DBMMS: DataBase MultiMedia System

Un sistema per la gestione di contenuti culturali su Internet

Gavino Paddeu, Enrico Stara, Franco G. Tuveri
gavino@crs4.it, ems@crs4.it, tuveri@crs4.it
17 Novembre 1998

Abstract

Questo documento descrive le modalità di attuazione e di realizzazione di alcuni tools per la gestione e l'organizzazione di informazioni aggregate nell'ambito di applicazioni su Internet caratterizzate dal possedere natura multimediale.

Il sistema software, descritto in questo documento, cerca di offrire all'utente, in maniera assolutamente trasparente, la capacità di navigare tra informazioni logicamente correlate ed organizzate. Tali informazioni saranno strutturate in maniera da fornire una chiave di lettura logica attraverso raccolte di dati idealmente guidate e mirate.

Le informazioni andranno a costituire un unico archivio organizzato secondo particolari criteri, utili per rendere piu' semplice e logica la fruizione delle risorse che saranno rese disponibili via Web.

Scenario di riferimento

Attualmente il World Wide Web risulta essere composto da una grande quantità di informazioni di ogni tipo, completamente privo o quasi di organizzazione. In questa situazione l'utente si trova spesso disorientato ed anche i motori di ricerca attualmente disponibili si rivelano inappropriati quando si vuole accedere non a singoli documenti ma, in modo organico e standardizzato, ad informazioni aggiornate e a servizi online. Cio' si ripercuote sulla diffusione sia di Internet sia della cultura informatica in generale ed è causa di un impatto negativo su chi vi si avvicina per la prima volta, in quanto puo' scoraggiare chi non ha la sufficiente esperienza di navigazione e di utilizzo degli strumenti a disposizione.

Internet pertanto rischia di rimanere una tecnologia d'elite riservata prevalentemente alle persone dotate di un discreto background di conoscenze informatiche che fanno uso della Rete per motivi strettamente professionali, accademici e per cultura ed interessi particolari.

D'altra parte, come dimostrato dalle statistiche dei motori di ricerca, appare evidente un impiego della rete per la ricerca di servizi aggiornati ufficiali ed istituzionali.
In particolare si osserva la crescente richiesta da parte degli utenti, di informazioni riguardanti gli aspetti turistici, culturali e ambientali, come di servizi legati alla viabilità ed ai trasporti. Per poter fornire informazioni di questo tipo in modo efficace è necessario gestire in modo adeguato informazioni

georeferenziate, cioè informazioni associate al territorio: regioni, strade di comunicazione, etc.

Il passaggio del Web da semplice vetrina a catalogo utilizzabile come servizio per il reperimento di informazioni in maniera aggregata e organizzata rappresenta la prossima tappa nel cammino di evoluzione di Internet.

Obiettivi

Il sistema discusso nell'ambito di questo documento si propone di definire una struttura capace di ospitare informazioni su un database e permettere la strutturazione di tali dati secondo modalità individuate da chi organizza le informazioni. L'utente del sistema potrà cosi' costruire oggetti come aggregati di informazioni secondo le proprie esigenze e registrarli nel database. La composizione di tali oggetti permetterà di realizzare percorsi per la navigazione tra i dati presenti nel database secondo riferimenti logici appositamente creati ed indicati.

Scelte e metodologie adottate

La ricerca, per quanto riguarda le nuove tecnologie su Internet, propone continuamente nuove soluzioni che producono ricadute sulla qualità dei servizi che il Web puo' offrire, nuove tecnologie che vanno provate e valutate.

Nel caso specifico, le possibilità offerte ci hanno portati a effettuare delle scelte in diversi ambiti, secondo le nostre esigenze.

Tecnologie ad oggetti

La tecnologia ad oggetti costituisce un importante fattore nella reingegnerizzazione del software in quanto permette alle imprese la costruzione di sistemi che rispecchiano i loro processi produttivi. Con questo tipo di tecnologia, l'organizzazione di una compagnia si puo' riflettere nell'organizzazione del software: i processi aziendali possono essere replicati in processi di applicazioni basate sugli oggetti, essendo questi ultimi flessibili abbastanza per essere ristrutturati in modo da riflettere i cambiamenti di tali processi o i nuovi processi che si potranno aggiungere.

UML

La diffusione delle tecnologie Object-Oriented, nelle diverse fasi di vita del software, dalla progettazione ed analisi alla fase di manutenzione fà si che esse si stiano affermando come le piu' corrette durante tutto il ciclo di vita del software, in quanto forniscono innegabili vantaggi, a partire dalla progettazione sino alle fasi finali di sviluppo, alla riusabilità degli oggetti e dei componenti, alla modularità del software.

