UNIVERSITÀ DEGLI STUDI DI MODENA
E REGGIO EMILIA
Facoltà di Ingegneria - Sede di Modena
Corso di Laurea in Ingegneria Informatica

Progetto del sito Web della Facoltà di Ingegneria di Modena: parte seconda

Relatore
Chiar.mo Prof.
Sonia Bergamaschi

Correlatore
Ing. Alberto Corni

Tesi di Laurea di
Cristian Samuel

Controrelatore
Chiar.mo Prof.
Giovanni S. Barozzi

Anno Accademico 1999 - 2000

Parole chiave:

Analisi dei requisiti
Database design
Web publishing framework
XML
XSL

RINGRAZIAMENTI

Desidero ringraziare la Prof. Sonia Bergamaschi per l'aiuto fornito durante la realizzazione della tesi e per la costante disponibilità dimostrata.
Un ringraziamento va inoltre al Prof. Giovanni S. Barozzi ed alla sig.ra Adriana Zuffi per la cortese collaborazione durante la fase di definizione dei requisiti.
Ringrazio infine l'Ing. Alberto Corni, per i consigli ed i chiarimenti di ordine pratico ed implementativo.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Premessa

Il presente lavoro di tesi è finalizzato alla progettazione ed alla parziale realizzazione del nuovo sito Web della Facoltà di Ingegneria dell'Università di Modena e Reggio Emilia.

Presentazione del progetto

Questa sezione ha lo scopo di fornire una visione globale ed introduttiva delle motivazioni e delle linee generali del lavoro ed è comune alla tesi [1], svolta nell'ambito del medesimo progetto.

Linee generali

Il contesto di riferimento di cui ci occupiamo sta attraversando una fase di transizione, sia dal punto di vista dell'organizzazione didattica che dal punto di vista logistico. Per quanto riguarda il primo aspetto, il passaggio al nuovo ordinamento e il probabile ampliamento dell'offerta didattica avranno un significativo impatto sull'organizzazione attuale. Inoltre, il trasferimento presso la nuova sede e il presunto potenziamento dell'organico comportano ulteriori cambiamenti.

Il sito dovrà quindi essere caratterizzato da una struttura flessibile, capace di rispondere alle future necessità. L'aggiornamento frequente delle informazioni, dovuto a questa situazione in continuo mutamento, comporta l'esigenza di concepire funzioni di manutenzione dei contenuti rapide e di semplice utilizzo.

Il successo del sito dipenderà in buona parte dalla sua usabilità. Occorre fare in modo che esso diventi uno strumento immediato ed intuitivo da utilizzare, sia da parte dei fruitori delle informazioni (docenti, studenti) ma anche di chi deve produrre, pubblicare ed aggiornare i contenuti.

E' utile prevedere un meccanismo di personalizzazione dell'accesso al sito,

finalizzato a presentare alle diverse tipologie di utenti soltanto i servizi di loro interesse ed a salvaguardare la sicurezza e riservatezza delle informazioni.

Occorre individuare accuratamente le peculiarità e le funzioni specifiche della Facoltà in rapporto a quelle del Dipartimenti (attualmente uno solo), allo scopo di ottenere una corretta integrazione con i servizi Web di Ateneo.

Il sito, oltre ad essere un contenitore di informazioni e servizi, deve rappresentare un ideale biglietto da visita con cui dimostrare le capacità di progettazione ed innovazione tecnologica proprie delle Facoltà di Ingegneria.

Da ultimo, per facilitare la consultazione da parte di utenti stranieri, il sito dovrà essere bilingue (italiano - inglese).

Raccolta dei requisiti

La prima fase della progettazione è consistita nella raccolta dei requisiti. A questa attività, fondamentale per la buona riuscita del lavoro, abbiamo dedicato particolare attenzione.

Per prima cosa abbiamo identificato diverse tipologie di utenti da interpellare allo scopo di individuare le informazioni e le funzioni che il futuro sito dovrà contenere.

Le categorie di utenti che abbiamo individuato sono le seguenti:

- cariche istituzionali (Preside, Presidenti di Corso di Laurea e di Diploma, . . .)
- docenti
- studenti
- visitatori generici

Abbiamo quindi organizzato gli incontri con gli utenti, grazie ai quali è stato possibile ampliare le specifiche del progetto. In particolare si è privilegiato il dialogo con il Preside, quale massima carica istituzionale della Facoltà e principale committente. Abbiamo inoltre interpellato diversi professori e ricercatori. Gli aspetti organizzativi e burocratici sono stati approfonditi con la collaborazione della Segreteria di Presidenza. Ulteriori informazioni sono state fornite dai responsabili della gestione del Web di Ateneo e del Dipartimento di Scienze dell'Informazione della Facoltà. Da ultimo, abbiamo cercato di interpretare le esigenze degli studenti e dei visitatori generici.

I requisiti raccolti sono stati raggruppati secondo le seguenti tipologie:

Requisiti relativi agli aspetti istituzionali si riferiscono agli aspetti relativi alla struttura organizzativa ed istituzionale della Facoltà, nonché alle informazioni di rappresentanza.

Requisiti relativi alla didattica comprendono la descrizione dell'offerta didattica ed i servizi di supporto alle attività di insegnamento.

Requisiti relativi ai servizi collaterali per gli studenti contemplano gli aspetti legati all'associazionismo studentesco, ai progetti di interscambio con università straniere e ad altre iniziative rivolte agli studenti.

Analizziamo ora con maggiore dettaglio le categorie sopra elencate.

Requisiti relativi agli aspetti istituzionali

- Il sito dovrà fornire una presentazione generale della Facoltà, comprendente documenti che descrivono sinteticamente gli obiettivi e le finalità dei corsi di studio e le opportunità offerte da questi. Ci saranno inoltre brevi notizie storiche ed informazioni sulla nuova sede.
- È prevista la pubblicazione del Regolamento di Facoltà.
- Verrà descritta l'organizzazione della Facoltà nei suoi vari aspetti, quali ad esempio le cariche istituzionali, i consigli e le commissioni. Il sito dovrà inoltre rendere note le convocazioni dei consigli e delle commissioni, con i relativi ordini del giorno. Sarà mantenuto un archivio dei verbali e documenti prodotti dai suddetti organi.
- Verranno fornite informazioni di servizio quali l'ubicazione e gli orari di apertura di dipartimenti e centri, segreteria studenti, biblioteca, aule e laboratori.
- Sarà presente un elenco del personale consultabile attraverso diverse chiavi di ricerca, contenente informazioni per ogni singolo nominativo. Nel caso dei docenti saranno indicati, tra le altre cose, l'orario di ricevimento, gli insegnamenti tenuti, le attività di ricerca e le pubblicazioni e le cariche istituzionali.
- È necessario prevedere uno spazio dedicato alle pratiche burocratiche relative al personale della Facoltà, con la disponibilità di moduli e scadenziari.
- Infine, sarà presente una bacheca virtuale in cui compariranno avvisi, comunicazioni e notizie di interesse generale.

Requisiti relativi alla didattica

- Verrà pubblicata la sezione del Regolamento Didattico di Ateneo riguardante la Facoltà.
- Verrà fornita una descrizione dettagliata dell'offerta didattica. Per ogni singolo percorso formativo verranno specificati il Manifesto degli Studi, la propedeuticità tra gli insegnamenti, le norme generali e le eventuali regole di ammissione.
- Saranno resi disponibili il calendario dell'anno accademico, i calendari delle sessioni di esame (con le date degli appelli e la composizione delle commissioni d'esame), le date delle sessioni di laurea e di diploma e delle sessioni per gli esami di abilitazione professionale.
- Per quanto riguarda gli appelli, è prevista la possibilità di iscriversi online alle liste d'esame e ricevere gli esiti delle prove sostenute via e-mail.
- Ogni insegnamento avrà uno spazio apposito, tramite il quale il docente potrà rendere disponibili il programma del corso, il materiale didattico ed eventuali comunicazioni.
- In concomitanza con l'inizio dei semestri sarà pubblicato l'orario delle lezioni.
- Le attività didattiche collaterali quali master, scuole di specializzazione post lauream e Accademia Militare, avranno anch'esse uno spazio specifico.

Requisiti relativi ai servizi collaterali per gli studenti

- Gli studenti potranno esaminare le proposte di tesi suggerite dai docenti.
- Adeguato rilievo verrà dato alle notizie relative ai progetti di studio all'estero quali Erasmus e Socrates.
- È previsto uno spazio dedicato alle pratiche burocratiche di pertinenza degli studenti (piani di studio personalizzati, richiesta di stage presso aziende, richieste di finanziamento per attività culturali...)
- Saranno disponibili spazi a cura delle associazioni studentesche e delle liste universitarie.
- Con l'obiettivo di assistere lo studente nella vita universitaria, verranno forniti informazioni e link verso i servizi dell'Ateneo (Servizio Immatricolazioni, Arestud, CUS ...) e della rete civica.

