (il motore di Access) e quello di SQL Server. In primo luogo, quindi, è stato necessario eseguire un approfondita revisione della struttura del database Access, in modo da renderla compatibile con quella di SQL Server, per evitare che la procedura di Upsize possa fallire.

Durante il processo di migrazione sono stati eseguiti centinaia di controlli. Nel presente capitolo, tralasceremo i dettagli su quali parti sono risultate problematiche (salvo alcuni casi particolarmente interessanti o delicati), indicando semplicemente il tipo di controllo eseguito e perché è stato eseguito.

Cosa Fare Prima della Migrazione

Le differenze tra il Jet Engine di Access e il motore SQL di SQL Server impediscono la perfetta compatibilità tra i due sistemi, la procedura di upsize riesce a correggere automaticamente la maggior parte di queste differenze; una di queste riguarda ad esempio il formato dei campi: Access utilizza dei formati diversi per i campi rispetto a quelli usati normalmente nel linguaggio SQL (ad esempio, un campo *text* viene memorizzato come *memo*).
Sfortunatamente, l'upsize Guidato non è in grado di gestire tutte le problematiche legate alla diversità dei 2 motori di database, quindi prima di avviare la procedura, sarà necessario controllare il database, per eliminare la presenza di alcune incompatibilità nella struttura, che possono causare un fallimento, o un'esecuzione incompleta o errata della procedura di Upsize.

In questo paragrafo vedremo quali sono gli errori che la procedura di upsize guidato non è in grado di correggere e che bisogna quindi controllare e saper gestire manualmente, prima di poter eseguire la migrazione.

Controllo chiavi Primarie

La mancanza di una chiave primaria in una tabella porta a un esecuzione errata della procedura di Upsize; sarà dunque necessario controllare tutte le tabelle, una ad una, e definire una chiave la dove è necessario.
Quando dobbiamo definire una chiave, possiamo prendere 2 strade diverse:

creare un nuovo campo di tipo contatore, che viene assegnato automaticamente a tutti i records già presenti nella tabella, oppure definire come chiave un campo già presente nella tabella. Se si sceglie questa strada, è importante ricordare che un campo chiave non può avere valori *null;* inoltre, non possono esistere due records con lo stesso valore nella chiave; in alternativa, se non si è in grado di trovare alcun attributo che non possegga valori duplicati, si possono prendere più attributi per formare una chiave.

Controllo Relazioni

Le relazioni tra le tabelle possono causare dei problemi: in Access le relazioni possono riferirsi, oltre che ad una chiave primaria, anche ad un attributo irripetibile, cioè con la proprietà *UNIQUE.* Questo non è consentito in SQL Server.
Un altro problema a cui sono soggette le relazioni è che in SQL Server non è possibile definire una relazione tra due campi con un formato diverso. Ad esempio, non possono esserci relazioni tra *varchar* e *nvarchar.* Quindi, oltre che controllare che tutte le relazioni si riferiscano a una chiave primaria, sarà necessario controllare che i campi coinvolti nella relazione abbiano lo stesso formato.
E' inoltre necessario verificare che non vi siano valori, inseriti in una colonna che funge da chiave esterna, che non hanno un corrispondente nella tabella correlata (violano i vincoli di integrità referenziale); per eseguire questo controllo Access mette a disposizione una procedura guidata che crea una query che consente di rilevare tutti i dati non corrispondenti tra 2 campi di 2 tabelle.

Controllo Nomi Tabelle

Anche i nomi dei campi e delle tabelle possono essere causa di un mancato successo nell'esecuzione della procedura di upsize: in SQL Server, i nomi dei campi e delle tabelle non possono contenere alcuni caratteri speciali (vedi **tabella 3.1**), oltre che le parole riservate di SQL, mentre Access concede questa

possibilità; se la procedura di upsize dovesse trovare dei campi contenenti dei caratteri di questo tipo, non sapendo come gestirli, si interromperà provocando così una migrazione incompleta.
La parte critica di questa fase non è tanto la correzione dei nomi, ma riportare tali correzioni in ogni parte dell'applicazione.
Per esempio se modifichiamo il campo *"Arrivata fattura"* in *"ArrivataFattura"*, dovremmo controllare e modificare tutte le macro, le query, i report e le maschere in cui viene usato questo campo, in modo che il nome usato coincida con quello appena modificato.

Tabella 3.1 – Caratteri non consentiti

Apice '	Virgolette "	Percentuale %	Asterisco *	Spazio

Per la lista completa di tutte le parole riservate SQL vedere la tabella all'indirizzo: http://technet.microsoft.com/it-it/library/ms189822.aspx

Finora abbiamo visto come la struttura del database possa portare a dei problemi durante la fase di upsize. Anche le proprietà e il formato dei campi possono essere fonte di errori nella migrazione dei dati.

Controllo Richiesto/Indicizzato

Se un campo ha la proprietà *richiesto* impostata su sì, la procedura di upsize non accetterà valori *null*, esportando solo la struttura della tabella, senza migrare i dati nel caso dovesse incontrarne. Bisogna dunque verificare quali campi (di ogni tabella) hanno la proprietà *richiesto* impostata su sì; successivamente, per ognuno di questi campi bisogna eseguire una query per verificare se sono presenti valori *null*. Nel caso ci si scontri con questo genere di problema si hanno 2 alternative per risolverlo: si può eliminare la proprietà *richiesto* dal campo, oppure si può cercare di assegnare un valore a questi campi.
Un altro tipo di errore si può trovare quando un campo ha la proprietà *richiesto* impostata su *no*, mentre la proprietà *indicizzato* è impostata su *sì(duplicati non*

ammessi) e si hanno almeno 2 records con valore *null* in questo campo: la procedura di upsize esporterà la struttura della tabella ma non i dati.
In questo caso sarà necessario verificare quali campi hanno *richiesto* impostato su *no* e *indicizzato* impostato su *sì(duplicati non ammessi)* ed eseguire una query che ci permetta di verificare se sono presenti più valori *null* per questi campi. Ancora una volta possiamo seguire 2 strade per risolvere il problema: si può impostare la proprietà indicizzato su *sì(duplicati ammessi)*, oppure se il numero di records problematici non è troppo elevato, si può cercare di sostituire i valori *null* del campo assegnando un valore manualmente.

Controllo Valore Predefinito/Valido se

Access consente di definire, nelle proprietà *Valore Predefinito* e *Valido Se,* delle vere e proprie funzioni in Visual Basic: la procedura di upsize proverà a convertire tali funzioni in un equivalente in T-SQL (il linguaggio di programmazione che sta alla base di SQL Server). Nel caso non dovesse riuscire, la procedura di upsize potrebbe non essere completata correttamente. Di norma le funzioni predefinite di Access (come ad esempio Now() per un campo di tipo data) vengono convertite, mentre le funzioni scritte dall'utente possono incontrare dei problemi; in tal caso è necessario eliminare tale funzione prima dell'upsize e ridefinirla in T-SQL una volta completata la migrazione.

Controllo Si/No

In SQL Server, nei campi di tipo bit non è consentito inserire il valore NULL, poiché possono assumere soltanto valore 0 o 1. Se sono presenti dei valori NULL la procedura di upsize migrerà correttamente sia la struttura sia i dati, ma non sarà possibile eseguire alcuna operazione di modifica sui records. Risulta quindi necessario verificare se nei campi di tipo Sì/No (vengono trasformati in bit durante l'upsize) sono presenti dei valori NULL:

```
UPDATE NOMETAB SET campo=No WHERE campo=Null
```

Controllo Data

I campi di tipo data causano sempre problemi: Access infatti non impone alcun vincolo sulle date, SQL Server invece non accetta valori precedenti al 1 Gennaio 1753 o successivi al 31 Dicembre 9999.
Ad un primo impatto si potrebbe pensare che molto difficilmente ci si trova ad aver a che fare con date che escono da questo dominio, invece a causa di errori di battitura non è raro scontrarsi con questo genere di problema.
Sarà dunque necessario controllare, tramite una query, se nei campi di formato Data sono presenti dei valori che sono al di fuori del dominio supportato da SQL server.

Una volta individuati i records errati bisognerà correggerli e ancora una volta potremo scegliere tra due possibilità: assegnare un valore Standard alle date errate (ad esempio 01/01/01) oppure cercare di ricostruire la data esatta, analizzando i valori contenuti nei records “vicini” a quello errato.
Un ulteriore elemento che può causare dei problemi è il formato ora: in SQL Server infatti viene usato un unico campo per *datatime* in cui vengono memorizzate sia la data sia l'ora. Se un campo di tipo ora viene esportato, verrà trasformato in un campo *datatime*, aggiungendo di default la data 01/01/1753; sarà necessario modificare il formato in testo prima dell’esportazione.

Controllo Oggetto OLE

MS Access permette di inserire come campo di una tabella un oggetto OLE. Un oggetto OLE può contenere documenti di MS Word o Excel, immagini suoni e ancora altri tipi di dati. In SQL Server non è prevista questa possibilità, quindi se viene incontrato un capo di questo tipo durante la procedura di upsize, verrà esportata soltanto la struttura delle tabelle, mentre i dati saranno ignorati. Il tipo di risoluzione per questo problema dipende in primo luogo dal tipo di dato contenuto

nel campo.
Supponiamo di avere memorizzate delle immagini: dovremo modificare il nostro campo in un campo di testo (in questo modo viene salvato il percorso del file) e, dopo la migrazione, trasformare il campo in un campo *immagine.*

Avviare la Procedura di Upsize

Una volta eseguiti i controlli appena descritti in ogni campo di ogni tabella del database, e dopo aver apportato le misure necessarie per correggere tutti gli errori di incompatibilità tra Access e SQL Server, il nostro database sarà pronto per essere migrato.
Per prima cosa è opportuno assicurarsi che SQL Server sia installato correttamente e che si dispongano dei permessi necessari per creare e gestire un database. Dopo aver installato SQL Server in quello che sarà il server del database, bisogna seguire passo per passo le seguenti indicazioni:

Aprire il Database Access che si intende Migrare e fare Click su *Strumenti*, *Utilità database*, *Upsize Guidato*; a questo punto si aprirà una finestra in cui verrà chiesto se si intende creare un nuovo database, oppure se si desidera aggiungere le tabelle esportate a un database già esistente. Se non è stato ancora creato un nuovo database in SQL Server, possiamo tranquillamente selezionare la prima opzione, mentre se è la seconda volta che si esegue la procedura di upsize, perché vogliamo esportare le tabelle un po' per volta, dobbiamo scegliere aggiungi

tabelle al database esistente.

Facendo click su *avanti* si aprirà una nuova finestra: questa volta viene richiesto il nome che si intende dare al database e il nome del Server; nel primo campo possiamo inserire il nome che più ci aggrada, mentre nel secondo dobbiamo indicare il nome del computer in cui è stato installato SQL Server o, se non si dispone del nome, l'indirizzo IP. In questa finestra sarà inoltre richiesta l'autenticazione dell'utente. Se si desidera usare l'autenticazione di Windows bisogna abilitare la checkbox *Usa connessione di tipo Trusted*, altrimenti dobbiamo inserire il nome utente e la password di un utente SQL Server che abbia i privilegi di creazione di database.

A questo punto ci troveremo in una finestra che ci consente di scegliere quali tabelle esportare. La quantità delle tabelle che si possono esportare eseguendo la procedura di upsize non è limitata, ma dipende in parte dalla numerosità dei records di ogni tabella, dal numero di colonne e dal tipo dei campi (esportare il contenuto di un campo *memo* richiede molto più tempo di un campo Si/No). Se si dispone di una piccola applicazione si può pensare di esportarla con un unica esecuzione della procedura di upsize, mentre se nel proprio database sono presenti tabelle con migliaia di records e decine di campi, sarà necessario avviare la procedura più volte esportando una ad una le tabelle più grandi. Questo limite è imposto da un Timeout di sistema che termina il processo di upsize quando l'esecuzione eccede il tempo prefissato; se si dispone di tabelle di dimensioni talmente elevate che, anche se prese singolarmente, non riescono ad essere migrate, o se si desidera aggirare il problema, è necessario cambiare le impostazioni di sistema. Per maggiori chiarimenti su come operare riportiamo un link al sito di supporto di Microsoft:

http://support.microsoft.com/kb/295231 (data 12/10/07)

Una volta selezionate le tabelle che si intende esportare, viene aperta un altra finestra nella quale è richiesto di indicare quali proprietà dei campi si intendono esportare. Se si esporta il database in una singola esecuzione della procedura di upsize, si può tranquillamente selezionare sia le relazioni sia le proprietà Indici, Valori Predefiniti e Regole di convalida; nel nostro caso, siamo stati costretti ad eseguire la migrazione del database tabella per tabella, quindi abbiamo deciso di non esportare le relazioni e di crearle successivamente in SQL Server.