In questo contesto hanno cominciato a proporsi diversi modelli formali di analisi e progetto, OOA e OOD (Object Oriented Analisys ed Object Oriented Design). Nella situazione attuale ne esistono una dozzina effettivamente utilizzati.

Un metodo di analisi e progetto comprende una notazione per esprimere il modello del sistema, un elenco di documenti di testo associati e una sequenza sistematica di operazioni da compiere per effettuare l'analisi.

L'Unified Modeling Language, è nato nel 1994 ad opera di James Rumbaugh e Grady Booch dalla fusione dei loro rispettivi metodi preesistenti e con il supporto, nel 1995, di Ivar Jacobson che ha condotto alla versione 0.8 del linguaggio UML, comprensivo dell'unione dei tre metodi.

UML nelle versioni 1.0 e 1.1 del settembre 1997 è stato presentato come standard all'OMG, Object Management Group.

UML viene definito linguaggio di modellizzazione e non metodo in quanto un metodo consiste, in linea di principio, sia di un linguaggio di modellizzazione sia di un procedimento. Il linguaggio di modellizzazione è una notazione, principalmente grafica, che i metodi usano per esprimere il design di un prodotto software, il procedimento consiste nei passi da seguire nella realizzazione del design. Il linguaggio di modellizzazione è comunque la parte del metodo che possiede maggiore rilevanza ed importanza in quanto permette a due persone di avere una base di discussione comune per confrontarsi sulla parte di design.

Il linguaggio UML fornisce costrutti in particolare per quanto riguarda le seguenti fasi di sviluppo del software:

- Analisi dei requisiti tramite i casi d'uso
- Analisi e progetto OO
- Modellazione dei componenti
- Modellazione della struttura e della configurazione

Al linguaggio UML, si sono ultimamente affiancati diversi tools che aiutano il progettista o il programmatore a lavorare su un progetto software. Tra questi citiamo doverosamente Rose 98, prodotto dalla Rational, una società proprietà dei tre autori di UML. Rose 98 è sicuramente il prodotto che meglio interpreta la definizione del linguaggio UML e che attualmente domina il mercato degli strumenti di analisi, modellazione e progettazione object-oriented.

Relational DBMS, Object Oriented DBMS

Il diffondersi della tecnologia ad oggetti ha mosso e sta muovendo molte compagnie a migrare verso la tecnologia dei database ad oggetti. Essa semplifica il riutilizzo del codice, fornisce applicazioni meglio strutturate e permette l'organizzazione dei dati in accordo con le esigenze di una applicazione.

A dispetto dei benefici che la tecnologia ad oggetti offre, molte compagnie stanno mantenendo una certa cautela nell'accostarvisi. Gli sviluppatori di applicazioni si trovano spesso a dover risolvere problemi di legacy col software già in uso nell'impresa. Negli ultimi dieci anni sono molte le imprese che hanno investito nell'acquisto di database relazionali e sentono il peso della scelta di dover ripetere una conversione verso una nuova tecnologia. Gli sviluppatori dovrebbero porre la massima attenzione sulla scelta dei tipi di dati che verranno impiegati nell'applicazione. Seppure ormai i database ad oggetti siano una reale e testata tecnologia, i database relazionali continuano e continueranno ad avere il loro ruolo nell'amministrazione di un certo tipo di dati. Inoltre, pure senza tener conto dei problemi di legacy, rimarrà comunque necessaria la possibilità di connessione tra dati di tipo relazionale e dati provenienti da DBMS ad oggetti.

Una delle tendenza di questi ultimi anni, in fatto di archiviazione di informazioni, è quella relativa ai dati di carattere multimediale. In questa situazione gli ODBMS sono generalmente indicati come i sistemi per la gestione di dati di tipo multimediale, sebbene le caratteristiche tecniche in termini di prestazioni siano in definitiva fortemente condizionate dalle caratteristiche hardware delle piattaforme sulle quali tali strumenti software si trovano ad operare e dai problemi legati all’ampiezza di banda delle reti. Di fatto problemi tipici di dimensione dei dati e di throughput di trasferimento in rete, sono ancora motivo di rallentamento nella loro diffusione.

Altro discorso sono i database distribuiti o D-DBMS.

Distributed DBMS

Un sistema distribuito è un sistema in cui sia il trattamento dei dati che delle transazioni vengono divisi su piu’ computers connessi attraverso una rete, in cui ciascun computer gioca uno specifico ruolo nel sistema. In un database distribuito i dati sono distribuiti su piu’ database residenti su piu’ computer.

Un database distribuito appare all’utente come un singolo server ma di fatto puo’ essere costituito da due o piu servers. Si puo’ accedere ai dati su ciascun server simultaneamente e modificarli via rete. Ciascun server del sistema distribuito è controllato dal suo database administrator locale e ciascun server coopera al mantenimento della consistenza delle informazioni del database globale.