Soluzioni tecnologiche adottate

Il principio ispiratore che ha guidato la scelta dell'architettura è stato quello di raggiungere una netta separazione tra la logica applicativa, i contenuti e la presentazione. Questa soluzione consente di suddividere gli aspetti dello sviluppo tra competenze ben distinte. Il programmatore si occupa di implementare la logica applicativa, disinteressandosi della presentazione; il Web design può essere delegato ad una figura specializzata che non deve necessariamente conoscere gli aspetti relativi alla programmazione. Gli autori dei contenuti, inoltre, possono intervenire direttamente sui contenuti senza avere alcuna nozione o visibilità degli aspetti logici dell'applicazione.

I benefici sono evidenti: è possibile modificare i contenuti e la grafica del sito senza dover mettere mano al codice dell'applicazione e viceversa. Inoltre la bassa commistione tra le competenze permette una maggiore specializzazione dei ruoli, facilitando il lavoro delle diverse figure coinvolte.

La soluzione adottata per conseguire gli obiettivi sopra descritti si basa sull'utilizzo di pagine dinamiche, generate a partire dalle informazioni contenute in una base di dati. L'architettura studiata presenta come elementi caratterizzanti l'uso dei linguaggi XML, XSL e Java unitamente a DBMS relazionali.

Dinamicità vs. staticità

Quando si considera l'integrazione tra Web e database, il concetto fondamentale è la *dinamicità*, contrapposto al concetto di *staticità*. [2]

- La pagina Web *statica* riflette, in ogni momento, la situazione esistente nell'istante in cui è stata creata e non quella dell'istante in cui viene letta.
- La pagina Web *dinamica* è una pagina che non esiste a priori, ma viene generata sul momento, a seguito di una richiesta dell'utente. Per questo motivo rifletterà in ogni momento la situazione esistente nell'istante in cui viene letta.

Affidarsi esclusivamente a pagine di tipo statico, soprattutto in presenza di siti vasti e complessi, che richiedono aggiornamenti frequenti, presenterebbe costi di gestione inaccettabili. La modifica delle informazioni contenute nel sito comporterebbe infatti l'aggiornamento manuale di centinaia (o migliaia) di pagine, richiedendo un grande sforzo per assicurare la coerenza dei contenuti, specie nel caso di singole informazioni richiamate in più pagine.

L'utilizzo di un database come luogo logico dove conservare i dati, da distribuire sul Web tramite pagine dinamiche, rappresenta una radicale soluzione a que-

sti problemi. La robusta e consolidata tecnologia dei DBMS costituisce un'ottimo strumento per gestire grandi quantità di dati in maniera efficiente e flessibile.

Il beneficio principale di questo approccio è una drastica riduzione dei costi di gestione. Un aggiornamento delle informazioni immagazzinate nella base di dati si riflette immediatamente ed automaticamente nei contenuti del sito, senza alcun intervento di modifica manuale delle pagine.

Inoltre, l'utilizzo di un database consente di garantire l'integrità e la coerenza dei dati in maniera semplice. Un ulteriore vantaggio è la possibilità di eseguire agevolmente manipolazioni complesse delle informazioni a disposizione, rendendo facile l'introduzione di nuove funzionalità ed aggregazioni di informazioni.

Contenuto della tesi

Nell'ambito del progetto del sito Web della Facoltà, il presente lavoro si concentra sugli aspetti relativi alla didattica ed ai servizi per gli studenti. La tesi [1] si occupa invece degli aspetti relativi all'attività degli organismi istituzionali della Facoltà.

Gli obiettivi della tesi sono i seguenti:

- Progettare e realizzare, dopo un fase di dettagliata analisi delle specifiche e dei requisiti, il database relazionale di supporto per gli aspetti citati.
- Proporre una soluzione architetturale per l'implementazione del sito Web e realizzare un prototipo che dimostri la bontà della scelta effettuata.

La tesi è organizzata come segue:

Nel **Capitolo 1** viene presentata l'analisi dettagliata dei requisiti relativi agli ambiti della didattica e dei servizi per gli studenti.

Il **Capitolo 2** illustra la progettazione e la realizzazione del database. Dopo aver presentato le fasi di progetto concettuale e di progetto logico, viene descritta l'implementazione dello schema relazionale prodotto.

Nel **Capitolo 3**, dopo un'esame delle problematiche legate allo sviluppo di siti Web, descrive l'architettura proposta per la realizzazione del sito, mettendone in luce i vantaggi e le particolarità.

Il **Capitolo 4** presenta la realizzazione del prototipo. Viene dapprima descritta l'implementazione dell'architettura di base. Successivamente si passa all'analisi delle funzionalità create per il sito Web.

La tesi è corredata da due appendici.

L'**appendice A** presenta lo schema E/R globale, risultato dell'integrazione degli schemi concettuali prodotti in questo lavoro di tesi e nella tesi [1].

Le fasi di progettazione e realizzazione dell'architettura dell'applicazione hanno comportato, oltre all'impiego degli strumenti e delle nozioni acquisiti acquisiti durante il corso di laurea, la necessità di effettuare uno studio comparativo sulle diverse tecnologie a disposizione per la pubblicazione di contenuti dinamici sul Web. Tale panoramica è illustrata nell'**appendice B**.

Capitolo 1

Analisi dei requisiti

Nell'introduzione si è visto come i requisiti del progetto siano stati raggruppati in tre distinte categorie, che corrispondono essenzialmente alle tipologie di servizi che il sito dovrà fornire. E' stata anche presentata una lista dei requisiti di base che sono emersi durante le prime fasi di raccolta e definizione delle specifiche.

In questo capitolo si procederà all'analisi dettagliata e al raffinamento delle specifiche iniziali, per le parti relative agli aspetti didattici ed ai servizi per gli studenti, con lo scopo di ottenere una lista completa e dettagliata dei requisiti.

La raccolta delle specifiche necessarie al progetto è stata portata a termine attraverso lo svolgimento di diverse attività:

- In primo luogo, sono stati organizzati numerosi incontri con diverse tipologie di utenti. tali incontri hanno prodotto la parte più rilevante delle informazioni.

- Contestualmente alle interviste, sono stati valutati numerosi siti Web di università italiane ed estere, allo scopo di effettuare un'analisi delle soluzioni già realizzate. Quest'attività di confronto ha fornito ulteriori spunti e idee per completare la raccolta dei requisiti.

- Da ultimo, è stata una effettuata una fase di rifinitura e di riorganizzazione allo scopo di conciliare e rendere coerenti i diversi aspetti.

Data la particolare natura di un'applicazione Web, molto spesso i requisiti sono espressi in termini di informazioni e di documenti che devono essere pubblicati. Ci sono poi requisiti di tipo funzionale, che descrivono le funzionalità che il sito deve fornire agli utenti.

Nel seguito si analizzano i requisiti facendo riferimento all'organizzazione di massima proposta per la parte del sito riguardante la didattica. In questo modo è possibile avere una visione più chiara della collocazione delle informazioni e delle funzionalità al'interno del sito.

1.1 I requisiti

1.1.1 Requisiti generali

Dalla home page del sito si accede alla sezione riguardante la didattica. La pagina principale di tale sezione deve fornire una visione d'insieme dell'offerta didattica della Facoltà, visualizzando l'insieme dei corsi di studio, dei master, delle scuole di specializzazione e dei corsi di dottorato disponibili.

Vengono ora esaminate le diverse informazioni che devono essere accessibili da questa pagina.

Regolamento Didattico

Deve essere consultabile la sezione di Regolamento Didattico di Ateneo riguardante la Facoltà di Ingegneria.

Orario delle lezioni

Le tabelle con gli orari delle lezioni sono un tipo di informazione molto importante e con alta frequenza di consultazione. È quindi preferibile che il percorso di navigazione sia mantenuto breve, per facilitare l'accesso: per questo motivo, gli orari delle lezioni devono essere accessibili direttamente dalla pagina principale della didattica (sarà ovviamente necessario che l'utente specifichi il corso di studio e l'anno di corso richiesti). Le tabelle con l'orario delle lezioni devono riportare, oltre ai nomi degli insegnamenti, anche le aule dove le lezioni hanno luogo.

Calendario degli esami di Laurea

Occorre pubblicare le date delle sessioni degli esami di Laurea e di Diploma.

Calendario degli esami di abilitazione professionale

Occorre pubblicare le date degli esami di abilitazione professionale.

Proposte di tesi

È necessario prevedere uno spazio dedicato a raccogliere le diverse proposte di tesi suggerite dai docenti. Ogni proposta di tesi deve presentare informazioni circa l'argomento trattato, la durata stimata e il tipo di impegno richiesto. Deve essere inoltre possibile aggiungere immagini a corredo della descrizione della proposta di tesi stessa.