Nella la finestra successiva invece vengono proposte 3 opportunità:

1. Creare una nuova applicazione Access client/server di Access.
2. Collegare le tabelle SQL Server all'applicazione esistente.
3. Nessuna (modifica).

Se la propria applicazione è divisa in un database Back-End e un'interfaccia Front-End, come nel nostro caso, non sarà necessario creare una nuova applicazione, perché il database aziendale dispone già di un interfaccia. Se invece tabelle, forms, query e report sono contenuti in un'unica applicazione, allora si può selezionare la seconda opzione. In questo modo l'applicazione originale viene trasformata in un applicazione Front-End, in cui le tabelle non contengono più i dati, ma sono sostanzialmente un collegamento alle tabelle esportate nel database SQL Server.

Se si fa click su avanti e successivamente fine, verrà finalmente avviata la procedura di upsize; al termine di questa verrà generato un report che indicherà quali tabelle sono state esportate, come sono stati convertiti i formati dei campi, quali proprietà sono state definite su questi. Inoltre, in caso dovessero verificarsi, verranno visualizzate le informazioni sugli errori.

Upsize di Tabelle Collegate

Se il proprio database contiene delle tabelle collegate ad Access, l'upsize di queste avviene in modo analogo a quanto indicato sopra. L'unica differenza risiede nel fatto che la procedura deve essere eseguita dal database Access contenete la tabella (non il collegamento), pertanto prima di migrare tali tabelle è necessario eseguire i controlli elencati nel primo paragrafo, in modo da prepararle per la migrazione.

Se invece si dispone di tabelle collegate a un documento Excel, la situazione diventa più complicata. Se si esportano queste tabelle usando la normale procedura di upsize, verrà creata una nuova tabella in SQL Server e non un collegamento, quindi tutte le modifiche apportate al documento Excel non verranno riflesse nei dati contenuti nella tabella.
Per risolvere questo problema, è necessario creare un Linked-Server nel database SQL Server. Un Linked-Server è una struttura che permette l'esecuzione di query (o viste) in oggetti di tipo OLE; in questo modo sarà possibile selezionare tutti i dati contenuti nel documento Excel, tramite appunto una vista, e creare una tabella collegata a quest'ultima nell'interfaccia Access per potervi accedere (vedi **figura 3.1**).
Usando un Linked-Server si ha il vantaggio che le ultime modifiche apportate al documento Excel vengono riflesse automaticamente nella tabella collegata alla vista. L'unico difetto consiste nel fatto che se il documento Excel rimane aperto non è possibile eseguire alcuna query.

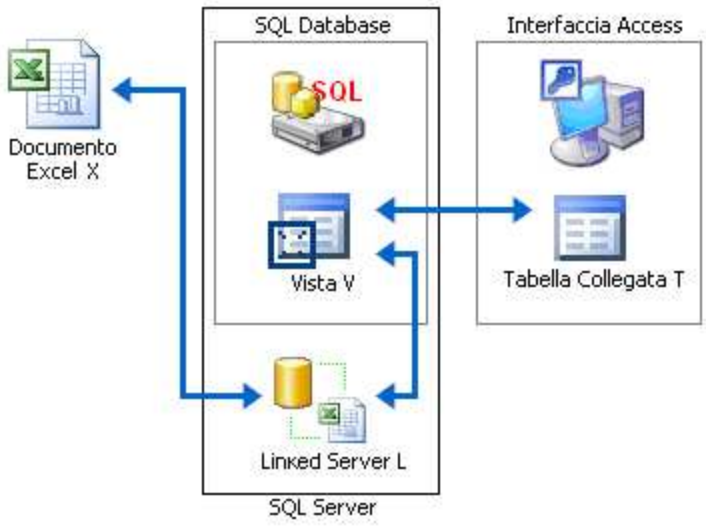
Per creare un Linked-Server ad un documento Excel è necessario aprire una finestra dei comandi da SQL Server Management Studio e inserire il seguente codice:

```
EXEC sp_addlinkedserver NomeLinkedServer,
'Jet 4.0',
'Microsoft.Jet.OLEDB.4.0',
'Percorso\NomeDoc.xls',
NULL,
'Excel 8.0;' GO
```

La sintassi per creare la vista che possa accedere ai dati contenuti nel documento Excel tramite il Linked-Server creato invece è la seguente:

```
SELECT *
FROM NomeLinkedServer...NomeFoglio$
GO
```

Figura 3.1 Schema Funzionamento Linked-Server



Lo schema in figura chiarisce il funzionamento del Linked-Server usato per consentire all'operatore di accedere ai dati contenuti nel foglio Excel (X).Il Linked-Server (L) permette al database SQL di eseguire delle query su X. Pertanto è possibile creare una vista (V) che seleziona tutti i dati presenti in X. Per rendere questi dati accessibili all'operatore bisogna creare, nell'interfaccia Access, una tabella collegata (T) a V; in questo modo, ogniqualvolta viene aperta una tabella collegata, viene eseguita V che restituirà all'operatore i dati contenuti in X, completi delle ultime modifiche apportate al documento.

Creare un Origine Dati ODBC

Quando si intende creare una tabella collegata a un database SQL Server, oppure quando si intende eseguire l'upsize di una tabella in un database SQL Server già esistente, viene richiesto di creare una nuova origine dati ODBC o selezionarne una già esistente.
Un'origine dati ODBC è costituita dalla **Fonte Dati** e dalle informazioni necessarie per accedere a tali **dati** da programmi e database che supportano il protocollo **ODBC** (Open Database Connectivity). Le origini dati ODBC vengono utilizzate per connettersi alle origini dati esterne ad Access che non dispongono di driver incorporati.
Microsoft Access si connette a Gestione driver ODBC che a sua volta utilizza un particolare driver ODBC, nel nostro caso il driver Microsoft SQL ODBC, per connettersi a un'origine dati, ovvero il database SQL Server.

Per creare un origine dati ODBC bisogna aprire la cartella strumenti di amministrazione e cliccare su Origine Dati ODBC. In seguito bisogna selezionare il driver per il quale si desidera creare l'origine dati, nel nostro caso *SQL Native Client* (se i driver per il Native Client non sono presenti è necessario installarli, nel sito di microsoft è disponibile un link per il download gratuito). A questo punto si aprirà una nuova finestra, nella quale bisognerà inserire il nome dell'origine Dati, la descrizione e il nome o l'indirizzo IP del computer che ospita SQL Server. Cliccando su Avanti verrà richiesta l'identificazione dell'utente, che potrà avvenire sfruttando l'identificazione di SQL Server o quella di Windows; successivamente verrà richiesto il nome del database nel quale creare l'origine dati ed infine verrà offerta la possibilità di eseguire un Test per verificare il corretto funzionamento dell'origine dati.

Cosa Fare dopo l'Upsize

Una volta eseguita la procedura di upsize e aver migrato tutte le tabelle, non possiamo ancora considerare l'applicazione completamente migrata: sarà infatti necessario eseguire un controllo sulla qualità della migrazione, per accertarsi che tutte le tabelle e tutti i dati siano stati esportati correttamente. In seguito sarà necessario ridefinire le relazioni e infine collegare le tabelle del database SQL Server all'interfaccia per testare il corretto funzionamento di quest'ultima.

Controllo Qualità Migrazione

Per verificare se la migrazione ha avuto buon fine è necessario controllare che la struttura del database in SQL Server corrisponda a quella del database originale. Sarà necessario un controllo su:

- Chiavi;
- Indici;
- Vincoli;
- Valore Predefinito/Regole convalida;
- Formato Campi.

Una volta terminato il controllo sulla struttura, è necessario verificare la presenza e la correttezza dei dati migrati. Tale controllo non può certo avvenire su ogni singolo record di ogni tabella, pertanto in primo luogo è necessario controllare se il numero dei records di ogni tabella è equivalente a quello del database originale; per un analisi più approfondita si possono selezionare casualmente un numero limitato di records (ad esempio 10) da ogni tabella e controllare se i dati coincidono con quelli del database originale.

Nel nostro caso non sono stati trovati errori durante questa fase, quindi, se tutte le correzioni descritte precedentemente vengono eseguite in modo opportuno, non si dovrebbero incontrare problemi. Se al contrario ci si scontra con qualche difficoltà, è probabile che sia sfuggito un campo ai controlli necessari a preparare il database alla procedura di upsize, quindi bisogna analizzare la tabella o le tabelle problematiche da capo, per poter individuare il problema e poterlo risolvere.

Definire le Relazioni

Come già detto precedentemente, a causa del problema in cui ci si scontra quando si prova a eseguire la procedura di upsize cercando di migrare con un unica esecuzione database di grandi dimensioni, non è stato possibile esportare il database completo delle relazioni. Risulta quindi importante, una volta accertato che la migrazione è riuscita in modo corretto, ridefinire le relazioni nel nuovo database.

Per creare le relazioni in SQL Server esistono due strade: la prima consiste nell'inserire, nell'apposita finestra dei comandi, il codice SQL necessario per modificare la tabella, per poi aggiungere la relazione; in alternativa, ci si può avvalere dell'interfaccia grafica che consente una creazione guidata: aprire la tabella contenente la chiave esterna in modalità modifica (*Modify*), fare click su *Relationship*, *Add*, cliccare su *Tables and Coluns Specification* e selezionare il campo per la chiave esterna, la tabella contenente la chiave primaria e la chiave primaria; in aggiunta si possono specificare altre proprietà delle relazioni, come i comportamenti da attuare in caso di modifica o cancellazione dei records contenenti la chiave primaria.

Collegare il Database all’Interfaccia

Una volta ridefinite le relazioni tra tabelle, possiamo affermare di aver completato con successo la procedura di migrazione; l’operatore, però, per poter accedere al

database ha bisogno di un'applicazione Front-End: dobbiamo quindi collegare il database alla nostra interfaccia Access e accertarci che tutte le funzionalità di questa siano operative.

Per collegare le tabelle del database all'interfaccia, dobbiamo aprire la nostra applicazione Access, cliccando su file, Carica dati Esterni, Collega Tabelle. Si aprirà una finestra che consente di selezionare un file di estensione .msd (l'estensione delle applicazioni Access). Dobbiamo quindi, cliccando su Tipo di File e selezionando Database ODBC, aprire una nuova finestra, in cui ci verrà richiesto di selezionare un'origine dati. Selezionando l'origine dati definita sul database SQL Server e cliccando su OK ci troveremo davanti a una nuova finestra: questa volta dobbiamo semplicemente selezionare le tabelle che desideriamo collegare e premere nuovamente OK.

Testing Interfaccia

Una volta collegate le tabelle all'interfaccia, inizia quella che è la fase più critica del processo di migrazione: la verifica del corretto funzionamento di tutte le funzioni dell'applicazione.

Per completare questa fase non vi è un modo prestabilito di procedere: lo svolgimento dipende infatti da come è stata progettata l'applicazione Front-End; di norma è necessario verificare, innanzitutto, che in tutte le tabelle (salvo alcune eccezioni legate alla struttura del database) siano possibili le operazioni di inserimento, modifica e cancellazione. In seguito è necessario controllare che tutte le query, i report, le maschere e le macro funzionino correttamente.
Un buon approccio è quello di confrontare l'output dell'interfaccia collegata al database SQL Server con quello dell'interfaccia collegata al database originale. Per prima cosa dobbiamo procurarci una copia del Back-End Access e dell'interfaccia, poi dobbiamo collegarle ed infine eseguire in parallelo le

operazioni sulle due applicazioni, per verificare che gli output siano equivalenti.

Questa fase rappresenta un'ottima opportunità per revisionare l'interfaccia Access. Infatti, assieme ai problemi nati a causa della procedura di migrazione, sono stati rilevati molti altri errori e piccole incongruenze che rendevano meno stabile e sicuro il funzionamento dell'applicazione.