La chiave di un database distribuito è la trasparenza delle informazioni. Nè gli utenti nè i programmatori hanno bisogno di sapere dove o come il database mentiene i suoi dati. All’utente le operazioni appaiono come eseguite su un unico database.

La gestione di un database distribuito comporta inoltre una serie di attenzioni particolari relative all’amministrazione della rete, traffico, volume di transazioni, sicurezza, e relativo alla gestione degli aspetti fisici dei computers che ospitano i database quali dischi, sistemi di backup, etc.

La gestione dell’aggiornamento dei dati residenti su database distributi possiede un livello di complessità elevato. Il fatto che piu’ utenti possano condividere l’accesso ai dati su differenti siti piuttosto che su un singolo sito è causa di una serie di considerazioni che riguardano la gestione della propagazione dell’aggiornamento dei dati (Replication), il controllo della concorrenza, la gestione del "recovery" delle transazioni.

Tra i prodotti commerciali piu’ noti, troviamo i prodotti della Oracle, Ingres e Multi-Star, prodotti che si basano su DBMS relazionali. Per quanto riguarda la tecnologia ad oggetti, citiamo i piu’ famosi, Versant 5.0, ObjectStore della Object Design, e il database della O2 Technology. Questi sistemi permettono la creazione di set di schemi su differenti locazioni fisiche distribuite sulla rete, garantendo la trasparenza della distribuzione dei dati e la sincronizzazione tra essi.

La realizzazione di una struttura comune che gestisca la replicazione delle informazioni attraverso database relazionale e ad oggetti e mantenga in maniera distribuita le informazioni su una rete geografica appare attualmente una valida alternativa all’uso di uno specifico prodotto. È quanto avviene con "Mariposa" un sistema, progettato presso la "University of California" di Berkeley da Michael Stonebraker.

In questa situazione viene definito un oggetto che possiamo chiamare "Transaction Manager" le cui funzioni sono quella di conoscere l’ubicazione delle informazioni richieste e la possibilità di gestire operazioni di modifica-aggiornamento e sincronizzazione tra i server.

JavaBeans

Un JavaBean è un componente software riutilizzabile che può essere manipolato con un builder tool visuale. Un JB si configura come un modulo funzionale che può avere caratteristiche grafiche di visualizzazione utilizzabili in una GUI, in tal caso potrebbe essere un semplice bottone o un banner, ma anche qualcosa di più complesso come una finestra di dialogo o un template; oppure potrebbe essere un modulo software non grafico che ad esempio si occupa di interfacciare un database.

In generale i JavaBeans hanno tra loro in comune una piattaforma di funzionalità che supportano:

- l'introspezione che permette tramite un builder tool di analizzarli;
- la customizzazione che permette ad un builder tool di configurare le loro caratteristiche e di modificarne il comportamento;
- la gestione degli eventi che costituiscono una piattaforma di comunicazione tra gli stessi JB e tra il JB e l'ambiente che lo circonda;
- il settaggio delle properties utilizzabili in fase di configurazione e programmazione;
- la persistenza che consente il salvataggio del JB nel suo stato, ad esempio una volta eseguita la sua customizzazione;

Il JB si configura quindi come un modulo software riutilizzabile, ma questo non significa che ogni classe Java che mostri avere caratteristiche modulari debba diventare un JB; il componente software JB diventa necessario quando si presenta la necessità di una manipolazione visuale del modulo o della sua customizzazione.

Occorre distinguere quelli che sono i due principali ambienti in cui un JB viene eseguito.

Il primo di questi è in generale un ambiente di sviluppo, dove viene eseguita la configurazione del JB e la modifica del suo comportamento; da ciò ne deriva che il JB deve avere con se una parte di codice che ne consenta la customizzazione.

Il secondo è l'ambiente run-time, ossia l'applicazione che utilizza il JB già configurato, nel quale il codice del JB relativo alla customizzazione non viene eseguito. I componenti JB vengono eseguiti localmente nello stesso spazio, quindi nella stessa Java Virtual Machine del loro contenitore (una Java Application, un Java Applet o più in generale un container);

L'architettura JB consente inoltre di lavorare efficientemente in sistemi distribuiti grazie al supporto della Java RMI (Remote Method Invocation). In un sistema client-server i JB potrebbero occuparsi dell'interfaccia GUI lato client, ed accedere via RMI, in maniera del tutto trasparente, ai metodi remoti dell'application server.

Per concludere, non necessariamente i JB presentano una rappresentazione GUI, essi possono anche fornire funzionalità diverse come, ad esempio, quelle di accesso a Data Base o in generale quelle di moduli software operanti lato server.