1.1.2 Requisiti relativi ai Corsi di Studio

Presentazione

Testo che descrive il corso di studio, illustrandone gli obiettivi formativi, la strutturazione e gli sbocchi professionali.

Avvisi

È necessario dedicare uno spazio agli avvisi riguardanti uno specifico Corso di Studio. Tali avvisi possono essere immessi direttamente dai docenti.

Manifesto degli studi

Documento che elenca gli insegnamenti attivi per l'anno accademico in corso, suddivisi per anno di corso e per periodo didattico di appartenenza. E' necessario conservare la serie storica dei manifesti degli studi degli anni accademici precedenti.

Elenco degli insegnamenti

Oltre al manifesto degli studi, è utile prevedere una lista comprendente tutti gli insegnamenti propri del Corso di Studio, consultabile in ordine alfabetico o attraverso una ricerca.

Lista delle propedeuticità

Bisogna predisporre una lista delle propedeuticità fra gli esami.

Calendario delle sessioni d'esame

È necessario pubblicare un calendario delle sessioni d'esame, con le date d'inizio e fine di ogni sessione.

Calendario degli appelli d'esame

Occorre pubblicare un calendario degli esami, suddiviso nelle diverse sessioni. Il calendario deve riportare le date degli appelli, l'aula dove l'appello si tiene e il tipo di verifica (scritta od orale).

Iscrizione alle liste d'esame

Lo studente deve avere la possibilità di iscriversi online alla lista dell'esame che intende sostenere. Deve anche essere prevista la possibilità, in caso di ripensamento, di cancellarsi dalla lista.

Esiti degli esami

Si prevede di rendere disponibili gli esiti degli esami attraverso lo strumento informatico. Per ragioni di privacy, non viene pubblicata sul sito la lista dei voti ricevuti dagli studenti; una soluzione migliore consiste nel comunicare l'esito di un esame direttamente ad ogni studente tramite posta elettronica. Questo sistema è reso possibile dal fatto che un progetto, in corso di attuazione, prevede l'assegnazione di una casella di posta elettronica per ogni studente dell'Ateneo.

1.1.3 Requisiti relativi agli insegnamenti

Uno dei principali motivi per la realizzazione di sezioni del sito dedicate ai singoli insegnamenti è la volontà di costituire un efficace mezzo per lo scambio di informazioni fra il docente e gli studenti.

La pagina di un insegnamento deve innanzitutto mostrare il nome del titolare del corso, con la possibilità di accedere alla sua pagina personale.

Avvisi del docente

Sulla pagina del proprio insegnamento il docente potrà pubblicare comunicazioni ed avvisi che riguardano l'insegnamento stesso (cambiamenti nell'orario di lezione, avvisi relativi alle verifiche, scadenze di consegna per elaborati o tesine, ...).

Programma dell'insegnamento

Ogni insegnamento prevede un proprio programma, che illustra gli argomenti trattati e propone i libri consigliati dal docente. È necessario che venga memorizzato il programma relativo all'anno accademico in corso; un docente, se lo ritiene, può decidere di mantenere i programmi relativi agli anni accademici precedenti.

Materiale didattico

Come supporto all'attività didattica, il docente ha spesso la necessità di fornire ai propri studenti materiale didattico di vario genere, tra cui (ad esempio) dispense, esercizi risolti, testi d'esame o testi di tesine. È utile quindi avere la possibilità di accedere a tali documenti in formato elettronico, per poterli scaricare, dalla pagina dedicato ad un insegnamento. Per alcuni tipi di materiale didattico è necessario sapere anche l'anno accademico di riferimento.

Altre informazioni

Devono essere presenti i link necessari per effettuare le seguenti operazioni:

- visualizzare le date degli appelli
- iscriversi alle liste d'esame relative all'insegnamento in questione
- visualizzare le eventuali propedeuticità richieste
- visualizzare l'orario delle lezioni dell'insegnamento

1.1.4 Requisiti relativi ai servizi per gli studenti

Per progettare un'area del sito dedicata agli studenti, occorre pensare al soddisfacimento di diverse esigenze. Di seguito vengono elencate le principali necessità da prendere in considerazione.

Progetti di studio al'estero

Occorre prevedere un adeguato spazio per le informazioni riguardanti i programmi di studio all'estero come Erasmus.

Ausilio per le pratiche burocratiche

Di grande utilità è la previsione di uno scadenziario che riporti le principali date entro cui presentare moduli e richieste. Si può inoltre pensare di preparare una raccolta dei principali moduli utilizzati per le pratiche burocratiche riguardanti gli studenti, affinché gli studenti possano scaricare tali moduli dal sito allo scopo di stamparli ed utilizzarli.

Studenti stranieri

Allo scopo di soddisfare le necessità di orientamento proprie degli studenti stranieri che studiano o desiderano studiare presso la Facoltà di Ingegneria, è utile preparare una guida (in inglese) contenente informazioni utili circa la vita universitaria, le possibilità di alloggio, i servizi della città.

Associazioni studentesche

Occorre pubblicare una lista delle associazioni studentesche, con le informazioni utili del caso.

Liste universitarie

Si deve predisporre un elenco delle liste universitarie. I rappresentanti degli studenti, inoltre, devono avere la possibilità di pubblicare eventuali comunicazioni rivolte agli studenti.

1.2 Tabella riassuntiva dei requisiti

I requisiti e le specifiche esaminati in precedenza vengono ora analizzati dal punto di vista delle informazioni che producono.

Allo scopo di facilitare la gestione ed il coordinamento dei contenuti del sito, ogni requisito è stato arricchito con una serie di ulteriori dati. Le informazioni che sono state considerate sono elencate di seguito.

- **Responsabile**: indica il responsabile del contenuto dell'informazione
- **Editore**: è la persona che produce l'informazione in questione
- **Publisher**: rappresenta la persona che si incarica di rendere disponibile l'informazione sul web
- **Utente**: indica le categorie di utenti maggiormente interessati all'informazione.
- **Frequenza di aggiornamento**: stima circa la frequenza indicativa di aggiornamento dell'informazione sul sito
- **Versione bilingue**: indica la necessità di prevedere una versione inglese dell'informazione
- **Protezione**: stabilisce se l'informazione è visibile liberamente o è soggetta a restrizioni d'accesso.

I requisiti sono stati raccolti e sintetizzati in forma tabellare. La legenda dei simboli utilizzati si trova a pag. 19.

Requisiti generali

Regolamento Didattico

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
P	Sf	W	D,R,Srap	a	No	No

Orario delle lezioni

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	D	W	S,D,T,R	t	No	No

Calendario degli esami di Laurea

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	Sf	W	S	a	Sì	No

Calendario degli esami di abilitazione professionale

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
P	Sf	W	S	a	No	No

Lista delle proposte di tesi

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	W	S	m	Sì	No

Requisiti relativi ai Corsi di Studio

Presentazione dei Corsi di studio

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	Sf	W	S,V	a	Sì	No

Avvisi

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	g	No	No

Manifesto degli Studi

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
P	Sf	W	S,V	a	No	No

Elenco degli insegnamenti

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	Sf	W	S,V	a	No	No

Lista delle propedeuticità

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	Sf	W	S,V	a	No	No

Calendario delle sessioni d'esame

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	Sf	W	S,D,R	t	Sì	No

Calendario degli appelli d'esame

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	Sf	W	S,D,R	t	No	No

Iscrizione alle liste d'esame

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	t	No	Sì

Esiti degli esami

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	m	No	Sì

Requisiti relativi agli insegnamenti

Titolare del corso

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	a	No	No

Orario di ricevimento

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	t	No	No

Avvisi del docente

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	g	No	No

Programma dell'insegnamento

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	a	No	No

Materiale Didattico

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	m	No	No

Date degli appelli dell'insegnamento

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	m	No	No

Orario delle lezioni dell'insegnamento

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	a	No	No

Requisiti relativi ai servizi per studenti

Progetti di studio all'estero

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
D	D	D	S	g	Sì	No

Scadenziario per moduli e richieste

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
W	Sf	W	S	g	Sì	No

Moduli e informazioni

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
Pcds	Sf	W	S	a	Sì	No

Studenti stranieri

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
W	S	W	S	g	Sì	No

Associazioni studentesche

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
W	S	W	S	g	No	No

Liste universitarie

Resp	Ed	Pub	Utente	FA	Bilingue	Protezione
W	Srap	W	S	g	No	No

Legenda dei simboli utilizzati

Ruoli:

P	Preside
Pcds	Presidente di Corso di Studio
Sf	Segreteria di Facoltà
D	Docenti
R	Ricercatori
T	Tecnici
S	Studenti
Srap	Rappresentanti degli studenti
V	Visitatori occasionali del sito
W	Webmaster

Frequenza di aggiornamento (FA):

a	annuale
t	trimestrale
m	mensile
g	giornaliera

Capitolo 2

Il Database

In questo capitolo si descrive il progetto e la realizzazione del database preposto alla gestione dei dati di interesse per il sito, concentrando l'attenzione sugli aspetti riguardanti il supporto all'attività didattica ed ai servizi per gli studenti.