ERRORI

Per poter comprendere quali sono i punti che possono rivelarsi problematici in questa fase, di seguito proponiamo una carrellata degli errori più comuni che si possono incontrare.

Fallisce l'inserimento di un record

Non è possibile effettuare un inserimento (non assegna alcun Id).

Soluzione: Potrebbe essere definita su un campo FOREIGN KEY una proprietà *Valore Predefinito* che si scontra con i vincoli di integrità referenziale (ad esempio IdCorso DEFAULT 0, quando nella tabella corsi non esiste alcun records con IdCorso =0). Bisogna eliminare tale proprietà, oppure aggiungere il record corrispondente nella tabella della chiave primaria.

Impossibile aggiornare il campo

Quando si tenta di aggiornare il valore di un campo, viene lanciato un messaggio di errore "*il campo non può essere aggiornato*".

Soluzione:

- Questo è l'errore più comune, è dovuto alla presenza di un origine di controllo errata nel campo; si verifica quando si è modificato il nome di un campo prima della migrazione: l'interfaccia non riesce a riconoscere il nuovo campo e quindi genera l'errore. Per risolvere il problema basta

modificare nell'*origine controllo* del campo il nome del campo: aprire la maschera, fare click con il destro sull'elemento assegnato al campo, selezionare proprietà e modificare il valore in origine controllo.

- Un'altra possibilità è che il campo non sia effettivamente presente nel database, in tal caso è necessario aggiungerlo.

Checkbox disabilitata

Quando si apre una maschera, una checkbox, che non ha la proprietà disabilitata, appare invece disabilitata.

Soluzione: Il problema è analogo a quello precedente: l'origine di controllo della checkbox è errata.

Combobox dinamica vuota

Quando si fa click su una combobox dinamica, per selezionare l'opzione desiderata, non viene visualizzata alcuna opzione.

Soluzione: Anche in questo caso il problema può essere dovuto al cambiamento del nome di un campo prima della procedura di migrazione. Questa volta però non è l'origine controllo ad essere errata, ma l'*origine riga*.

Combobox dinamica errata

Quando si fa click su una combobox dinamica per selezionare l'opzione desiderata, le opzioni non vengono visualizzate correttamente.

Soluzione: L'errore è dovuto al fatto che la colonna associata alla combobox è sbagliata. Ciò accade quando nella tabella contenente i dati da inserire nella combobox è stato aggiunto un campo prima della migrazione (per esempio perché mancava una chiave); bisogna cambiare la colonna associata al campo.

Impossibile aggiornare un records

Non è possibile modificare un record preesistente.

Soluzione: Il problema può essere dovuto al fatto che è stato aggiunto un campo di tipo bit nel database.
Aggiungendo il nuovo campo bit, se non specificato altrimenti, in tutti i records già esistenti viene inserito il valore Null; questo non è possibile in SQL Server, che impedisce le modifiche su questi records. E' necessario modificare tutti i valori Null del campo bit in 0, e inserire come valore di default 0.

Errore nella visualizzazione del Report

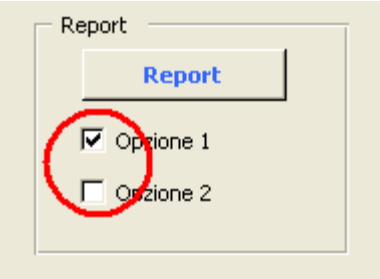
Quando si apre un report viene visualizzata la stringa "# Errore" invece dei dati.

Soluzione:

1. L' origine dati del report risulta errata: bisogna modificare la query di origine dati in modo che selezioni correttamente i dati; problemi di questo tipo, nelle query, sono dovuti ancora una volta alla presenza di un campo rinominato prima della migrazione;
2. L'origine controllo di uno o più campi è sbagliata; ancora una volta il problema è dovuto a una modifica del nome campo prima della migrazione Bisognerà modificare l'origine controllo;
3. Il problema può essere dovuto a un caso più particolare, come illustrato nell'approfondimento.

Una volta terminata la fase di verifica dell'interfaccia possiamo dichiarare chiusa la procedura di migrazione; ora la nostra applicazione gode di maggiore scalabilità, affidabilità e sicurezza e possiamo attivarci per rendere disponibile il nostro database nel web.

figura 2.3 Report con checkbox opzionali



Il report R può funzionare in diverse modalità abilitando le apposite opzioni. Quando si abilitano le checkbox opzionali (vedi figura 2.3) il report non viene visualizzato correttamente. Questo accade perché in SQL Server il campo bit può valere o 0 (falso) o 1 (vero), mentre in Access un oggetto checkbox assume sempre valore 0 quando è falso, ma assume valore -1 quando è vero.

Di conseguenza, quando la query di origine dati per R confronta i valori provenienti dal form con quelli presenti nella tabella, essa non trova nessuna corrispondenza, e quindi il report viene visualizzato in modo errato.

Soluzione: Una possibile soluzione è quella di sostituire le checkbox con delle combobox contenenti i valori 0 o 1; in questo modo il report risponde correttamente, tuttavia questa soluzione risulta scomoda per l'utente.

Una seconda soluzione al problema consiste nel modificare la query in modo che trasformi il valori -1 ricevuti in input in 1, per poter poi eseguire correttamente il confronto. A tal proposito è sufficiente modificare la condizione della query moltiplicando i valori ricevuti in input dal form per se stessi, in questo modo avremo: 0*0=0 e -1*-1=1.

Pubblicazione nel WEB del Database DBS

In questo capitolo descriveremo tutte le operazioni effettuate allo scopo di rendere dinamico il sito Web Aziendale:
Nell'Introduzione abbiamo illustrato lo stato del sito statico di MB Scambi culturali, evidenziando il problema del numero sempre crescente delle pagine statiche relative alle schede dei corsi, e mostrando come la pubblicazione nel Web del database DBS (il database relativo ai corsi) potrebbe rappresentare una valida soluzione a questo inconveniente.
Per poter raggiungere questo scopo, non è stato necessario partire da zero con la programmazione della parte dinamica del sito Web. MB Scambi Culturali, infatti, disponeva già di un applicazione dinamica progettata in Coldfusion. Il problema è che era stata progettata in modo da fruire i dati da un database Access. Inoltre, in alcune parti risultava ancora incompleta.

Lo scopo primario del progetto è stato pertanto quello di “migrare” l’applicazione Web in SQL Server, mentre successivamente sono state avviate le procedure necessarie per migliorare e incrementare la funzionalità. Abbiamo affrontato il problema passando attraverso diversi passaggi chiave:

PRIMA DI COMINCIARE

Riporteremo una descrizione dettagliata della struttura e dei contenuti dell'applicazione, in modo da poter facilitare la comprensione dei paragrafi successivi. Inoltre descriveremo i passaggi necessari per **configurare il sito** in Dreamweaver, in modo da avere la possibilità di usare tutte le funzionalità di Dreamweaver.

SECONDA FASE – Verifica Funzionamento Applicazione

In questa fase sarà necessario verificare che l'applicazione esistente sia perfettamente funzionante. Questa fase gioca un ruolo critico nella procedura, in quanto un errore generato a questo punto sarà difficilmente individuabile nelle fasi successive; non introducendo alcun riferimento al database in SQL Server, possiamo distinguere gli errori che possono nascere in questa fase da quelli che potranno sorgere nella fase successiva.

TERZA FASE – Collegamento a SQL Server

A questo punto si procede modificando il sito web in modo che l'application Server si connetta al database SQL Server, quindi si apportano le modifiche necessarie affinché si possa interrogare correttamente il database SQL Server.

QUARTA FASE – Modifiche e miglioramenti

Verranno proposte le modifiche effettuate all'applicazione per incrementarne la funzionalità.

QUINTA FASE – Approdare nel Web

Verranno proposte le operazioni svolte al fine di caricare l'applicazione nel Web.

ERRORI errori e ancora errori

Sarebbe ingenuo pensare di portare a termine una procedura tanto complessa senza incontrare il minimo errore.

Durante le fasi sopra elencate, si sono verificati una moltitudine di errori, pertanto si è deciso di creare una sezione apposita per ogni fase del procedimento, con lo scopo di catalogare gli errori incontrati e fornire la soluzione.

Prima di Cominciare

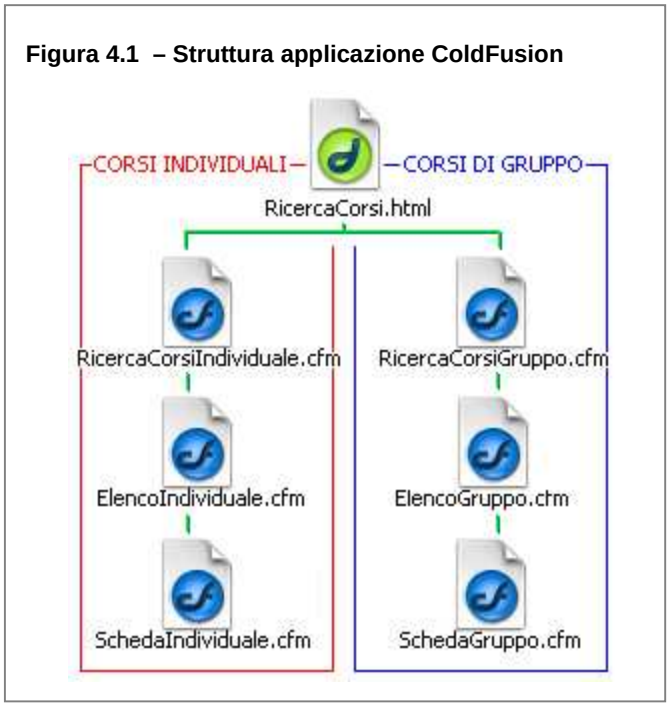
Quando si lavora a una qualsiasi applicazione, progettata da un altro programmatore, non è mai semplice riuscire ad orientarsi: ogni individuo infatti ha un modo proprio di programmare, e studia soluzioni in base alla propria esperienza.
Prima di iniziare le operazioni per riabilitare l'applicazione, è necessario pertanto analizzarla cercando di individuare i diversi componenti e come sono state studiate le soluzioni ai vari problemi.

Struttura dell'Applicazione

Di seguito studieremo la struttura dell'applicazione dinamica. Ciò è servito come punto di riferimento quando abbiamo iniziato a lavorare all'applicazione. Inoltre, rappresenta una chiave di lettura per comprendere meglio le operazioni descritte in seguito.

L'applicazione è divisa in due parti simmetriche (vedi **figura 4.1**), la prima inerente ai corsi individuali, la seconda riguardante i corsi di gruppo:

Figura 4.1 – Struttura applicazione ColdFusion



- Pagina iniziale: offre la possibilità di accedere alle pagine di ricerca per corsi individuali o corsi di gruppo;

- Pagina di ricerca dei corsi: la pagina offre la possibilità di inserire i parametri per la ricerca dei corsi: Lingua, Paese, Città, Tipo Corso;

- Pagina dell'elenco dei corsi: la pagina contenente i Corsi risultati positivi alla ricerca avviata in base ai parametri selezionati nella pagina precedente;

- Pagina della scheda dei corsi: la pagina contenente la scheda del corso selezionato nella pagina precedente.

Nel nostro progetto ci siamo concentrati principalmente nella parte relativa ai corsi individuali, in quanto l'azienda deve ancora inserire, nel proprio database, le informazioni riguardanti l'offerta dei corsi di gruppo.

RicercaCorsi.cfm

La Pagina di ricerca corsi è costituita da un form, in cui compaiono 4 caselle di selezione (Lingua, Paese, Città, Tipo Corso). Il contenuto di ogni casella è dinamico: ogni volta che viene selezionato un valore la pagina viene ricaricata, eliminando le combinazioni che possono dare così esito a una ricerca nulla.

Per determinare questo comportamento, i 4 oggetti di input sono assegnati a 4 variabili sessione: ogni volta che viene selezionato un valore dalle caselle di selezione, le variabili sessione assumono quel valore e la pagina viene ricaricata; ad ogni caricamento vengono eseguite 4 query (una per ogni parametro), che selezionano, dalla tabella relativa ai corsi, la lingua, il paese, la città e il tipo corso sulla base dei valori che assumono le variabili sessione.

Nella pagina è presente infine un pulsante che consente di accedere alla pagina successiva.