CORBA, DCOM, RMI

Tra le infrastrutture middleware piu' diffuse possiamo citare i tre principali modelli di distributed computing oggi in competizione tra loro: DCOM (Distributed Component Object Modelling) della Microsoft, CORBA (Common Object Request Broker Architecture) del consorzio OMG, Object Management Group, e Java RMI (Remote Method Invocation) della SUN Microsystem.

CORBA

CORBA costituisce un elemento fondamentale nel passaggio dalla tecnologia desktop-centric alla nuova concezione network-centric, in quanto permette di far migrare una applicazione tipicamente client-server verso un insieme di componenti collaborativi.

La tecnologia ad oggetti distribuiti di CORBA, permette di unire complessi sistemi client-server estendendo e assemblando oggetti. Il valore maggiore di questa tecnologia object oriented è quello di fornire flessibilità nell'eseguire una componente nella sua piattaforma nativa pur essendo utilizzata da una applicazione residente su un'altra piattaforma. Il linguaggio standard di Corba per la definizione dell'interfaccia degli oggetti è il suo IDL, Interface Definition Language, caratterizzato dall'essere puramente dichiarativo e progettato per ambienti distribuiti. Esso è inoltre disponibile per linguaggi C, C++, JAVA, SmallTalk, ADA, COBOL.

DCOM

Analogamente a CORBA, DCOM separa l'interfaccia dell'oggetto dalla sua implementazione e richiede che tutte le interfacce siao dichiarate usando una IDL. L'IDL della Microsoft è basato su DCE (Distributed Computing Environment) e, ovviamente, non è compatibile con CORBA. Un componente DCOM puo' supportare interfacce multiple per permettere la riusabilità di un oggetto incapsulando l'intefaccia. Poichè non supporta l'ereditarietà, DCOM consente il riutilizzo degli oggetti attraverso l'aggregazione e il contenimento.

Un oggetto DCOM non è strettamente un oggetto come inteso dalla tecnologia Object Oriented. Le interfacce non hanno uno stato e non possono essere istanziate per creare un oggetto unico. Una interfaccia DCOM è piuttosto un gruppo di funzioni correlate.

Java RMI

JAVA RMI, Remote Method Invocation, è un modello di distributed object per la piattaforma Java. RMI è unico in quanto è un modello basato sul linguaggio che si avvantaggia del sistema network-centric di Java. RMI estende il modello a oggetti di Java oltre lo spazio di indirizzi di una singola macchina virtuale (VM). I metodi degli oggetti possono essere invocati fra diverse VM attraverso la rete e gli oggetti reali possono essere passati come argomenti o restituiti durante l'invocazione del metodo. Java RMI utilizza la serializzazione degli oggetti per convertire le loro proprietà in stream binari per il trasporto. Qualunque tipo di oggetto Java puo' essere passato durante l'invocazione inclusi i tipi primitivi, classi core, classi definite dagli utenti e JavaBeans. Java RMI puo' essere descritto come una naturale evoluzione dell'RPC (Remote Procedure Call) adattato al paradigna object-oriented.

I tool software

La valutazione di quanto sopra indicato, ci ha portato a effettuare le scelte seguenti sulla base di una analisi approfondita dei requisiti tecnici dei tools software disponibili oggi sul mercato:

- **Database: Oracle Enterprise Manager nella versione 7.3 per SUN solaris**
  Oracle è il database relazionale piu' venduto al mondo. È un software estremamente affidabile, ma complesso nella sua gestione. Si è scelto di lavorare nel campo database con la tecnologia relazionale in quanto possediamo all'interno del gruppo una discreta esperienza che ci permette di valutare con una certa chiarezza i costi di progettazione della base dati e la sua realizzazione.

- **Middleware: Java RMI**
  La Remote Methods Invocation permette di invocare da parte client, metodi di classi che risiedono su un server, in esecuzione su un'altra macchina. Una buona progettazione permette di distinguere tra parte client e parte server in modo da rendere piu' leggero possibile il client per motivi di prestazione e lasciare al server le operazioni piu' onerose, quali connessione al database, risoluzione di query complesse ed altro.

- **Ambiente di sviluppo: JavaBeans e Jbuilder**
  Jbuilder della Borland, nella versione Professional è l'ambiente di sviluppo che abbiamo scelto. JBuilder2 offre una vastissima gamma di strumenti di sviluppo visuali per creare applicazioni distribuite a livello aziendale. Fornisce:
  - ambiente di sviluppo RAD visuale
  - piu' di 200 componenti JFC/Swing e JBCL
  - BeansExpress per la creazione dei JavaBean
  - VisiBroker integrato per lo sviluppo di applicazioni CORBA/IIOP
  - DataExpress per la generazione di applicazioni per database

  Alcune difficoltà sono state riscontrate nella creazione ed utilizzo degli oggetti di JBuilder con Oracle. Spesso siamo stati costretti a terminare l'applicazione per riprendere il controllo della macchina.
  In ogni caso l'approccio al problema consigliato dai manuali non si è dimostrato adeguato al nostro caso costringendoci a formulare nuove soluzioni di procedimento per affrontare il problema della connessione ad Oracle.