La metodologia seguita per il progetto è quella classica usata per i database relazionali. Essa si articola nei seguenti passi:

- **Progetto concettuale**
  L'obiettivo di questa fase è quello di rappresentare la realtà di interesse ad un alto livello di astrazione. Si giunge alla costruzione dello *schema E/R* (Entity/Relationship), una rappresentazione grafica che descrive i concetti della realtà in esame e le associazioni che li legano.

- **Progetto logico**
  Consiste nella traduzione dello schema concettuale in uno schema logico, nella fattispecie lo schema logico relazionale. Tale trasformazione è indispensabile per l'implementazione della base di dati, dal momento che non esistono DBMS in grado di operare direttamente sugli oggetti di uno schema E/R.

Nell'ultima parte del capitolo verrà descritta l'implementazione dello schema ottenuto su RDBMS Oracle 8.0.5.

Per facilitare la creazione di una base di dati unitaria e coerente nell'ambito del progetto globale del sito, la stesura dello schema concettuale ha richiesto un'opera di coordinamento con l'altro lavoro di tesi [1] impegnato nel medesimo progetto. Lo schema E/R globale, risultato del lavoro di integrazione dei due schemi realizzati, viene presentato in Appendice A.

2.1 Progetto concettuale

Per la costruzione dello schema E/R verrà impiegata una strategia di tipo *mixed*. Tale metodo prevede l'utilizzo combinato dell'approccio *top-down*, tramite il quale lo schema finale viene ottenuto per successivi raffinamenti a partire da concetti generali, e dell'approccio *bottom-up*, che prevede l'integrazione progressiva di concetti elementari in schemi di complessità crescente.

2.1.1 Definizione dello schema scheletro

Allo scopo di facilitare la costruzione dello schema, conviene suddividere i requisiti in sottoinsiemi da considerare separatamente. I collegamenti tra le varie partizioni vengono rappresentati in un apposito *schema scheletro*, che serve quindi da base di partenza per lo schema finale.

Dall'esame dei requisiti emergono alcuni concetti fondamentali che rappresentano dei buoni candidati per lo schema scheletro.

- Innanzitutto, occorre definire un'entità CORSO_DI_STUDIO, attraverso la quale si possano rappresentare i dati relativi ai diversi corsi di studio offerti dalla Facoltà (laurea, laurea specialistica, corsi del vecchio ordinamento quali laurea e diploma di laurea e corsi di dottorato).
- Un'altra importante entità è INSEGNAMENTO, alla quale fa capo la gestione delle informazioni legate ai singoli insegnamenti.
- La gestione dell'orario delle lezioni e degli appelli d'esame viene delegata al concetto ESAMI_LEZIONI, lasciato deliberatamente vago per essere esaminato nel dettaglio più avanti.
- Esaminando i requisiti, si nota che alcuni tipi di informazioni, di valenza generale, non possono essere ricondotte nè ai corsi di studio nè ai singoli insegnamenti. Per rappresentare questa porzione di specifiche viene creata un'entità FACOLTÀ.
- Occorre infine memorizzare i dati relativi ai docenti ed agli studenti. Per questo motivo vengono introdotte le entità DOCENTE e STUDENTE.

Le entità appena definite sono collegate fra loro dalle associazioni descritte di seguito.

- DOCENTE è collegata ad INSEGNAMENTO dall'associazione INSEGNA.

- Ogni corso di studio comprende diversi insegnamenti: per esprimere questo legame si introduce l'associazione INSERITO_IN.

- ESAMI_LEZIONI è in relazione sia con INSEGNAMENTO che con STUDENTE, attraverso le due associazioni Associati_a e PARTECIPA. Ogni studente, inoltre, è associato al corso di studio da lui scelto tramite l'associazione ISCRITTO.

- Da ultimo, ARTICOLATA_IN collega CORSO_DI_STUDIO e FACOLTÀ e quest'ultima è legata all'entità DOCENTE dall'associazione DIPENDE.

In Figura 2.1 si può vedere lo schema scheletro risultante.

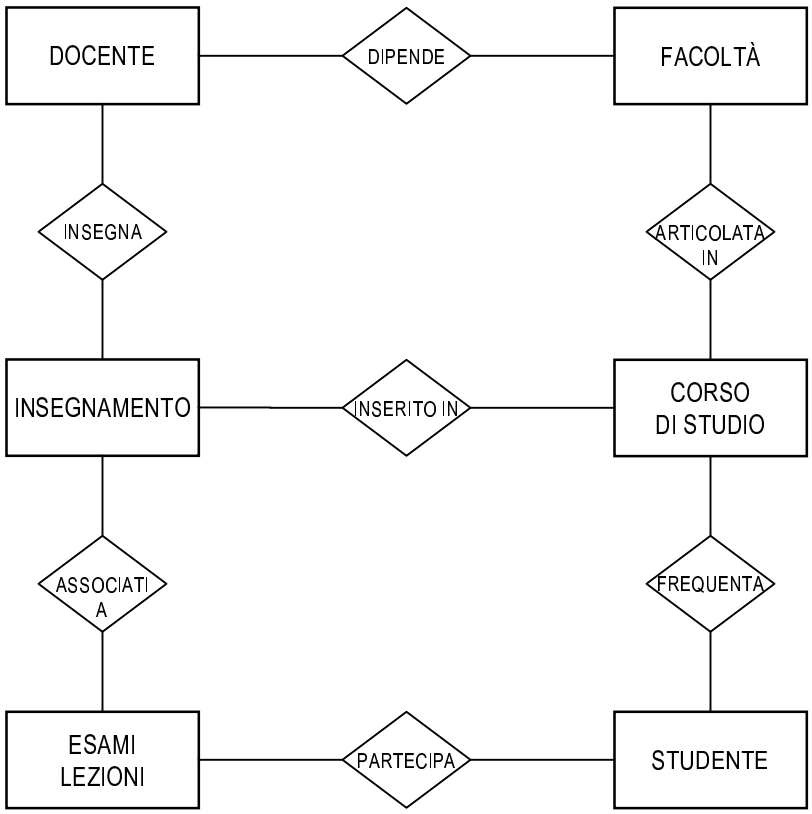


Figura 2.1: Schema scheletro

Dopo aver definito lo schema scheletro, occorre procedere analizzando singolarmente i sottoinsiemi di requisiti individuati, allo scopo di raffinare i concetti di base e farli evolvere in sottoschemi maggiormente articolati.

2.1.2 Raffinamento dell'entità *Corso_di_Studio*

Le istanze dell'entità CORSO_DI_STUDIO possono essere sia i corsi di studio propriamente detti, tra cui quelli previsti dal nuovo ordinamento (laurea di primo livello e laurea specialistica), sia i corsi di studio particolari, quali corsi di dottorato, master e scuole di specializzazione.

Esaminiamo ora gli attributi dell'entità. Un corso di studio è identificato dal codice alfanumerico *CodCorso*, e possiede una denominazione (*nome*). Allo scopo di poter agevolmente distinguere i diversi tipi di corso, è opportuno inserire un attributo *tipo*. La tabella 2.1 riporta un esempio dei valori che l'attributo può assumere.

Valore	**Significato**
L	Laurea di 1^o livello
LS	Laurea Specialistica
Lold	Corso di Laurea (vecchio ordinamento)
D	Corso di Diploma (vecchio ordinamento)
DO	Corso di Dottorato
M	Master
SS	Scuola di Specializzazione

Tabella 2.1: Valori dell'attributo *Tipo*

Ogni corso di studio deve prevedere una pagina di presentazione, che ne illustri le finalità e le prerogative e ne dia una sintetica descrizione, e tale pagina deve avere una versione inglese. Per soddisfare questo requisito si ha quindi la necessità di memorizzare i testi della descrizione, nelle due diverse lingue.

A causa delle dimensioni non irrisorie che questi testi possono raggiungere, si è scelto di conservare tali presentazioni in file esterni al database e di memorizzare solamente i riferimenti ad essi relativi. Naturalmente una soluzione di questo genere comporta l'onere di mantenere la consistenza tra l'organizzazione dei file nel file system e i riferimenti presenti nella base di dati, ma produce indubbi vantaggi in termini di praticità.

L'entità necessita pertanto di due ulteriori attributi, *Presentazione_I* e *Presentazione_E*, nei quali memorizzare i riferimenti ai file contenenti il testo di presentazione rispettivamente in versione italiana ed inglese.