ElencoIndividuale.cfm

La pagina è costituita da una query che seleziona i corsi in base al valore assunto dalle variabili sessione definite nella pagina precedente; i corsi selezionati vengono successivamente visualizzati in elenco, mostrando per ognuno le informazioni generali quali il nome, la città, la lingua, il tipo e il paese. Per ogni corso è presente un link alla pagina SchedaIndividuale.cfm, che passerà come parametro alla pagina successiva l'id del corso selezionato.

SchedaIndividuale.cfm

La pagina è costituita da 4 query distinte, che per il corso prescelto selezionano e visualizzano:

1. le informazioni sulla Scuola e la struttura ;
2. le informazioni sul Corso;
3. le informazioni sulle date di inizio del corso;
4. i listini prezzi del corso.

I listini dei prezzi non sono visualizzati immediatamente: esiste una casella di selezione che permette di accedere al listino desiderato in base al numero di

lezioni per settimana. Questa scelta risulta essere scomoda, richiedendo infatti un secondo caricamento della pagina, quando la stessa informazione potrebbe essere restituita immediatamente; più avanti vedremo come è stato risolto il problema.

Configurare il Sito

Prima di cominciare a editare una o più pagine, Dreamweaver richiede di **configurare un Sito:** un sito di Dreamweaver è un modo di organizzare tutti i documenti associati ad un sito Web.
Si può considerare allo stesso modo di un progetto: l'organizzazione dei file in un sito rende possibile l'utilizzo di Dreamweaver con la tecnologia FTP, per caricare il sito sul server Web, verificare e gestire automaticamente i collegamenti, gestire e condividere i file e sfruttare le potenzialità del **Server di Prova**.

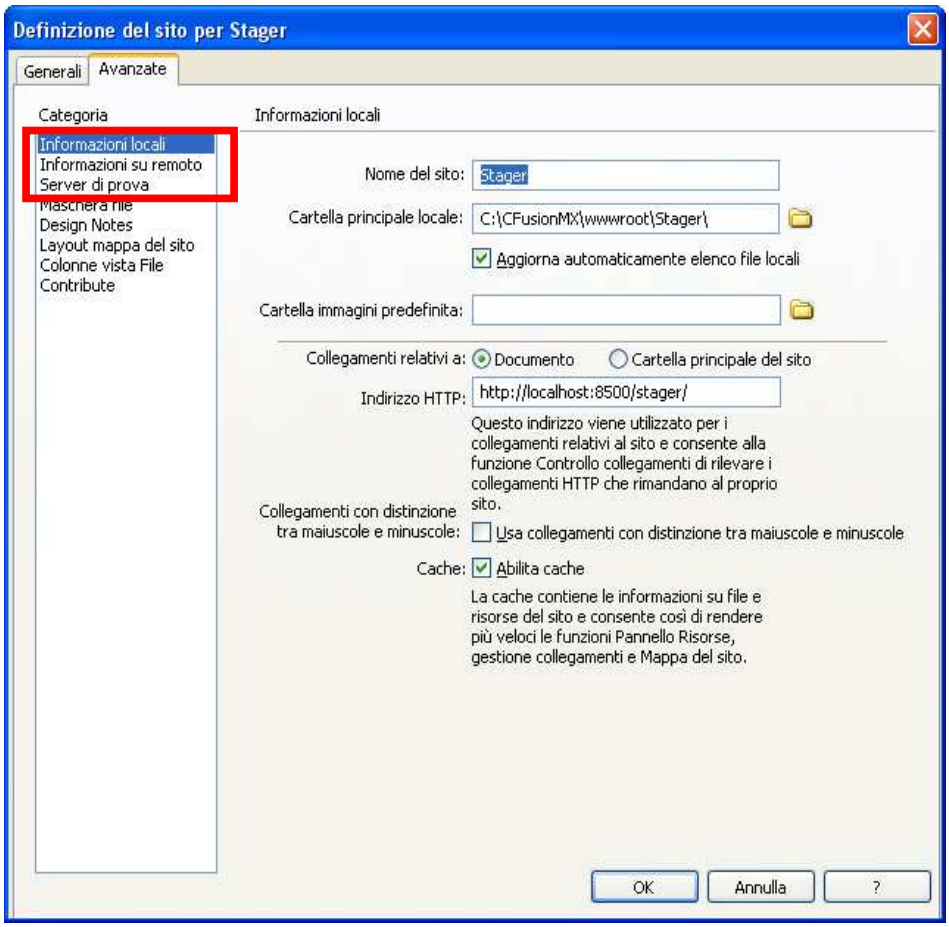
Un server di prova è necessario per generare e visualizzare il contenuto dinamico durante la progettazione. Tramite questo strumento si può dunque testare il funzionamento di ogni componente della pagina, senza dover necessariamente eseguire in continuazione le pagine dinamiche per potersi accertare del funzionamento di ogni componente. Questo rappresenta un enorme vantaggio in fase di sviluppo, e permette di ottimizzare i tempi di progettazione dell'applicazione, evitando al programmatore inutili perdite di tempo.

La configurazione del nostro sito in Dreamweaver richiede di installare Coldfusion application Server (come abbiamo mostrato nel capitolo Strumenti Utilizzati), per stabilire una connessione al database (vedremo successivamente come fare).

Una volta installati Coldfusion e Dreamweaver apriamo la finestra di gestione dei siti, cliccando su Nuovo e selezionando Sito si aprirà una seconda finestra (figura 4.2), in cui verrà richiesto di definire una cartella locale, una cartella remota

(opzionale) e una cartella del server di prova, più altre informazioni opzionali che non vedremo.

Figura 4.2: Definizione del Sito



DEFINIRE LA CARTELLA LOCALE

La cartella locale è la cartella di lavoro in cui si prepara il sito. È necessario definire una cartella locale per ciascun sito Web di nuova creazione; se si usa il web server di Coldfusion administrator è importante che la cartella locale si trovi all'interno della cartella principale di Coldfusion (C:\CFusionMX\wwwroot\). In caso contrario riscontreremo un errore nella visualizzazione delle pagine.

Quando si definisce la cartella locale viene richiesto anche l'indirizzo URL del sito: se la cartella locale è inserita dentro la cartella wwwroot di Coldfusion inserire

http://localhost:8500/nomesito/, altrimenti bisogna specificare il percorso.

DEFINIRE LA CARTELLA REMOTA

La cartella remota corrisponde alla posizione di archiviazione dei file, per la verifica, la produzione, la condivisione. Se si sta usando il web server messo a disposizione da Coldfusion, per il momento non è necessario specificare alcuna cartella remota (la verifica infatti avviene localmente); Quindi, sulla finestra Informazioni su Remoto selezionare Accesso: Nessuno. Altrimenti, bisogna definire la cartella principale del server Web come cartella remota di Dreamweaver.

DEFINIRE LA CARTELLA DEL SERVER DI PROVA

La cartella del Server di prova serve per l'elaborazione delle pagine dinamiche: è necessaria per generare e visualizzare il contenuto dinamico durante la progettazione.

Per configurare la cartella cliccare su Server di Prova e selezionare il modello server desiderato (Coldfusion); una volta inserito il modello server bisogna inserire la modalità di accesso e il percorso della cartella del server di prova: nel nostro caso l'application server è situato sul computer locale, quindi selezioneremo: Locale/Rete e useremo lo stesso percorso specificato per la cartella locale.

A questo punto bisognerà specificare l'URL della cartella del server di prova. Se la cartella del server di prova coincide con quella locale, dobbiamo specificare lo stesso indirizzo usato per la cartella locale.

Terminato l'inserimento delle informazioni su locale, remoto e server di prova, bisognerà copiare i file del sito web all'interno della cartella principale del sito. Ora siamo pronti per programmare la nostra applicazione disponendo di tutte le potenzialità di Dreamweaver.

Seconda Fase - Verifica Funzionamento Applicazione

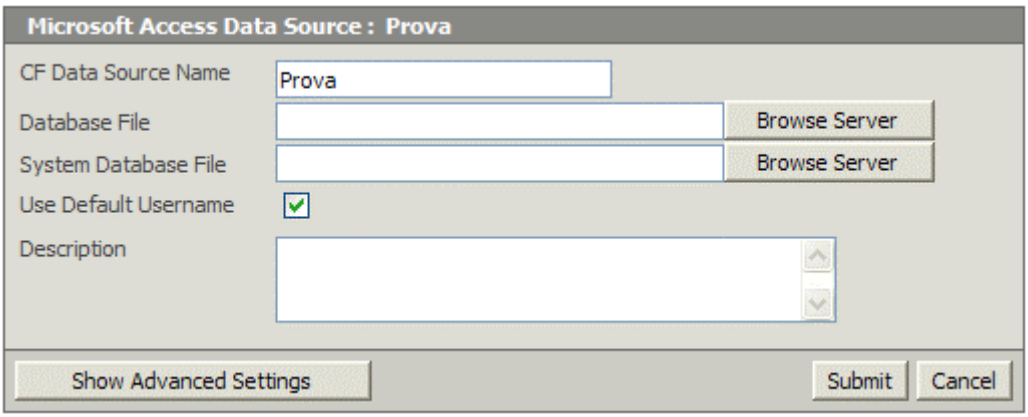
Terminata la definizione del sito web, non siamo ancora pronti per verificare il funzionamento dell'Applicazione Access. Infatti, se si prova ad aprire una delle pagine dinamiche, verrà visualizzato un messaggio di errore che dichiara che la pagina aperta fa riferimento a un'origine dati mancante.
Questo succede perché nelle pagine Coldfusion non sono presenti le informazioni per connettersi al database usato, ma solo il nome dell'origine dati usata; come già detto in precedenza quando abbiamo parlato delle origini dati ODBC, un origine dati è costituita dalla fonte più le informazioni necessarie per accedervi. Questa volta però non useremo il protocollo ODBC per connetterci alla nostra fonte, ma sarà Coldfusion application server a stabilire la connessione.

Per creare un origine dati, dobbiamo aprire Coldfusion Administrator, inserendo in una finestra del browser l'indirizzo:

http://localhost:8500/CFIDE/administrator/index.cfm

Dopo essersi autenticati (con la password definita durante l'installazione di Coldfusion), bisogna cliccare su Data Sources; a questo punto dobbiamo inserire un nome per la propria origine dati e specificare a che tipo di database desideriamo connetterci; una volta definiti entrambi i parametri, facendo Click su *Add*, ci troveremo davanti a una nuova maschera da compilare(vedi **figura 4.3**).

figura 4.3 – Maschera per la creazione di un'origine dati Access



Nel campo Database File bisognerà specificare il percorso del file del database Access, mentre il campo successivo (System database File) è opzionale: serve solo se si desidera creare un contesto di sicurezza. In tal caso è necessario definire il percorso del database contenente le informazioni di sicurezza. Se il campo *Use default Usermame* è selezionato, Coldfusion non usa alcun user name e password per connettersi al database Access. Se nella propria applicazione è richiesta un autenticazione, è necessario specificare Nome Utente e Password nelle impostazioni avanzate. Nel nostro caso useremo l'origine dati creata solo per testare l'applicazione, quindi non sarà necessario impostare campi come *System Database File* o *Description* (la descrizione), o le altre configurazioni avanzate.

Una volta definita l'origine dati, Coldfusion riuscirà a connettersi al database, sfruttando gli appositi driver, come illustrato nello schema in **figura 4.4**, ora siamo finalmente pronti per testare il corretto funzionamento dell'applicazione.

Possiamo dividere la fase di testing dell'applicazione in 2 parti principali:

1. Verifica della corretta visualizzazione delle pagine;
2. Verifica della correttezza delle informazioni visualizzate.

Per quanto riguarda la prima parte è sufficiente verificare se le pagine vengono

aperte correttamente. Può darsi che venga restituito qualche messaggio di errore: nel prossimo paragrafo è riportata una carrellata di quelli che sono gli errori incontrati in questa fase.