- **Progettazione ed Analisi: Rational Rose98**
  Rational Rose98 fornisce un ottimo supporto per UML. Supporta UML 1.1 e permette la modellazione visuale in tutte le fasi del progetto, dall'analisi dei requisiti alla realizzazione dell'applicazione.
  Si è scelto di lavorare nella parte di progettazione con UML, Unified Modelling Language, per procedere seguendo determinate linee di progettazione e in modo da definire uno standard interno per la documentazione.

  In particolare si intende utilizzare Rose98 per quanto riguarda le opzioni di *reverse engineering* su alcune parti del codice.
  Il reverse engineering è il processo di creazione o aggiornamento di un modello analizzandone il codice. Rose98 per Java consente di effettuare il reverse engineering del codice trovando le classe e gli oggetti dai files e includendoli in un modello.

La struttura delle informazioni

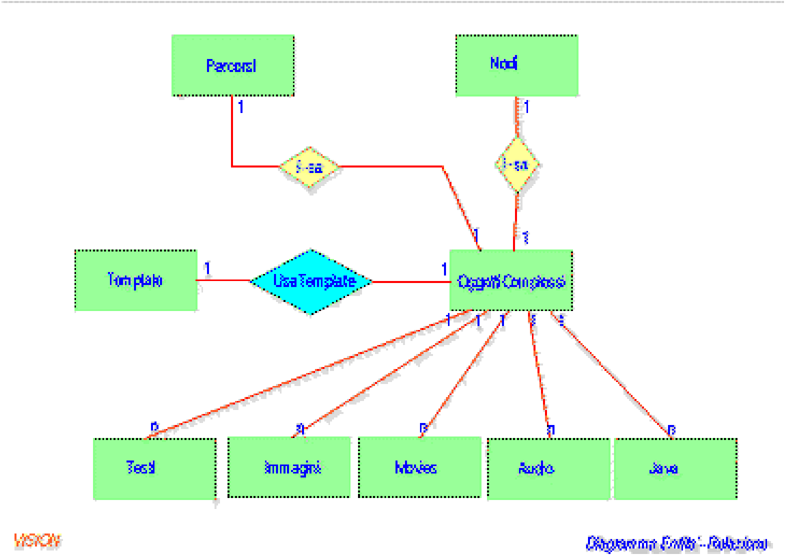
Il sistema permette la catalogazione e la navigazione tra le informazioni individuando come possibili

applicazioni risorse quali musei, biblioteche siti archeologici, percorsi turistici.
Le informazioni, immagini, testi, files audio e video, classi Java, documenti in diversi formati, che chiameremo oggetti di tipo semplice, vengono amministrate dal sistema tramite un database. Esse sono classificate e organizzate secondo particolari strutture che noi abbiamo definito oggetti complessi. Gli Oggetti Complessi sono degli oggetti costituiti da una serie di componenti che possono essere a loro volta Oggetti Complessi o oggetti di tipo semplice rappresentati dalle informazioni base del database.

Costituire gli oggetti complessi come unione di piu' componenti fornisce il vantaggio di una gestione evoluta delle informazioni, in una visione di un loro utilizzo piu' moderno delle informazioni e maggiormente adeguato al mezzo che Internet vuole rappresentare, ed è questo uno degli aspetti essenziali del sistema.

Una seconda particolarità dovuta alle caratteristiche degli Oggetti Complessi è la possibilità di definire dei Percorsi per muoversi tra le informazioni strutturate del database.
L'organizzazione in Percorsi permette di suggerire una o piu' modalità di fruizione delle informazioni, secondo organizzazioni logiche precedentemente settate.

Il diagramma Entità-Relazione illustra il modello dei dati del progetto software.



Le Entità in basso alla figura indicano le tabelle relative agli oggetti semplici, non strutturati, cioè Testi, Immagini, files audio e files movies nei diversi formati ed eventuali classi Java. Le tabelle conterranno esclusivamente i riferimenti del percorso sul file system in cui gli oggetti risiedono fisicamente, esclusi i testi.

L'Entità OggettiComplessi è l'oggetto portante dello schema del database. Infatti, nella tabella che essa rappresenta vi sono le informazioni associate ai Nodi, ai Percorsi ed ai Template, che sono gli oggetti di più alto livello del nostro sistema.