Rimane ora da considerare un ultimo aspetto. Dall'esame dei requisiti emerge che tra le categorie di avvisi che occorre presentare all'utente ce n'è una che

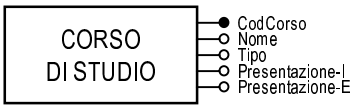


Figura 2.2: Entità CORSO_DI_STUDIO

riguarda strettamente i singoli corsi di studio. È necessario quindi introdurre una nuova entità, che viene chiamata AVVISI_CDS.

L'identificatore scelto è *CodAvv*, un codice che permette di distinguere in maniera univoca gli avvisi. Si introducono poi due attributi di tipo data: il primo, *Data*, rappresenta la data in cui l'avviso viene pubblicato; il secondo, *DataScad*, rappresenta la data di scadenza, oltre la quale l'avviso o la notizia perde la sua utilità e può essere rimosso dal sistema.

L'attributo *Descr* è una breve descrizione da presentare come "titolo" dell'avviso; il testo vero e proprio dell'avviso stesso è memorizzato tramite l'attributo *Testo*. Si noti che, dal momento che non si è ritenuto necessario proporre questa categoria di avvisi in versione bilingue, è presente il solo testo in italiano. E' stato introdotto, infine, un attributo *Autore*, che permette di identificare il referente della notizia pubblicata.

AVVISI_CDS (figura 2.3) è collegata a CORSO_DI_STUDIO tramite l'associazione CDS_AVV. Dal momento che un avviso riguardante un corso di studio, se esiste, è riferito ad un solo corso, la cardinalità dell'entità AVVISI_CDS nell'associazione è (1,1). Un corso di studio, d'altra parte, può avere zero o più avvisi che lo riguardano, per cui la cardinalità di CORSO_DI_STUDIO in CDS_AVV è (0,n).

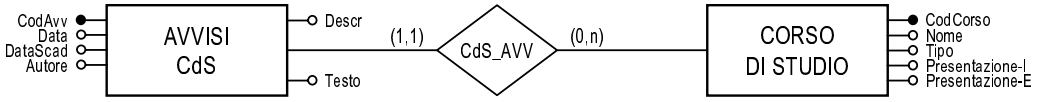


Figura 2.3: Entità AVVISI_CDS

2.1.3 Raffinamento dell'entità *Insegnamento*

Le specifiche indicano che le informazioni da memorizzare relativamente agli insegnamenti costituiscono un insieme articolato e piuttosto eterogeneo. Iniziamo la nostra analisi esaminando per prima cosa i dati che possono essere agevolmente rappresentati sotto forma di attributi dell'entità *Insegnamento* (figura 2.4).

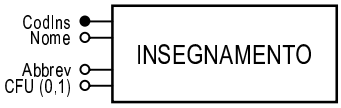


Figura 2.4: Entità INSEGNAMENTO

Innanzitutto, un insegnamento ha un codice ad esso associato (*CodIns*), che ne costituisce anche l'identificatore. Si ha poi la denominazione dell'insegnamento (*Nome*).

L'attributo *Abbrev* contiene il nome in forma abbreviata. L'utilità di questo attributo è evidente nei casi in cui occorre presentare aggregazioni di informazioni quali ad esempio la tabella settimanale con l'orario delle lezioni. In una situazione del genere, infatti, occorre fare i conti con lo spazio a disposizione sulla pagina web.[1] Per non essere costretti ad utilizzare un carattere troppo piccolo, o peggio a spezzare la tabella in più porzioni, degradando in maniera inaccettabile la leggibilità dell'orario, è necessario usare i nomi delle materie in forma abbreviata.

Per completare gli attributi relativi ad INSEGNAMENTO, rimane da memorizzare il numero di Crediti Formativi Universitari (CFU) che vengono attribuiti allo studente al superamento delle verifiche di una determinata materia. Occorre quindi introdurre l'attributo opzionale *CFU*, che assumerà valore nullo per gli insegnamenti che non appartengono ai corsi di laurea del nuovo ordinamento didattico.

Il programma

Ogni insegnamento prevede un proprio programma, che illustra gli argomenti trattati e propone i libri consigliati dal docente. E' stato richiesto che venga memorizzato il programma relativo all'anno accademico in corso e quelli dei due anni

[1] Tale spazio, com'è logico, è direttamente legato alle dimensioni del monitor utilizzato dall'utente. Si deve considerare che la misura media dei monitor attualmente più diffusi non supera i 15 pollici di diagonale, per cui occorre pensare ad una risoluzione di riferimento massima di 1024×768 punti.

accademici precedenti. Anche in questo caso, trattandosi di testi piuttosto estesi, si preferisce utilizzare dei file separati, dei quali si annota la posizione all'interno del file system.

La soluzione scelta per memorizzare tali informazioni (illustrata in figura 2.5) è quella di creare un'entità PROGRAMMA, con i due attributi *Programma* e *AnnoAcc*, che contengono rispettivamente il riferimento al file esterno e l'anno accademico a cui il programma si riferisce.

L'entità deve essere collegata ad INSEGNAMENTO tramite l'associazione SEGUE. La partecipazione di INSEGNAMENTO è obbligatoria (un insegnamento deve infatti avere almeno il programma dell'anno accademico corrente), con cardinalità massima uguale a n, dal momento che un docente, se lo ritiene, può mantenere la serie storica dei programmi degli anni precedenti.

L'identificatore usato è di tipo composto e *mixed*, formato dall'associazione con INSEGNAMENTO e dall'attributo *AnnoAcc*.

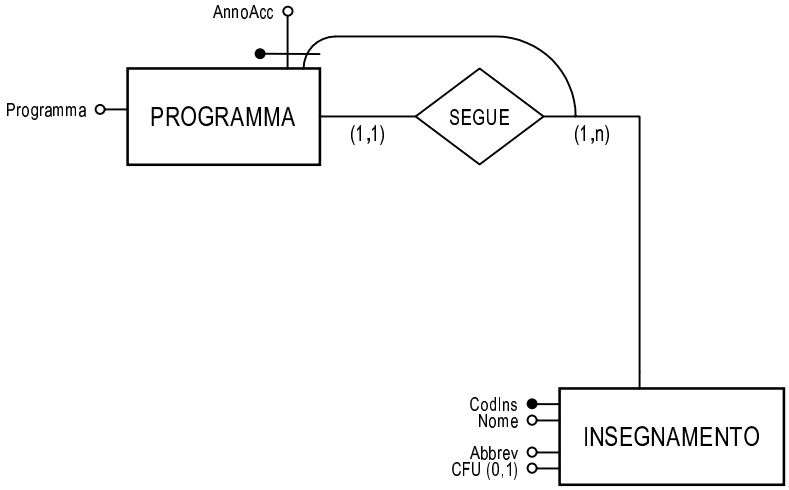


Figura 2.5: Entità PROGRAMMA

Il materiale didattico

Come supporto all'attività didattica, il docente ha spesso la necessità di fornire ai propri studenti materiale didattico di vario genere. Per modellare questo aspetto, viene creata un'entità MATERIALE_DIDATTICO. Gli attributi che caratterizzano l'entità sono i seguenti:

- *CodMat*: codice numerico univoco che svolge il compito di identificatore.

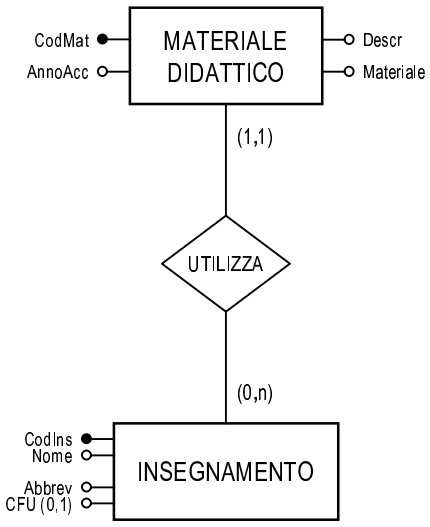


Figura 2.6: Entità MATERIALE_DIDATTICO

- *AnnoAcc*: indica l'anno accademico in cui il materiale viene introdotto.
- *Descr*: breve descrizione del contenuto.
- *Materiale* memorizza il riferimento al file esterno che costituisce il materiale didattico vero e proprio.

L'entità viene collegata ad INSEGNAMENTO per mezzo dell'associazione UTILIZZA. Dato che un insegnamento può avvalersi di zero o più elementi di materiale didattico, mentre ogni istanza di materiale didattico si riferisce ad un solo insegnamento, l'associazione è di tipo uno a molti, con partecipazione opzionale per INSEGNAMENTO.

La propedeuticità

Per gestire la propedeuticità fra gli insegnamenti è sufficiente fare le seguenti considerazioni. Uno stesso insegnamento può essere propedeutico per più insegnamenti susseguenti; viceversa, un insegnamento può richiedere la propedeuticità di una o più materie.