Quando tutte le pagine vengono visualizzate correttamente, è necessario controllare se anche le informazioni visualizzate sono corrette: un errore di tipo logico o nella sintassi del codice CFML, infatti, potrebbe dare origine a una query "sbagliata". In questo modo le informazioni visualizzate rischiano di essere errate o incomplete; è necessario dunque verificare che le informazioni visualizzate dall'applicazione web coincidano con quelle presenti nel database.

figura 4.4 – funzionamento Applicazione Web con Coldfusion

Lo schema illustra il funzionamento della nostra applicazione:

1. il Browser Web richiede una pagina dinamica;
2. il server Web individua la pagina e la trasmette a Coldfusion application Server;
3. Coldfusion application server cerca le istruzioni nella pagina;
4. Coldfusion application Server invia la query ai driver del database attraverso le informazioni contenute nell'origine dati;
5. i driver eseguono la query nel database;
6. il database restituisce al driver un recordset contenente le informazioni richieste dalla query;
7. i driver trasmettono il recordset a Coldfusion application server;
8. Coldfusion application server costruisce la pagina inserendo i dati appena ricevuti, quindi invia la pagina al Server Web;
9. il Server Web invia la pagina (contenete solo codice html) al browser Web.

Errori

Per poter comprendere quali sono i punti che possono rivelarsi problematici in questa fase, di seguito proponiamo una carrellata degli errori più comuni che si possono incontrare.

Impossibile Aprire pagina

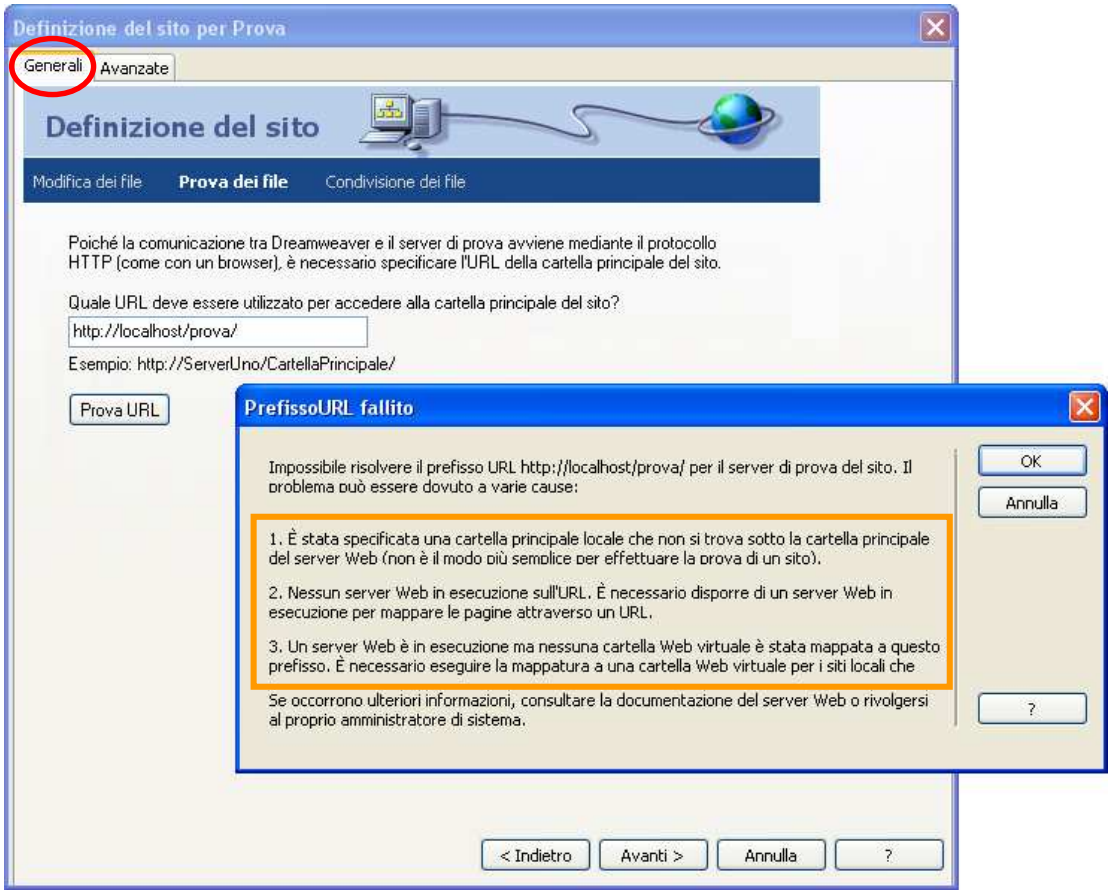
Quando si prova a visualizzare l'anteprima di una pagina .cfm, appare una finestra con un messaggio che chiede di inserire un nuovo server di prova. Una volta specificato il server di prova, se si tenta di visualizzare l'anteprima della pagina si apre una seconda finestra, che chiede se aprire o salvare il file; se si tenta di aprire il file, viene visualizzato solamente il codice. Il problema può essere dovuto al fatto che l'indirizzo URL della cartella principale del sito non risponde correttamente, a causa di un errore nella procedura di configurazione del sito.

Suggerimento:

Quando si esegue la procedura di configurazione del sito, può tornare utile verificare se l'indirizzo URL della cartella principale del sito risponde correttamente.

Per eseguire una prova cliccare su Gestione Siti, selezionare il sito e cliccare su Modifica e in seguito su Generali; avanzare fino alla pagina prova dei file e cliccare su Prova URL; se la connessione fallisce verrà visualizzato un messaggio che suggerisce quali possono essere le cause del fallimento (**figura 4.4**):

figura 4.4: prova URL



Soluzione:

Dalla figura si può notare che esistono 3 motivi diversi che possono generare l'errore:

1. Se, come nel nostro caso, si lavora in locale, è probabile che il problema risieda nel fatto che è stata specificata una cartella principale del sito che non si trova sotto la cartella principale del server web; la cartella principale del server web di Coldfusion si trova al seguente indirizzo: *C:\ColdFusion MX\wwwroot\;*

2. Se la cartella locale è definita correttamente, allora il problema può riscontrarsi nel fatto che il server web non è in esecuzione: bisogna quindi avviare il processo;

3. Se non si lavora in locale, il problema può risiedere nel fatto che la cartella locale del sito non è stata mappata nella cartella principale del web Server. In Coldfusion administrator è possibile eseguire la mappatura di una cartella selezionando la voce mapping dal menu principale.

Impossibile visualizzare pagina

Quando si prova ad aprire una pagina dinamica, se Coldfusion application server non è in grado di interpretare il codice, restituirà un messaggio di errore inserendo una breve descrizione del problema incontrato. Questo tipo di errore è tra i più comuni che si possono ottenere quando si lavora a un sito web; le cause possono essere svariate. Di seguito vengono indicate quelle da noi incontrate.

Data Source mancante

La pagina fa riferimento a un'origine dati inesistente nel file jrun-resources.xml.
In questo caso vi è stato un errore di digitazione del nome dell'origine dati ColdFusion durante la stesura della pagina, oppure l'origine dati è mancante e bisogna aggiungerla.

Soluzione:
Per aggiungere un origine dati aprire ColdFusion Administrator (aprire la finestra di un browser e inserire il questo indirizzo URL: **http://localhost:8500/CFIDE/administrator/index.cfm**), inserire la password di amministratore, aprire il collegamento Data Sources e inserire una nuova origine dati specificando il nome, il tipo del file e il percorso. Una volta creata l'origine dati, si consiglia di testarla con l'apposito comando: se il test va a buon fine l'errore nella visualizzazione della pagina non dovrebbe più presentarsi.

Parametri Insufficienti

Quando si cerca di aprire una pagina dinamica, il browser restituisce il seguente messaggio di errore:

```
[MERANT][SequeLink JDBC Driver][ODBC Socket][Microsoft][Driver ODBC Microsoft Access] Parametri insufficienti: Previsto 1.
...
```

Questo errore, in genere, si verifica quando selezioniamo dei campi che non sono realmente presenti nella tabella o nella query a cui facciamo riferimento, oppure che sono presenti ma hanno un nominativo diverso.

Soluzione:

Per risolvere il problema è necessario controllare che i campi a cui si riferisce la query siano effettivamente presenti nella tabella; in caso contrario bisogna controllare se coincidono con altri campi ma hanno un nominativo diverso: in questo caso è sufficiente aggiungere al campo della query l'operatore AS, più il nome usato nell'applicazione Coldfusion per identificare il campo.

Tabella o Query non trovata

La pagina fa riferimento a un oggetto (tabella query) che non esiste nel database di origine. In questo caso vi è stato un errore di digitazione durante la stesura della pagina, oppure l'oggetto è mancante e bisogna aggiungerlo.

Durante questa fase non è stato raro scontrarsi con questo problema, perché la copia del database Access usato per sviluppare l'applicazione è andata perduta prima che si potessero salvare tutte le query usate dalle pagine dinamiche nel database principale.

Soluzione:

Per risolvere il problema bisogna controllore la lista di tabelle del database e assicurarsi che l'oggetto a cui si fa riferimento esiste con un altro nome; è possibile infatti che nella fase di preparazione del database, prima della migrazione, alcuni nomi tabella siano stati cambiati. In questo caso sarà sufficiente modificare il nome dell'oggetto nel sorgente della pagina dinamica. Se l'oggetto

invece non viene individuato tra quelli esistenti, è necessario crearlo. Per poter definire la struttura che deve avere tale oggetto, bisogna studiare l'applicazione e comprendere in che ambito viene usato.

Ad esempio la pagina RicercaCorsi.cfm deve eseguire una selezione sulla query *LinguaPaese;* sfortunatamente, questa risulta essere mancante nel database Access. Osservando la nostra pagina, però, possiamo notare che la query è usata per trovare la corrispondenza tra: Lingua, Tipo corso (definite nella tabella dei corsi) e Paese e Città (definite nella tabella fornitori esteri); siamo dunque in grado di ricreare la seguente query:

Query: LinguaPaese

```
SELECT TCorsi.Lingua, TCorsi.TipoCorso, FornitoriEsteri.Paese,
       FornitoriEsteri.Citta
FROM   FornitoriEsteri INNER JOIN TCorsi
ON     FornitoriEsteri.IdScuola = TCorsi.IdScuola;
```

Errori nella Scrittura del Codice

Quando la pagina viene visualizzata correttamente, ma ha un comportamento anomalo, non previsto, è probabile che vi sia un errore nella stesura del codice.
Errori di questo tipo dipendono fondamentalmente da 2 fattori:

1. la sintassi del codice: potrebbe essere presente un errore di battitura del codice (ad esempio una parentesi messa al posto sbagliato);
2. la logica dell'algoritmo: l'algoritmo pensato per risolvere il problema presenta una falla e va sistemato.

Quando ci si trova davanti a questi errori è necessario quindi prima un controllo del codice; nel caso l'errore non risieda nel codice è necessario controllare l'algoritmo.

Terza Fase – Collegamento a SQL Server

Terminata la fase di restauro dell'applicazione Coldfusion/Access, possiamo iniziare le procedure che permetteranno al nostro sito web di visualizzare le informazioni ricevute dal database definito in SQL Server.

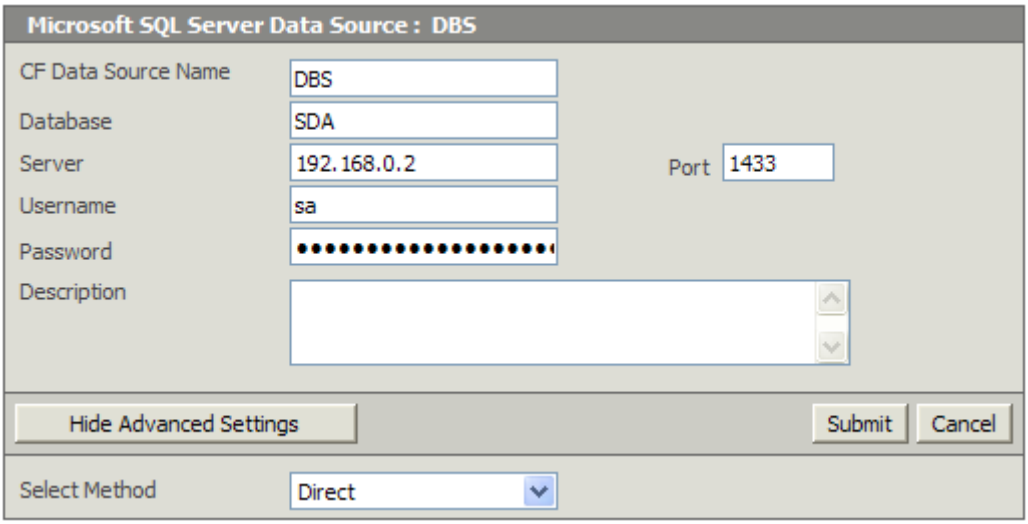
Per procedere in questa fase è necessario innanzitutto preparare il database ad essere interrogato dalla nostra applicazione web; se si crea un origine dati che punta al database SQL server, e si provano le pagine, queste non verranno visualizzate, in quanto nel database SQL server non sono ancora definite tutte le query usate dalle pagine dinamiche del sito per ottenere i dati. Di conseguenza, il primo passo consisterà nel riprogettare le query in SQL Server.