La tabella OggettiComplessi contiene le informazioni necessarie per poter identificare gli oggetti di alto livello quali possono essere Quadro, Museo, Foto, etc.

Il concetto è il seguente: ciascun record della tabella OggettiComplessi contiene le informazioni relative al template che viene utilizzato per la realizzazione di quell'oggetto complesso tramite un riferimento alla tabella Template, un riferimento alla tabella relativa ai tipi di oggetto semplice che assieme formano l'oggetto composto. Un identificatore permette di raggruppare gli oggetti semplice che costituiscono l'oggetto complesso. Il template costituisce il contenitore e la struttura dell'oggetto composto. La tabella UsaTemplate fornisce la relazione che permette di associare un template ad un oggetto complesso.

Il nome dell'oggetto serve ad identificare il tipo di oggetto complesso in uso.

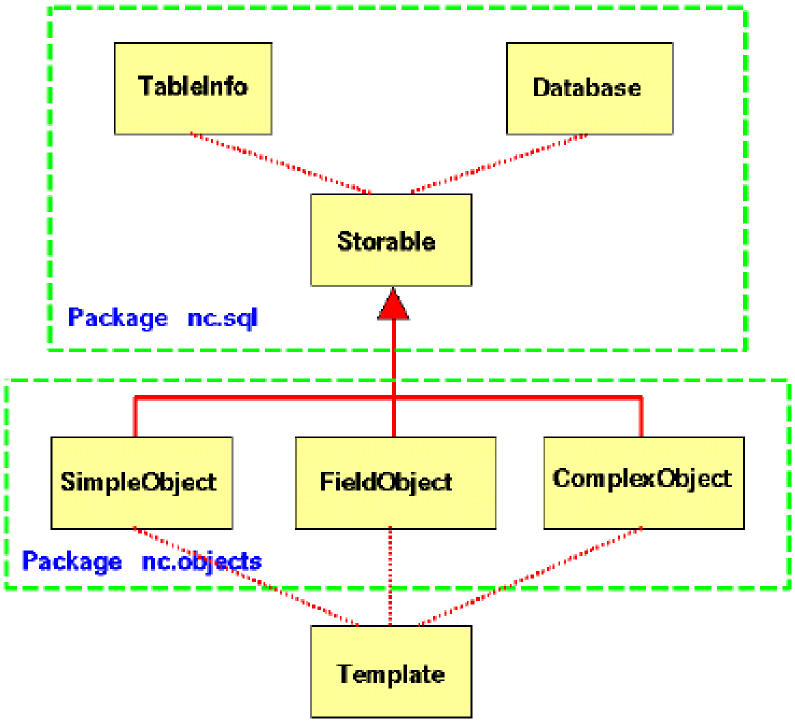
Al livello superiore troviamo le Entità Nodi e Percorsi relative alle tabelle omonime. La tabella Nodi generalizza le informazioni della tabella OggettiComplessi in quanto ne estende il significato con il concetto di Nodo.
Un Nodo fa parte di un Percorso, perciò avrà necessità di possedere informazioni sull'oggetto complesso e sul Percorso al quale tale Nodo è associato.

La tabella Percorsi, poichè anche il Percorso è un oggetto di tipo complesso, conterrà anche essa dei riferimenti alla tabella OggettiComplessi, più un campo relativo al proprio nome. Questo riferimento è sufficiente per definire l'oggetto Percorso in quanto il template associato conterrà le informazioni necessarie per poter effettuare la navigazione sul Percorso.

I Packages

Il lavoro descritto in questa sezione è relativo alla sola parte server, Di fatto il client usufruisce fortemente di quanto viene sviluppato in questa parte in quanto la progettazione e la realizzazione dei Bean va ad impattare sull'insieme delle caratteristiche dell'interfaccia che utilizzerà questi stessi Bean per la visualizzazione delle informazioni gestite dall'applicazione.

Il package nc.sql



Sql è il package che contiene le classi che si interfacciano via JDBC al database Oracle.

Lo schema riportato nella figura a fianco mostra la relazione tra le classi che compongono il package.

Il Package **nc.sql** è caratterizzato dalle seguenti classi:

- Storable
- DataBase
- TableInfo

La classe **Storable** attraverso i suoi metodi consente di interrogare il database relativamente ad una generica tabella e recuperarne, tramite la classe TableInfo, i metadati cioè quelle informazioni relative ai campi della tabella, alla loro dimensione e al loro tipo.
Storable costruisce in base a queste informazione il record tipo di quella tabella. Possiede inoltre i metodi per l'archiviazione, la modifica e la cancellazione di un dato record. La classe **DataBase** gestisce la connessione al DBMS Oracle, carica il driver proprietario per la gestione di JDBC e verifica l'esito della connessione.