La soluzione più naturale per memorizzare le propedeuticità è quella di introdurre un anello sull'entità INSEGNAMENTO, con le etichette *richiede* e *richiesta* ad indicare il diverso ruolo degli insegnamenti nell'associazione. Le cardinalità per entrambi i ruoli vengono poste a (0,n).

La figura 2.7 mostra l'associazione PROPED appena definita.

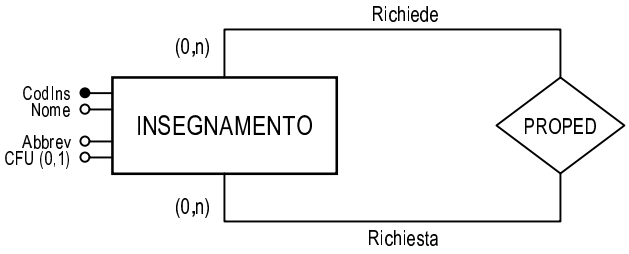


Figura 2.7: Associazione PROPED

Gli avvisi riguardanti un insegnamento

Uno dei principali motivi per la realizzazione di sezioni del sito dedicate ai singoli insegnamenti è la volontà di costituire un efficace mezzo per lo scambio di informazioni fra il docente e gli studenti. In particolare, sulla pagina del proprio insegnamento il docente potrà pubblicare comunicazioni ed avvisi che riguardano l'insegnamento stesso (cambiamenti nell'orario di lezione, avvisi relativi alle verifiche, scadenze di consegna per elaborati o tesine, . . .).

Per memorizzare tali informazioni si rende necessaria la creazione di un'entità AVVISI_INS. Gli attributi sono del tutto analoghi a quelli visti per l'entità AVVISI_CDS (pag. 24); si hanno pertanto un identificatore *CodAvvIns* e gli attributi *Data*, *DataScad*, *Descr* e *Testo*. Rispetto a AVVISI_CDS manca l'attributo *Autore*, non necessario in questo caso.

AVVISI_INS (figura 2.8) è collegata a CORSO_DI_STUDIO dall'associazione INS_AVV, con cardinalità (1,1) per AVVISI_INS e (0,n) per INSEGNAMENTO.

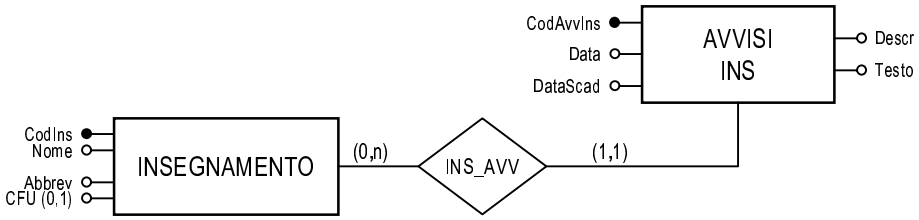


Figura 2.8: Entità AVVISI_INS

Con l'esame di quest'ultimo aspetto, il raffinamento della parte relativa ad INSEGNAMENTO può dirsi concluso. In figura 2.9 si può vedere il sottoschema risultante.

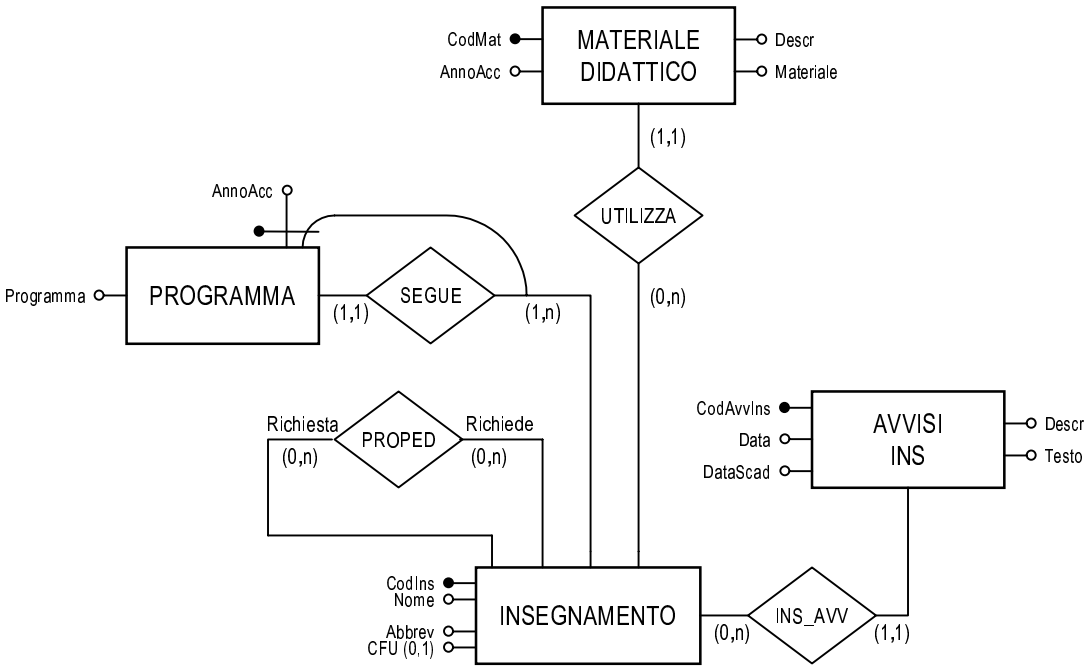


Figura 2.9: Sottoschema di INSEGNAMENTO

2.1.4 Raffinamento dell'entità *Esami_Lezioni*

L'attenzione si sposta ora sulla gestione dell'orario delle lezioni e degli appelli d'esame. Dopo una prima analisi dei requisiti da soddisfare, si intuisce che la costruzione del sottoschema associato al concetto ESAMI_LEZIONI richiede l'introduzione delle seguenti entità:

- LEZIONE
- APPELLO
- SESSIONE
- AULA

La figura 2.10 illustra la struttura del sottoschema che si vuole definire, comprensiva delle associazioni che legano le entità. Per ragioni di chiarezza è opportuno rappresentare anche l'entità INSEGNAMENTO; si noti come l'associazione ASSOCIATI_A, che nello schema scheletro lega INSEGNAMENTO ed ESAMI_LEZIONI, sia stata ridefinita con le due associazioni DI e PREVEDE.

Le entità sopra elencate vengono ora esaminate nel dettaglio.

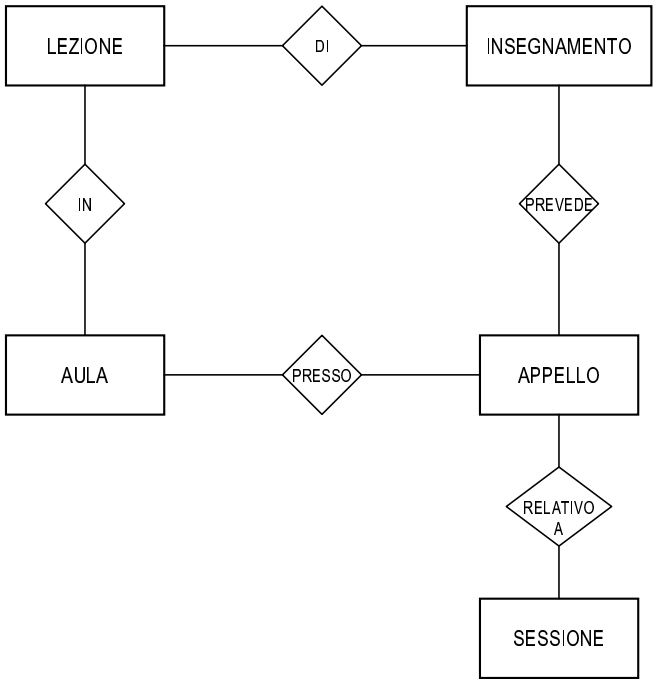


Figura 2.10: Struttura del sottoschema di ESAMI_LEZIONI

Lezione

Questa entità è popolata dall'insieme delle ore di lezione che si tengono nell'arco del ciclo didattico corrente. E' sufficiente tenere conto delle lezioni di una settimana, dato che lo schema settimanale si ripete.

Ogni istanza rappresenta una singola ora di lezione riferita ad un singolo insegnamento, ed è caratterizzata dagli attributi *Giorno* e *Ora*, che rappresentano rispettivamente il giorno della settimana e l'ora in cui si svolge la lezione.

Appello

L'appello è il concetto centrale per la gestione delle verifiche e degli esami. Gli attributi *DataApp* e *OraApp* memorizzano la data e l'ora in cui si svolge un appello. L'attributo *Tipo* indica se la verifica è di tipo scritto od orale.