Una volta ridefinite tutte le query, si potrà procedere con il creare un'origine dati Coldfusion, questa volta però dovremo permettere al nostro application Server di connettersi al database SQL Server e non a quello Access.

Per creare la nuova origine dati, dobbiamo ancora una volta connetterci a Coldfusion Administrator, selezionare Data Source e indicare il nome e il tipo di database che intendiamo usare (per evitare di dover modificare il nome dell'origine dati in ogni pagina dell'applicazione web, si consiglia di inserire lo stesso nome usato per la Data Source Access, mentre nel tipo di database selezioneremo ovviamente SQL Server).

A questo punto dovrebbe apparire una maschera come quella mostrata in **figura 4.5**, che dovrà essere compilata come descritto:

figura 4.5 – Maschera per la creazione di un'origine dati SQL Server



Database: Il nome del database in SQL Server.

Server: Il nome, o l'indirizzo IP del Server che ospita il database.

Port: Il port usato da SQL Server. Di default viene usato il 1433.

Username: Il nome di un utente SQL Server (coldfusion non accetta l'autenticazione di Windows, quindi se il proprio database non è impostato per accettare autenticazioni di SQL è necessario abilitare tale opzione).

Password: La password relativa all'utente SQL inserito.

Description: La descrizione dell'origine dati. Questo campo è opzionale.

Method: Il metodo che ColdFusion Application Server usa per connettersi al database. Nel nostro caso è necessario selezionare *Direct*, perché ColdFusion non supporta la connessione *Cursor* con SQL Server.

Terminata la compilazione della maschera, Coldfusion Administrator ci offre la

possibilità di effettuare un test per verificare il corretto funzionamento dell'origine dati creata. Se il test va a buon fine siamo pronti per eseguire la fase di testing dell'applicazione, altrimenti dobbiamo controllare la correttezza dei dati inseriti per rintracciare l'errore (alla fine del paragrafo riportiamo come sempre la documentazione sui possibili errori che si possono incontrare in questa fase).

Dopo aver creato l'origine dati sul database SQL Server e averne verificato il corretto funzionamento, è necessario provare a visualizzare una ad una le pagine .cfm della nostra applicazione, in modo da poter verificare il corretto funzionamento di questa; potrebbero infatti essere presenti alcuni problemi nelle query dovuti alle incompatibilità tra la sintassi di Access e quella di SQL Server o, come vedremo nel prossimo paragrafo, altri tipi di errore.

Errori

Per poter comprendere quali sono i punti che possono rivelarsi problematici in questa fase, di seguito proponiamo una carrellata degli errori più comuni che si possono incontrare.

Impossibile creare Data Source

Quando viene eseguito il test per verificare il corretto funzionamento dell'origine dati coldfusion, viene visualizzato il seguente messaggio di errore:

```
Login failed for user ''. The user is not associated with a trusted SQL Server connection.
...
```

Soluzione:

Il problema può essere dovuto ad un errore nell'inserimento dei dati dell'utente, necessari a creare l'origine dati: per esempio, è possibile che siano state inserite le credenziali di un utente non definito in SQL Server. In tal caso è necessario crearne uno:

- aprire SQL Server Management Studio;
- connettersi al database engine;
- aprire il database;
- cliccare su sicurezza, utenti, nuovo utente e compilare i campi della maschera.

Se i dati inseriti nella maschera sono corretti, allora il problema risiede nelle impostazioni di SQL Server, che non consentono di accettare autenticazioni al di fuori di quelle di Windows:

Coldfusion si connette a SQL Server usando l'autenticazione di SQL Server. Questo però può essere impostato in modo da usare esclusivamente l'autenticazione di Windows. Per poter permettere a Coldfusion di connettersi a SQL Server è dunque necessario abilitare l'opzione "usa autenticazione di SQL Server e Windows". Ecco come si fa:

1. aprire SQL Server Management Studio;
2. connettersi al database engine;
3. aprire la finestra delle proprietà e cliccare su sicurezza;
4. abilitare l'apposita checkbox;

Successivamente è necessario creare un nuovo login che sfrutti l'autenticazione SQL Server e che abbia perlomeno i permessi di lettura dei dati.
Una volta completati questi passaggi, è sufficiente inserire lo Username e la Password dell'utente appena creato nelle apposite caselle nella maschera di creazione di un origine dati in coldfusion Administrator.

Flusso del protocollo RPC TDS in entrata non corretto

Conclusa la creazione dell'origine dati del database SQL Server, se si prova a visualizzare una pagina contenente una query, viene visualizzato il seguente errore:

[Macromedia][SQLServer JDBC Driver][SQLServer]Il flusso del protocollo RPC (Remote Procedure Call) TDS (Tabular Data Stream) in entrata non è corretto. Parametro 1 (""): tipo di dati 0x38 sconosciuto.
...

Soluzione:

Il problema risiede nel fatto che Coldfusion MX non supporta la connessione *Cursor* con SQL Server. Per risolvere il problema è sufficiente cambiare il metodo di connessione da *Cursor* a *Direct*, nel pannello di amministrazione delle origini dati (bisogna mostrare le configurazioni avanzate).

Non viene visualizzato il contenuto delle combobox

Quando si apre una pagina contenete delle combobox dinamiche, non viene visualizzato il contenuto di alcune di esse.

Soluzione:

Il problema risiede in un errore nelle query di origine dati delle combobox: le query affette da questo problema seguono la sintassi di Access e presentano la condizione "<> Null". In SQL Server tale condizione deve essere espressa con la sintassi "is not Null"; è dunque necessario modificare tutte le query in cui è presente tale condizione, come mostrato in **figura 4.6.**

figura 4.6: modificare sintassi <> null

Nome Oggetto non valido

Quando si prova ad aprire una pagina viene visualizzato il seguente messaggio di errore:

[Macromedia][SQLServer JDBC Driver][SQLServer]Il nome di oggetto 'OGGETTO' non è valido.

Soluzione:

Il problema è dovuto a un errore di scrittura: il nome dell'OGGETTO è stato digitato in maniera non corretta, pertanto per risolvere il problema è sufficiente correggere il nome.

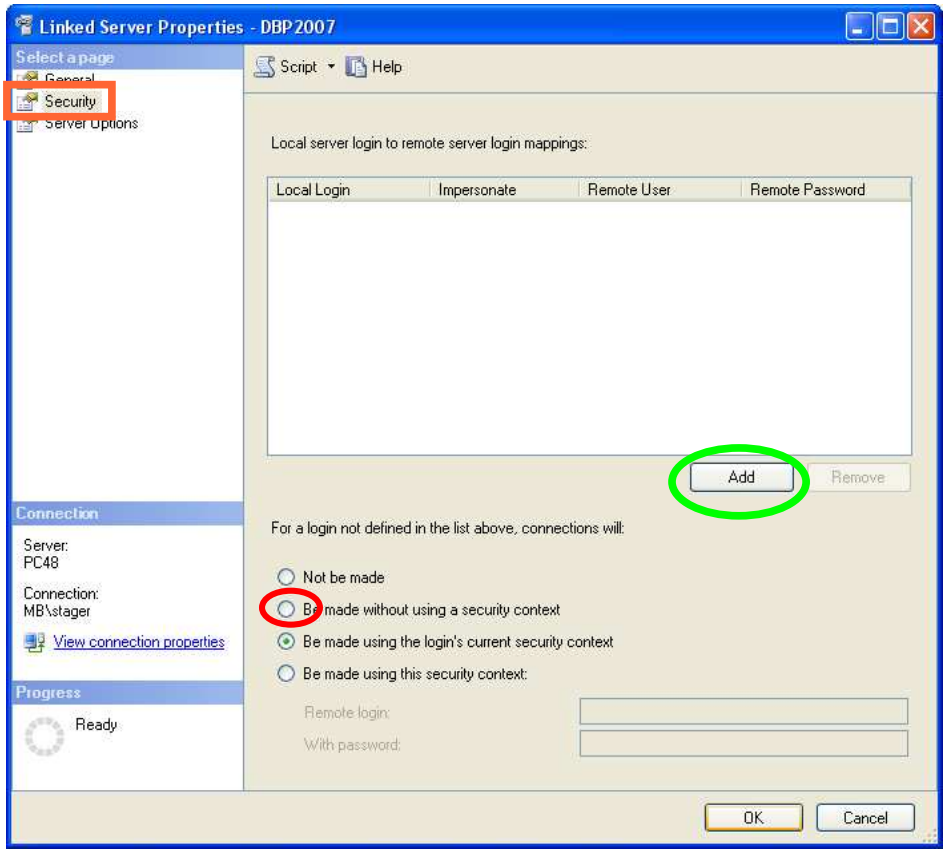
Autenticazione non riuscita

Quando si prova ad aprire una pagina che esegue un'interrogazione a un Linked Server, viene generato un errore di autenticazione dal Server Collegato. L'errore è dovuto al fatto che l'utente SQL Server usato da Coldfusion per accedere ai dati non ha i permessi necessari per eseguire un'interrogazione al Linked Server.

Soluzione:

Bisogna modificare le proprietà dei server collegati, in modo che consentano l'accesso all'utente Coldfusion: Aprire SQL Server Management Studio, selezionare proprietà del server collegato e la finestra relativa alla sicurezza (**figura 4.7**). A questo punto, se siamo in un ottica locale, in cui non è necessario definire particolari livelli di sicurezza, possiamo tranquillamente abilitare l'opzione "be made without using a security context"; altrimenti dobbiamo cliccare su *Add* e aggiungere all'utente usato da Coldfusion i permessi necessari per collegarsi al Linked Server.

Figura 4.7: cambiare livello sicurezza



Impossibile eseguire Query

Quando si prova ad aprire una pagina, viene visualizzato il seguente errore:

> [Macromedia][SQLServer JDBC Driver][SQLServer]Impossibile selezionare come DISTINCT il tipo di dati ntext perché non è confrontabile.

Soluzione:

Il problema risiede nel fatto che in SQL Server non è possibile confrontare 2 campi di tipo *ntext* (mentre in Access sì), quindi non è possibile inserire l'operatore DISTINCT in una query che seleziona un campo di questo tipo. Per risolvere l'inconveniente è necessario eliminare l'operatore DISTINCT dalla query. Questo, però, può rappresentare un problema, perché i dati selezionati dalla nostra query verranno ripetuti. Per questo è necessario studiare un metodo per aggirare il problema.

Quarta Fase – Modifiche e Miglioramenti

In questo paragrafo entreremo un po' più nello specifico, illustrando le modifiche apportate alla nostra applicazione, cercando di migliorare da una parte la dinamicità del sito, in modo che il gestore debba intervenire nelle pagine il meno possibile, dall'altra l'esperienza di navigazione del visitatore, cercando di ridurre al minimo il numero dei caricamenti delle varie pagine e aumentando i parametri di ricerca dei corsi, rendendo possibile all'utente l'accesso alle informazioni desiderate nel minor numero di click possibile.

Modifica automatica del listino prezzi

Nella pagina relativa alle schede dei corsi, per alcuni corsi sono disponibili i listini dei prezzi, contenuti nelle tabelle DBP*anno*, dove *anno* indica a che anno fanno riferimento i prezzi; nel momento in cui si volesse modificare il listino dei prezzi (per esempio passare da quello del 2007 a quello del 2008), nella pagina bisognerebbe modificare i nomi di tutte le query che chiamano le tabelle relative ai prezzi dell'anno.

Questa operazione risulta essere molto delicata e richiede la dovuta attenzione, in quanto un errore rischierebbe di sporcare la visualizzazione di dati molto importanti. Per rendere il più semplice possibile (anche se non automatico), il passaggio da un listino prezzi a un altro possiamo sfruttare le potenzialità di

Coldfusion: permettendo l'aggiornamento di tutti i nomi delle query (che chiamano le tabelle relative ai prezzi) semplicemente modificando il valore di una variabile.