La classe **TableInfo** recupera le informazioni sui metadati delle colonne delle varie tabelle e permette a

Storable di costruire il record per quella tabella.

Sopra il package **nc.sql** si posiziona il package **nc.objects**, un package costituito dalle seguenti tre classi:

- SimpleObject
- FieldObject
- ComplexObject

Queste classi sono quelle che si occupano di comporre e decomporre gli oggetti complessi.
La composizione, come la scomposizione, devono essere ricorsive in quanto un oggetto complesso puo' contenere a sua volta piu' oggetti complessi.

La loro funzione è quella di mappare in oggetti Java le tabelle del database.

La classe **SimpleObject** permette di lavorare sulle tabelle relative agli oggetti semplici, quali Testi, Immagini, Audio, Movie e Java. Queste tabelle sono caratterizzate dall'avere due soli campi, di cui il primo è l'identificatore della riga ed il secondo è il contenuto o il percorso dell'oggetto in questione.

Le classi **FieldObject** e **ComplexObject** permettono rispettivamente di lavorare sulle tabelle Fields e ComplexObject. Attraverso i metodi, get* set*, è possibile lavorare sui singoli campi della tabella Fields.

Su questi due package lavora il package **nc.template**, un package costituito da una classe particolare che consente l'utilizzo dei JavaBeans come templates per l'applicazione. Attraverso Template.class è infatti possibile invocare il Bean capace di gestire un particolare oggetto complesso recuperato dal database.

Lo schema del database

Lo schema del database del sistema è realizzato secondo le specifiche standard di SQL 92 per il database Oracle Enterprise Manager v.7.3.

La progettazione ha dovuto tenere conto delle necessità derivanti dalla volontà di gestire informazioni composite costituite da oggetti di tipo testuale, immagini, files audio e video, eventualmente in diversi formati, e classi Java.

La struttura è stata realizzata secondo i diversi livelli di complessità dell'informazione. Ad un livello piu' basso gli oggetti semplici, ad un livello intermedio le informazioni relative agli oggetti composti e al livello superiore le informazioni relative ai Nodi ed ai Percorsi logici.

La sezione seguente illustra la progettazione di questa parte del sistema.

Oggetti complessi

Un Oggetto complesso come già detto è un oggetto composto da altri oggetti, complessi e semplici.
Supponiamo di avere un oggetto complesso Fotografia composto dagli oggetti semplici:

- una immagine (foto)
- un testo che la descrive (didascalia)

- il nome dell'autore (autore)

Quando l'oggetto complesso Fotografia viene inserito nel database avviene che prima viene inserita una nuova riga nella tabella ComplexObjects, quindi vengono inseriti i records relativi agli oggetti semplici nelle tabelle del livello fisico, in questo caso specifico le tabelle Testi ed Immagini e per ultimi vengono aggiornati i campi nella tabella Fields.
Ogni volta che ciascun oggetto semplice viene inserito, attraverso la sequence relativa, l'applicazione mantiene traccia dell'ID del record dell'oggetto, cio' serve per registrare nella tabella Fields l'avvenuta registrazione dell'oggetto semplice e la sua appartenenza ad un certo oggetto complesso.
Per ultimo viene registrato l'oggetto complesso nella tabella Fields, in modo che l'ID che gli viene assegnato possa essere inserito nella colonna ID_Obj delle righe della tabella Fields relative agli oggetti semplici di tipo immagine e di tipo testo.

Vediamo l'esempio attraverso il contenuto delle tabelle:

Tabella: Testi	
ID	**CONTENT**
2	Anfora greca del II sec. A.C.
3	Lisistrato Ipoterione

Tabella: Immagini	
ID	**CONTENT**
2	/usr/people/DBMMS/archivio/immagini/anfora.gif

Tabella: Fields				
ID	**ID_OBJ**	**FIELDTYPE**	**FIELDID**	**NAME**
2	5	Testi	2	Didascalia
3	5	Testi	3	Autore
4	5	Immagini	2	Immagine

Tabella: ComplexObjects		
ID	**TYPE**	**NAME**
5	Fotografia	Anfore greche

Il secondo aspetto importante del sistema è quello relativo alla navigazione tra le informazioni aggregate. Questo avviene attraverso i concetti fondamentali di **Nodo** e di **Percorso**. Sia i Nodi che i Percorsi sono oggetti complessi. Un insieme di Nodi ordinati secondo una logica costituisce un Percorso. Un Percorso permette la navigazione tra i Nodi che lo compongono. L'utente definisce, attraverso un apposito tool, una serie di collegamenti tra i Nodi che costituiscono il percorso logico che sarà seguito durante la navigazione. Un Percorso puo' contenere Nodi già presenti in altri Percorsi. Ogni Percorso ha vita a se.