Sessione

Per completare le informazioni relative agli esami, bisogna tenere conto delle sessioni in cui gli appelli si svolgono. La soluzione più appropriata è quella di introdurre un'entità SESSIONE, collegata ad APPELLO da un'associazione RELATIVO_A. Gli attributi dell'entità sono il nome della sessione (*NomeSess*) e le date di inizio e fine (*DataInizio* e *DataFine*).

La soluzione alternativa di modellare i dati relativi alla sessione come attributi di APPELLO è stata ritenuta poco efficiente. Il fatto che molti appelli appartengano ad una stessa sessione, infatti, avrebbe portato ad un'inutile ridondanza nei valori di questi attributi.

Aula

L'interesse per la aule in questo schema è in funzione del loro utilizzo in occasione di lezioni od esami.

Per definire l'entità AULA occorre tenere presente la situazione reale. Oltre alle dieci aule della nuova sede, la Facoltà di Ingegneria continuerà ad utilizzare, per le sue attività didattiche, aule di altri dipartimenti. Le aule da gestire saranno quindi divise tra vari edifici, per cui e' necessario memorizzare il dipartimento di appartenenza. Oltre all'attributo *NomeAula*, che rappresenta la lettera o il numero che caratterizza l'aula, si introduce quindi l'attributo *Sede*. E' opportuno inoltre definire l'attributo *Capienza*, ossia il massimo numero di posti presenti nell'aula.

In figura 2.11 si possono vedere gli attributi introdotti.

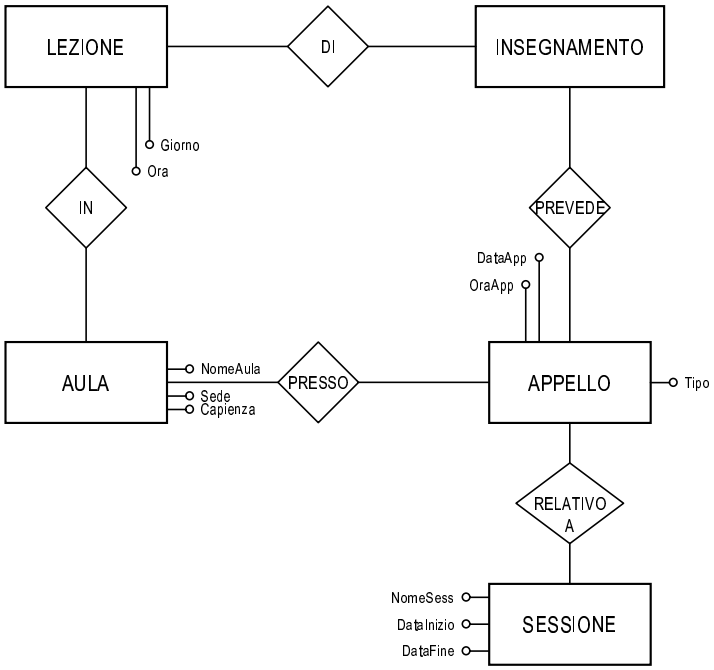


Figura 2.11: Attributi di LEZIONE, APPELLO, AULA, SESSIONE

Scelta degli identificatori

Occorre ora procedere alla scelta degli identificatori per le entità sopra descritte, effettuando contestualmente alcune riflessioni sulle associazioni presenti nel sottoschema.

Innanzitutto, per identificare AULA si inserisce un codice univoco, *CodAula*. Questo attributo non è strettamente necessario, dato che un'aula può essere identificata dalla combinazione di *Nomeaula* e *Dipartimento*, ma rappresenta una soluzione più pratica rispetto a quest'ultima.

Consideriamo ora l'entità SESSIONE. Dal momento che non interessa mantenere la serie storica delle sessioni d'esame, ma soltanto le sessioni dell'anno accademico in corso, non succederà mai che due istanze dell'entità abbiano lo stesso nome. E' quindi ragionevole utilizzare come identificatore l'attributo *NomeSess*.

Il discorso per l'entità LEZIONE è un po' più articolato. Come spiegato in precedenza, un'istanza di questa entità rappresenta un'ora di lezione settimanale riferita ad un particolare insegnamento.

Ogni ora di lezione è associata all'insegnamento relativo tramite l'associazione DI. Una lezione deve appartenere ad un solo insegnamento: di qui la partecipazione con cardinalità (1,1). D'altra parte, per uno stesso insegnamento si terranno tipicamente più ore di lezione settimanali, ma esistono anche insegnamenti che non partecipano all'associazione perchè non appartenenti al periodo didattico corrente. La cardinalità di INSEGNAMENTO è quindi (0,n).

L'associazione IN esprime invece il collegamento tra un'ora di lezione e l'aula nella quale viene effettuata. Analogamente a DI, una lezione è obbligatoriamente legata ad una sola aula, mentre un'aula può ospitare zero o più lezioni. Le cardinalità saranno quindi (1,1) e (0,n), rispettivamente.

A questo punto, per la scelta dell'identificatore di LEZIONEsi presentano due possibilità (figura 2.12), entrambe di tipo composto e *mixed*, ognuna delle quali serve ad imporre un determinato vincolo:

1. Identificatore formato dagli attributi *Giorno* ed *Ora* unitamente all'associazione DI: questa soluzione non permette che una stessa lezione si tenga contemporaneamente in due aule diverse.

2. Identificatore formato dagli attributi *Giorno* ed *Ora* insieme all'associazione IN: questa soluzione esprime il vincolo che impedisce la contemporanea presenza di due lezioni nella stessa aula.

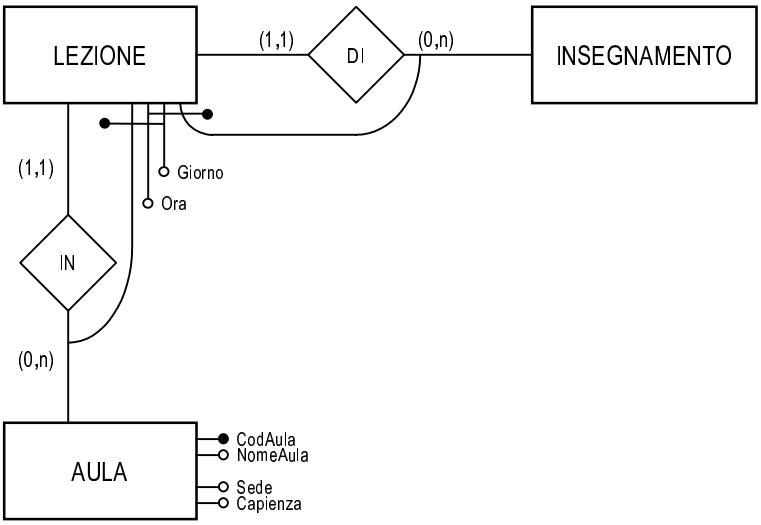


Figura 2.12: Identificatori dell'entità LEZIONE

Dal momento che le due soluzioni esprimono dei vincoli altrettanto importanti e necessari, si decide di non optare in questo momento per l'una o per

l'altra. Nello schema rimangono quindi indicati entrambi gli identificatori; nella successiva fase di progetto logico, analizzata nella sezione 2.2, uno dei due verrà eletto a chiave primaria, mentre l'altro costituirà una chiave alternativa.

Da ultimo, rimane da considerare l'entità APPELLO. Essa è legata ad INSEGNAMENTO e ad AULA per mezzo di due associazioni uno a molti, PREVEDE e PRESSO; il ragionamento sulle cardinalità è del tutto analogo a quello condotto per le associazioni dell'entità LEZIONE.

L'analogia con tale entità è ravvisabile anche nella scelta dell'identificatore, anche se in questo caso l'esito è leggermente diverso. Anche per APPELLO, infatti, si presentano due possibilità:

1. Identificatore formato dagli attributi *DataApp* ed *OraApp* insieme all'associazione PREVEDE.

2. Identificatore formato dagli attributi *DataApp* ed *OraApp* unitamente all'associazione PRESSO.

Tuttavia, analizzando la situazione con maggiore cura, si nota che il vincolo inposto dalla seconda alternativa in questo caso è indesiderabile. Può infatti succedere che appelli relativi ad insegnamenti diversi vengano fatti coincidere ed usufruiscano della stessa aula, soprattutto nel caso di verifiche scritte: è il caso di insegnamenti tenuti dallo stesso docente, oppure di appelli relativi ad insegnamenti analoghi, ma appartenenti a corsi di studio diversi.

Per questa ragione, la scelta dell'identificatore ricade sulla prima possibilità esposta, mentre la seconda viene scartata.

Per concludere, la figura 2.13 riproduce il sottoschema ottenuto.

2.1.5 Raffinamento dell'entità *Studente*

Di tutti i possibili dati relativi agli studenti, l'insieme delle informazioni che un database di questo genere ha interesse a conservare ne costituisce soltanto una parte. Per gli scopi dell'applicazione, ad esempio, non occorre gestire le informazioni relative alle immatricolazioni ed alle iscrizioni (essendo questo un compito spettante alla segreteria studenti), e non è neppure necessaria una schedatura esaustiva dei dati anagrafici.