Per prima cosa è necessario definire una variabile che dovrà essere modificata in caso si desideri passare al listino prezzi dell'anno nuovo; la **figura 4.8**, mostra la definizione della variabile e i commenti inseriti per spiegare all'amministratore del sito web come modificarla per ottenere il risultato desiderato .

Figura 4.8: creare variabile e inserire i commenti.

```
<!---              LISTINO PREZZI ANNO                          --->
<!--- Per visualizzare i corsi con il listino prezzi dell'anno --->
<!--- desiderato è sufficiente modificare il valore a cui è    --->
<!--- inizializzata la variabile anno definita qui sotto.      --->

<cfset anno=2007>

<!--- Modificando il valore della variabile anno si è deciso   --->
<!--- di visualizzare i corsi con il listino prezzi            --->
<!--- dellìanno selezionato                                    --->
```

Dopo aver creato la variabile anno è necessario modificare il nome delle tabelle (o viste) relative ai prezzi, come mostrato in **figura 9.4**:

figura 4.9: modificare nomi query.

```
<cfquery name="DBP2005" datasource="DBS">
SELECT  *
FROM SchedaDBP#anno#
WHERE IdCorso = #session.IdC#
<cfif isDefined("form.ComboLez")>AND CodiceCorso = '#form.ComboLez#'</cfif>
order by nSettimane ASC
</cfquery>
```

Se nella variabile anno abbiamo inserito, per esempio, 2007, verrà eseguita la query di selezione dalla tabella **SchedaDBP2007**, ottenendo così il listino prezzi dell'anno 2007.

La soluzione appena proposta rappresenta certamente un grosso passo avanti,

ma è esposta a un piccolo inconveniente: quando si decide di passare al listino prezzi dell'anno successivo, la tabella contenente i prezzi dovrà essere completa dei listini prezzi di tutti i corsi. Quindi, per iniziare a visualizzare i prezzi del anno nuovo, dovremmo attendere che sia completata la fase di stesura del listino per ogni corso.
Il problema consiste nel fatto che, nel nostro caso, sono presenti centinaia di corsi nel database aziendale, quindi l'aggiornamento dei listini può richiede alcuni mesi. Vogliamo dunque trovare un modo per visualizzare, per i corsi dove sono presenti i nuovi listini, i prezzi dell'anno nuovo, mentre quelli dell'anno corrente per i corsi non ancora aggiornati.

La soluzione proposta prevede di usare il campo *DataUltimaRevisione* (che viene aggiornato quando vengono aggiornati i listini dei prezzi), presente nella tabella TCorsi. Per assegnare dinamicamente il valore alla variabile *anno,* è necessario aggiungere una query che selezioni l'anno dal campo DataUltimaRevisione. Una volta ottenuto il valore dell'anno, è sufficiente assegnare tale valore alla variabile anno: in questo modo la query di selezione dei listini prezzi selezionerà i prezzi dalle tabelle relative all'anno in cui sono stati aggiornati i corsi.

Visualizzazione Completa Dei Listini Prezzi

Sebbene il sito risulti perfettamente funzionante, si è deciso di modificare la parte riguardante la visualizzazione dei prezzi. Nella versione precedente esisteva una casella di selezione che permetteva di selezionare il listino dei prezzi in base al numero di lezioni per settimana. Questa scelta però risulta discutibile per 2 motivi:

1. Per visualizzare il listino prezzi desiderato, l'utente deve attendere il caricamento della pagina. Nel caso volesse visualizzare il listino successivo, dovrà attendere un ulteriore caricamento;
2. Per alcuni corsi esistono più listini prezzi con lo stesso numero di lezioni per settimana (in questo caso i prezzi dipendono dalla stagione), e non

è possibile associare un campo che permetta di identificare tale differenza (l'unico sarebbe il codice corso, ma essendo un codice è di difficile interpretazione per l'utente).

Pertanto è stato deciso di modificare la pagina della scheda del corso, in modo che vengano visualizzati fin da subito i prezzi.

In primo luogo è necessario definire una query che selezioni tutti i listini prezzi per ogni corso. Per distinguere i listini prezzi in base alla stagione o al numero di settimane è sufficiente eseguire un controllo su CodiceCorso; in fase di visualizzazione, quando il CodiceCorso cambia rispetto al record precedente, iniziamo ad inserire i dati all'interno di una nuova tabella in modo da poter distinguere i diversi listini prezzi.

Query: QListinoPrezzi

```
<cfquery name="QListinoPrezzi" datasource="DBS">
SELECT nLezSett, CodiceCorso,Famiglia,Residence,IDCorso,nSettimane,nLezSett
FROM SchedaDBP#anno#
GROUP BY nLezSett, CodiceCorso,Famiglia,Residence,IDCorso,nSettimane,nLezSett
HAVING IdCorso = #URL.IdC#
</cfquery>
```

I listini dei prezzi vengono associati alle date di inizio dei corsi. Per la visualizzazione delle date è necessario creare una seconda query; anche in questo caso per distinguere le date in base alla stagione o al numero di lezioni è sufficiente un controllo sul CodiceCorso.

Query: QDateLezSett

```
<cfquery name="QDateLezSett" datasource="DBS">
SELECT  CodiceCorso,IDCorso,DataInizioCorso
FROM SchedaDBP#anno#
GROUP BY  CodiceCorso,IDCorso,DataInizioCorso
HAVING IdCorso = #URL.IdC# </cfquery>
```

Il problema a questo punto risiede nel fatto che non è possibile annidare due tag

<cfoutput>, quindi risulta impossibile visualizzare le date di inizio dei corsi nell'intestazione di ogni listino. Per risolvere il problema è stato dunque necessario eseguire, prima, la query QdateLezSett, e salvare le date di inizio del corso in un array, in modo che ogni elemento dell'array contenesse la lista di date per ogni listino prezzi da visualizzare.

Alcuni dei corsi presenti nel database non hanno specificata alcuna data di inizio, in quanto questi corsi iniziano ogni lunedì. Abbiamo dunque creato una query che controllasse se, per il corso selezionato, esistono effettivamente le date di inizio dei corsi. In caso contrario, al posto delle date, viene visualizzata la Stringa "I Corsi iniziano ogni Lunedì".

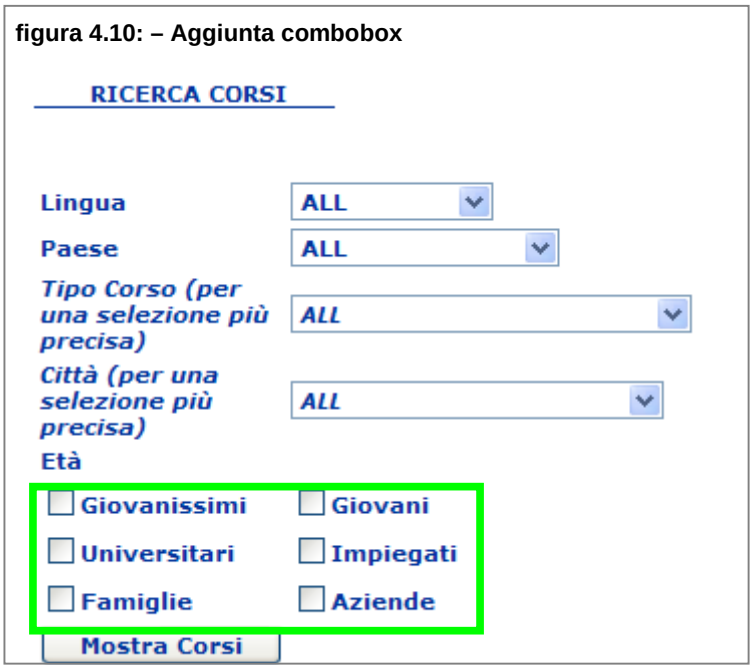
Selezione Corsi Tramite Categoria Utente

Per agevolare il visitatore nella ricerca dei corsi più adatti alle proprie esigenze, si vuole introdurre un nuovo parametro per la selezione dei corsi. Più in particolare, vogliamo poter selezionare i corsi in base alle seguenti categorie di utenti: giovanissimi, giovani, universitari, impiegati, aziende e famiglie.

In primo luogo è necessaria una modifica al Database: nella tabella TCorsi, bisognerà infatti inserire dei campi di tipo bit, per ciascuna delle categorie sopra elencate.

In seguito bisognerà modificare la pagina RicercaCorsi.cfm aggiungendo delle checkbox, ciascuna associata a una nuova variabile di sessione, per poter permettere la selezione delle nuove categorie (**figura 4.10**), e il passaggio alla pagina ElencoIndividuale, dei valori selezionati per ciascuna casella.

figura 4.10: – Aggiunta combobox



Il terzo passo consiste nel modificare la query che seleziona la lista dei corsi in base ai parametri scelti dall'utente, inserendo le condizioni relative ai nuovi campi creati:

Query: CorsiIndividuali

```
<cfquery name="CorsiIndividuali" datasource="DBS">
SELECT IdCorso, NomeCorso, TipoCorso, Lingua, DescrizioneCorso,
Citta, EtaMinAmmessa, EtaMaxAmmessa, Paese, LivelloMinimo
FROM QCorsiIndividuali
WHERE Attivo= 0
<cfif #Session.Paese# NEQ "ALL">AND Paese like '#SESSION.Paese#' </cfif>
<cfif #Session.lingua# NEQ "ALL"> AND Lingua like '#Session.lingua#' </cfif>
<cfif #Session.TCorso# NEQ "ALL">AND TipoCorso like '#Session.TCorso#'</cfif>
<cfif #session.TCitta# NEQ "ALL">AND Citta like '#Session.TCitta#'</cfif>
<cfif #session.SGiovani# NEQ "FALSE">AND TipoCorsoGiovani = '#Session.SGiovani#'</cfif>
<cfif #session.SGiovanissimi# NEQ "FALSE">AND TipoCorsoGiovanissimi='#Session.SGiovanissimi#'</cfif>
<cfif #session.SUniversitari# NEQ "FALSE">AND TipoCorsoUniversitari='#Session.SUniversitari#'</cfif>
<cfif #session.SImpiegati# NEQ "FALSE">AND TipoCorsoImpiegati='#Session.SImpiegati#'</cfif>
<cfif #session.SAziende# NEQ "FALSE">AND TipoCorsoAziende='#Session.SAziende#'</cfif>
<cfif #session.SFamiglie# NEQ "FALSE">AND TipoCorsoFamiglie='#Session.SFamiglie#'</cfif>
ORDER BY Paese, Citta
</cfquery>
```

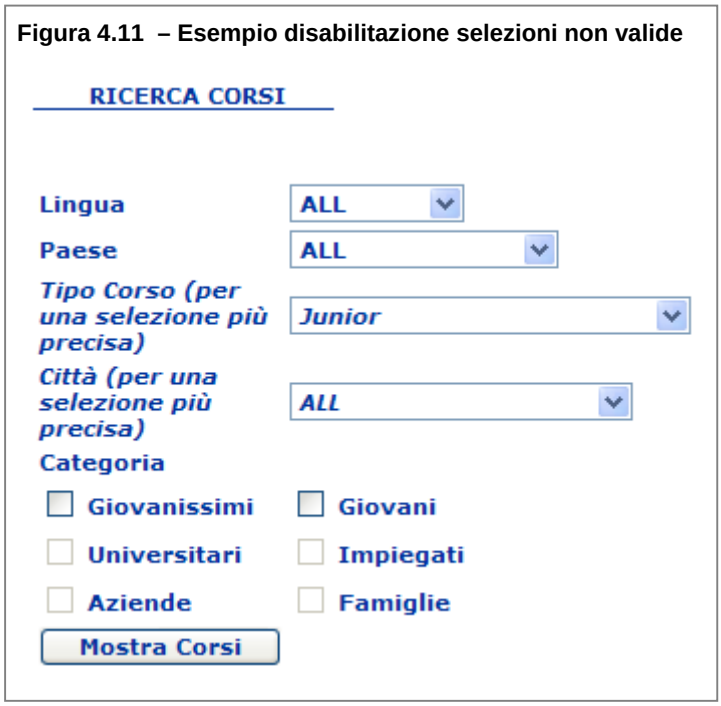
Selezione Corsi Tramite Categoria Utente 2

La soluzione presentata al punto precedente, sebbene sia funzionante, va incontro ad un problema: esiste infatti la possibilità di avere come risultato della ricerca

una pagina vuota, il che risulta terribilmente scomodo e frustrante per l'utente finale.