L’eliminazione dei Percorsi deve tenere conto della presenza dei propri Nodi in altri Percorsi.

Un esempio molto semplificato di Nodi e di Percorso puo’ essere il seguente:

Un museo contiene diverse sale, ciascuna identifica periodi artistici diversi. Per ciascun periodo artistico si individuano le opere presenti nelle sale. Per costruire un Oggetto Complesso "opera" si puo’ partire dall’immagine di ciascuna opera accompagnata da una breve didascalia e da una nota sull’autore. Queste informazioni vengono aggregate per costituire l’oggetto "opera". Esisterà un ulteriore oggetto complesso che chiameremo "sala", e sarà costituito da un insieme di opere. Ci saranno diverse istanze dell’oggetto sala, una per ciascuna sala del museo. Ciascuna sala verrà quindi identificata come possibile Nodo di un Percorso. Quando si definisce il Percorso verranno recuperati questi possibili Nodi e ordinati logicamente, in modo da consigliare una possibile visita al Museo.

Vediamo ora quanto avviene a livello del database. L’interrogazione per il reperimento degli oggetti complessi relativi ad un dato argomento avviene attraverso un controllo sulla tabella Fields, in alcune colonne della quale, etichettate in maniera opportuna, sarà possibile identificare caratteristiche che ne permettono l’identificazione.

Una prima interrogazione mi restituisce gli oggetti complessi candidati a diventare Nodi di un Percorso.

L’utente seleziona quindi gli oggetti secondo un ordine logico di visita in modo che essi possano essere ordinati e costituire un Percorso. La definizione del Percorso nel database avviene attraverso l’inserimento nella tabella Nodi dell’ID dell’oggetto complesso selezionato, del Percorso appena creato e da un indice necessario per identificare l’ordine di lettura dei Nodi. Nella tabella Percorsi viene registrato invece il nome del Percorso e l’ID relativo all’oggetto complesso che costituisce il Percorso.

La struttura del Percorso viene regolata tramite un template.

Un template non è altro che un Bean specializzato nella gestione di un tipo particolare di oggetto complesso. Questa definizione di template è necessaria in quanto oggetti complessi differenti trattano differenti tipi di dati, organizzati all’interno di una interfaccia definita al momento della progettazione del template. Il template quindi puo’ essere riutilizzato per applicazioni diverse aventi oggetti complessi comuni o riprogettato a seconda delle esigenze dell’utente.

Conclusioni

Il sistema permette di studiare la realizzabilità di alcuni tools per la gestione e l’organizzazione di informazioni aggregate in vista di ulteriori evoluzioni del sistema e della sua eventuale applicabilità nell’ambito di altre applicazioni.

Il sistema software in fase di sviluppo, offrirà all’utente, in maniera assolutamente trasparente, la capacità di navigare tra informazioni logicamente correlate ed organizzate fornendo una chiave di lettura logica in campi di applicazione identificati e collegati al patrimonio turistico e culturale della Sardegna.

I packages **nc.sql**, **nc.objects nc.template**, fanno parte di una parte piu complessa, un server, alle cui funzionalità si potrà accedere via RMI, Remote Method Invocation da un client, costruito attraverso i JavaBeans, che costituirà la parte di interfaccia al sistema.

Riferimenti Bibliografici

The knowledge-based economy,
Rapporto OCSE, OCSE/GD(96), 1996

Europe and the global Information Society, *1994*

R.Orfali, D.Harkey:
Client/Server programming with Java e CORBA,
John Wiley 1998

M.T. Özsu and P. Valduriez:
Principles of Distributed Database Systems,
Prentice-Hall, 1991.

Michael Stonebraker,
EECS Dept., University of California at Berkeley
Mariposa: Distributed Database Management System
http://mariposa.CS.Berkeley.EDU:8000/mariposa/

R.Ben-Natan:
CORBA: a guide to Common Object Request Broker Architecture,
McGraw Hill, 1995

A.Vogel, K.Duddy:
Java Programming with CORBA (2 edition),
John Wiley, 1998

M.Fowler, R.Scott:
UML Distilled: Applying the standard Object Modeling Language,
Addison-Wesley, Object Technology Series, 1998

M. Marchesi:
UML: il nuovo standard per analisi e progetto OO
Dipartimento di Ingegneria Elettrica ed Elettronica, Univ. Cagliari.

RMI Architecture and Functional Specification
Sun Microsystems, Inc
Java Remote Method Invocation, 1997

JavaBeans v.101
http://java.sun.com/beans/index.html
Editor Graham Hamilton Sun Microsystems, Inc.