Per risolvere il problema, è necessario inserire alcuni controlli che impediscano all'utente di effettuare una selezione non valida, cioè di ottenere come risultato una pagina vuota.
Ogni volta che viene effettuata una selezione, è necessario ricaricare la pagina; quando la pagina viene caricata, eseguendo una query per ogni parametro della selezione, possiamo stabilire quali sono le selezioni valide o meno per quel parametro: in questo modo, possiamo semplicemente disabilitare le opzioni che possono generare delle selezioni non valide (la **figura 4.11**, ad esempio, mostra come, selezionando il Tipo Corso *Junior*, vengono disabilitate le checkbox Universitari, Impiegati, Aziende, Famiglie).

Figura 4.11 – Esempio disabilitazione selezioni non valide



Per impedire la presenza di opzioni non valide all'interno delle caselle di selezione, dobbiamo estendere i controlli alle query che filtrano i valori non validi

per le combobox di lingua paese, città e tipo corso, in modo da eseguire i controlli anche sui valori contenuti nelle checkbox della categoria di età.

Per disabilitare le checkbox che possono originare selezioni non valide, è necessario creare delle nuove query che contino, per ogni valore della categoria, in base alle selezioni effettuate, il numero di corsi; se per una categoria questo numero è uguale a 0, bisognerà disabilitare la checkbox relativa a quella categoria; ad esempio, se viene selezionato tipo corso *Junior*, verranno eseguite 6 query che conteranno ciascuna il numero di corsi di tipo *Junior;* se questo numero per una categoria è uguale a zero, la checkbox di quella categoria verrà disabilitata, in modo da impedire all'utente una selezione non valida.

Quinta Fase – Approdare nel WEB

Terminata la fase di restyling dell'applicazione, siamo finalmente pronti per pubblicare il nostro database nel Web, assieme al nostro sito.
Per rendere pubblica la nostra applicazione abbiamo bisogno di 3 cose:

1. Un dominio (nel nostro caso www.mbscambi.com);
2. Un web server, che supporti i comandi CFML (deve essere installato Coldfusion application server), per ospitare il nostro sito;
3. Un Server che ospiti il nostro database.

Fortunatamente, nel Web è facile trovare aziende che offrono la possibilità di ospitare siti web, così come i relativi database nei loro Server e che si occupano di assegnare il dominio da noi scelto all'indirizzo IP dei loro Server (per forsi un

idea basta inserire “web hosting“ in un motore di ricerca).

Dando uno sguardo a qualcuno dei siti web di queste aziende, però, scopriremo subito che la nostra scelta non sarà così facile. Ogni azienda, infatti, offre servizi diversificati rispetto ai concorrenti: si va da un servizio di Hosting dedicato, in cui viene messo a disposizione un intero Web server per la nostra applicazione, a servizi di hosting condiviso, in cui il nostro sito sarà ospitato in un web Server contenente applicazioni di molti altri utenti.

Se si sceglie un piano di Shared Hosting (Hosting Condiviso), ci troviamo spesso e volentieri davanti alla scelta del pacchetto, dove ogni pacchetto presenta servizi e prestazioni via via maggiori, in base al prezzo.
La fase critica, quindi, non è tanto trovare l'azienda che offre il servizio, ma poter identificare tra la moltitudine di operatori del settore del Web Hosting, qual è il più consono alle nostre esigenze, evitando inutili sprechi di risorse economiche abbonandosi ad un piano di hosting fin troppo articolato per le proprie esigenze, oppure scegliendo un servizio di qualità scadente.

Per comprendere qual è il piano che fa per noi, dobbiamo innanzitutto capire quali sono le nostre esigenze, ovvero di che cosa ha bisogno la nostra applicazione per funzionare.
Come abbiamo detto all'inizio del paragrafo, abbiamo bisogno di un web server che sia in grado di gestire i comandi CFML e di un server che ospiti il nostro database: per progettare il sito, abbiamo usato la versione MX di Coldfusion, mentre il nostro database è stato definito in SQL Server 2005. Quindi i parametri essenziali che dovrà avere il nostro hosting saranno il supporto a Coldfusion MX e SQL Server 2005.
Se le dimensioni della nostra applicazione non sono tali da costringerci a scegliere un hosting dedicato, come nel nostro caso, dobbiamo valutare le offerte dei vari pacchetti in base ai seguenti parametri principali:

Dominio: Bisogna verificare se l'hosting offre la possibilità di registrare un dominio, o di cambiare l'indirizzo IP di riferimento di uno già esistente.

Spazio sul disco: La quantità di spazio sul disco del server riservata alla memorizzazione delle pagine e dei file del nostro sito.

Bandwidth: Il traffico massimo mensile, misura il numero di Byte al mese che possono essere inviati dal web Server ai visitatori; le dimensioni del Bandwidth necessario a un applicazione dipendono dall'ampiezza del bacino di utenza del sito e dalla dimensione delle informazioni contenute.

Uptime garantito: Un numero, espresso in percentuale, che esprime la garanzia che il Sito sia sempre accessibile, più alto è questo numero maggiore sarà la garanzia che il nostro sito sarà sempre accessibile.

Statistiche: La possibilità di accedere al log file del sito (contenente le informazioni sugli accessi).

Servizi: Possono comprendere diverse cose: dal pannello di controllo online per gestire i file del sito, alla garanzia, al supporto tecnico, alla sicurezza, alla possibilità di creare caselle di posta elettronica, etc.

TAG CFML Supportati: Alcuni TAG CFML che eseguono delle chiamate al sistema vengono disabilitati per motivi di sicurezza; è necessario controllare che i TAG usati nella propria applicazione siano supportati.

Spazio su SQL Server: La quantità di spazio sul disco del Server SQL, riservato alla memorizzazione del database.

Prezzo installazione:	Il prezzo necessario per attivare il servizio.
Canone mensile:	Il prezzo mensile da pagare per continuare ad usare il servizio.

Individuato l'Hosting ideale per la nostra applicazione, per procedere con la pubblicazione del sito web abbiamo bisogno dell'indirizzo IP del Web Server che ospiterà il nostro sito e del nome o indirizzo IP del Server SQL che ospiterà il nostro database.

Dreamweaver permette di selezionare il metodo di accesso al Server Web in cui andremo a caricare i file del sito, tramite la creazione di una cartella remota. Nel nostro caso, configureremo la cartella remota in modo da potervi accedere usando il protocollo FTP (File Transfer Protocol), che useremo per poter trasferire i file del sito nel Server Web remoto. Sarà necessario inserire:

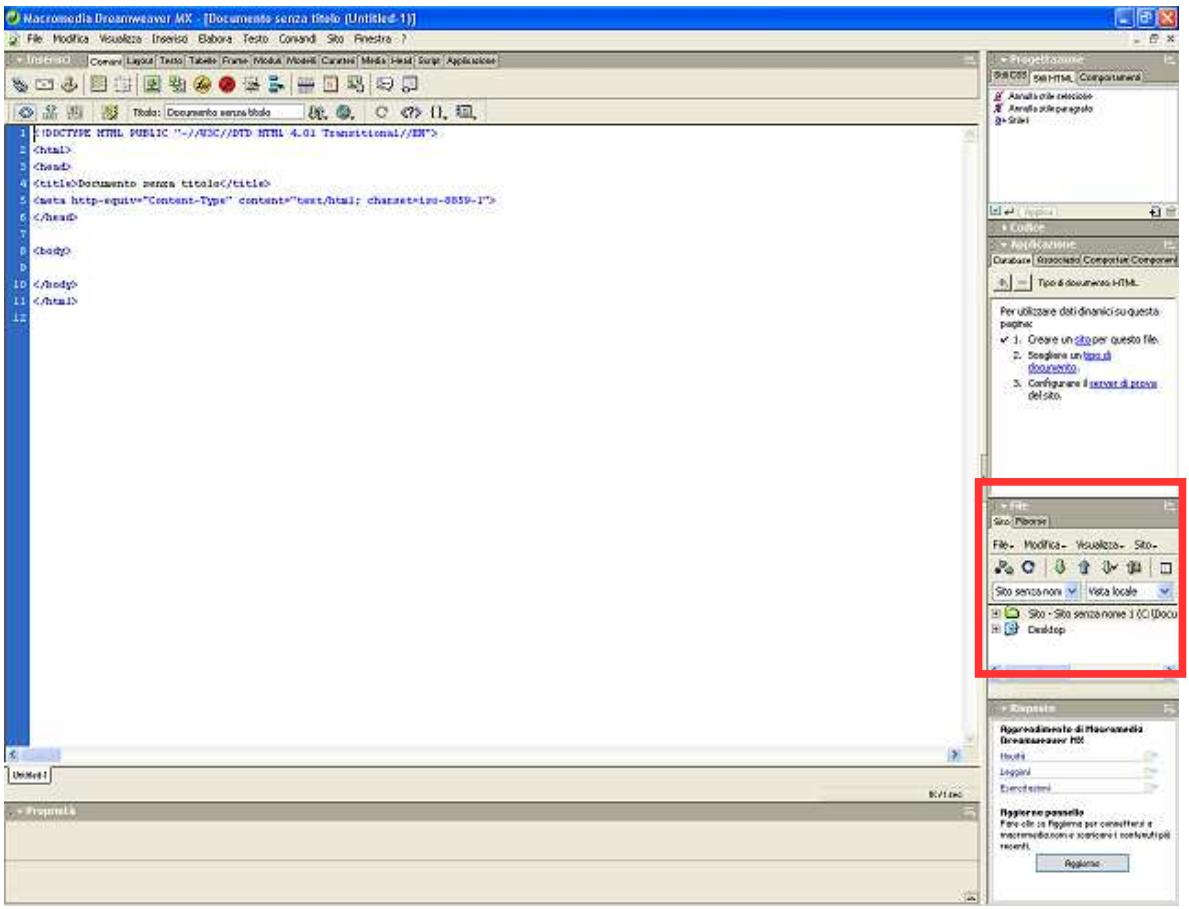
1. Il nome dell'host FTP su cui verranno caricati i file del sito Web. Il nome dell'host FTP è il nome Internet completo di un computer (ad esempio, ftp.mindspring.com);
2. Il nome della directory host del sito remoto in cui sono archiviati i documenti visibili al pubblico;
3. Il login e la password utilizzati per connettersi al server FTP.

Configurata la cartella remota, è sufficiente selezionare i file che desideriamo caricare dalla finestra dei file, e cliccare su carica i/il file (vedi **figura 4.12**).

Per ultimare il progetto dobbiamo caricare il nostro database nel Server del nostro hosting. SQL Server dispone di un tool che permette di trasferire le tabelle da un database SQL Server a un altro; per avviare questo strumento dobbiamo aprire SQL Server Management Studio, fare click con il destro nell'icona rappresentante

il nostro database, selezionare *Task* ed *Export Data.* A questo punto si aprirà una finestra in cui dobbiamo selezionare il database che desideriamo esportare, il nome (o indirizzo IP) del computer in cui è presente il database e l'autenticazione di un utente SQL con permessi di gestione.

figura 4.12 – Trasferire i file nel web Server



Nella finestra successiva ci verranno richieste le stesse informazioni, solo che in questo caso dobbiamo selezionare il database di destinazione, il nome (o indirizzo IP) del computer in cui è presente il database e l'autenticazione di un utente SQL con permessi di gestione. Una volta definite queste informazioni, dobbiamo semplicemente selezionare le tabelle che desideriamo esportare e cliccare su fine.

Terminato il trasferimento del database, dobbiamo semplicemente accertarci dell'effettivo funzionamento della nostra applicazione. Se non incontriamo problemi, possiamo finalmente affermare di aver ultimato il nostro progetto. Ora un qualsiasi utente da ogni parte del mondo può connettersi al nostro sito Web e accedere alle informazioni dei corsi contenute nel nostro database. Ovviamente non possiamo pensare di abbandonare la nostra applicazione, ma dovremo continuare a mantenerla aggiornata, ricaricando periodicamente le tabelle del database in modo che contengano sempre i dati aggiornati, studiando sempre nuove soluzioni per migliorare l'esperienza di navigazione degli utenti del sito e trovando nuovi metodi per poter permettere a quest'ultimi di interagire con il nostro database.