Gli elementi che interessa memorizzare sono stati individuati nella seguente lista di attributi:

- *Matr*: matricola dello studente, e rappresenta l'identificatore dell'entità

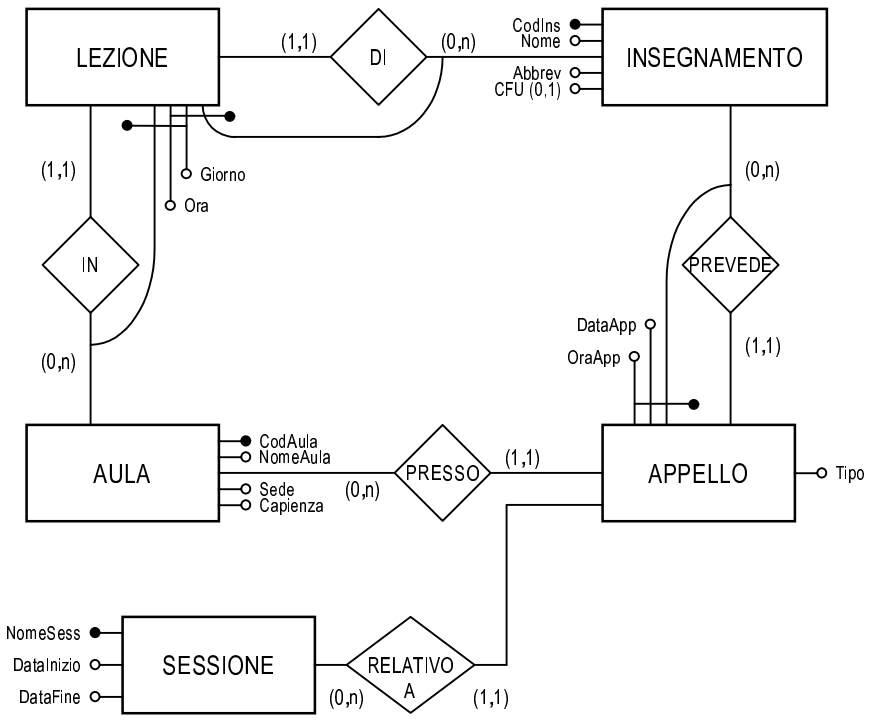


Figura 2.13: Sottoschema di ESAMI_LEZIONI

- *Cognome*
- *Nome*
- *E-Mail*
- *Username*
- *Password*

Le informazioni di autenticazione *Username* e *Password* vengono assegnate agli studenti per un duplice scopo. Il loro utilizzo permette sia l'accesso ad aree riservate del sito (per limitare ad esempio l'accesso al materiale didattico ai soli studenti interessati), sia l'accesso ai laboratori di informatica. L'assegnazione un'unica coppia username/password per ogni studente, oltre a costituire una maggiore comodità per gli studenti stessi, semplifica la gestione delle informazioni.

L'attributo *E-Mail* memorizza l'indirizzo di posta elettronica che verrà assegnato ad ogni studente dell'Ateneo, in base ad un progetto in fase di attuazione, e può essere usato per far pervenire agli studenti comunicazioni mirate, mediante

la creazione di apposite *mailing lists*.

Tra gli studenti sono annoverati anche i rappresentanti, delegati a partecipare ai vari organi istituzionali secondo le modalità previste dallo Statuto. I rappresentanti, inoltre, devono avere la possibilità di pubblicare avvisi sul sito, nello spazio dedicato agli studenti. Diventa quindi necessario poter discriminare fra i rappresentanti e gli studenti; tale distinzione viene effettuata specializzando l'entità STUDENTE con il *subset* RAPPRESENT.

La nuova entità introduce un unico attributo specifico, *Lista*, che identifica la lista per la quale il rappresentante è stato eletto.

Viene inoltre definita un'entità AVVISI_RAPP, collegata a RAPPRESENT dall'associazione RAPP_AVV. In stretta analogia con quanto visto per AVVISI_INS, l'entità è caratterizzata da un identificatore *CodAvvRapp* e dagli attributi *Data*, *DataScad*, *Descr* e *Testo*.

In figura 2.14 si può vedere il sottoschema appena definito.

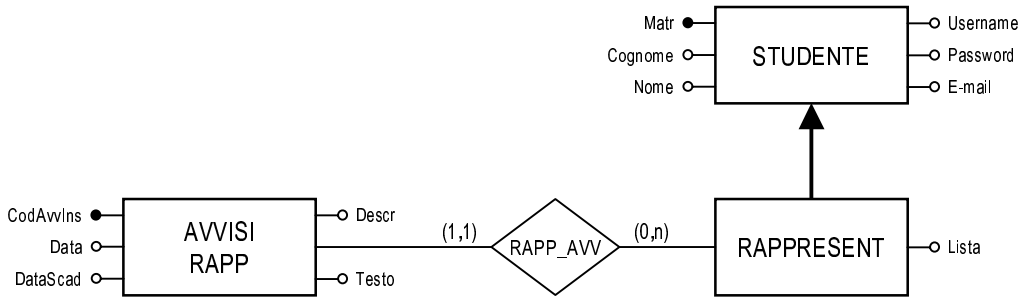


Figura 2.14: Sottoschema di STUDENTE

2.1.6 Raffinamento dell'entità *Facoltà*

Questa parte dello schema deve servire a rappresentare le informazioni di carattere generale, che non sono riconducibili ai singoli corsi di studio od ai singoli insegnamenti.

Osservando i requisiti globali del progetto, si nota che è presente una serie di informazioni di difficile collocazione. Tra queste, ad esempio, ci sono i dati generali di presentazione della Facoltà, il Regolamento di Facoltà, o il calendario

dell'anno accademico che illustra i periodi di lezione e le sessioni di esami. Una scelta possibile potrebbe essere quella di escluderle dal database e di mantenerle all'interno di pagine web statiche. Si è tuttavia deciso di inserire nello schema anche questi dati, per una migliore coerenza del progetto.

Per le ragioni sopra esposte, ha senso definire un'entità FACOLTÀ, che raccolga tramite i suoi attributi questo genere di informazioni. Si nota subito come FACOLTÀ sia un'entità anomala, essendo popolata da una sola instanza (non interessa infatti memorizzare i dati di altre Facoltà). Per evidenziare questa peculiarità, FACOLTÀ verrà rappresentata con un rettangolo dagli angoli arrotondati (figura 2.15.

La presenza di questa entità ha inoltre il compito di migliorare la comprensione dello schema concettuale, evidenziando in maniera più chiara le relazioni tra le diverse sezioni.

Il prossimo passo è quello di definire meglio l'entità FACOLTÀ, descrivendo gli attributi che la caratterizzano:

- *Presentazione_I*: testo di presentazione della Facoltà in versione italiana
- *Presentazione_E*: testo di presentazione in versione inglese dello stesso testo
- *CalendarioAA_I*: calendario dell'anno accademico, con le date che scandiscono lo svolgersi dell'attività universitaria
- *CalendarioAA_E*: idem, in versione inglese
- *UbicazioneOrari_I*: riporta le informazioni generiche quali l'indirizzo della sede, gli orari di apertura, i numeri di telefono ecc.
- *UbicazioneOrari_E*: idem, in versione inglese
- *RegolFacoltà*: il Regolamento di Facoltà consultabile online.

Si fa notare che tutti i documenti rappresentati dagli attributi sopra descritti vengono memorizzati in file esterni, referenziati dagli attributi per mezzo del nome file.

Non si è inoltre proceduto alla scelta di un identificatore; tale operazione non è necessaria, essendo l'entità costituita da un'unica istanza.

Per completezza, occorre memorizzare alcune informazioni relative ai dipartimenti che afferiscono alla Facoltà. Dal momento che i dipartimenti (attualmente ne esiste uno solo, anche se a breve diventeranno tre) saranno dotati di web autonomi, tutto ciò che interessa per la nostra applicazione è il nome del dipartimento e la URL del corrispondente sito. Gli attributi dell'entità DIPARTIMENTO saranno

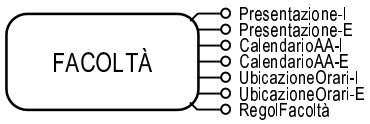


Figura 2.15: Entità FACOLTÀ

quindi *NomeDip* e SITOWEB, oltre ad un codice *CodDip* introdotto in qualità di identificatore.

FACOLTÀ e DIPARTIMENTO sono legati dall'associazione ORGANIZZATA_IN (figura 2.16). La Facoltà deve essere associata almeno ad un dipartimento, senza che sia stabilito un limite superiore; la partecipazione ad ORGANIZZATA_IN avviene quindi con cardinalità (1,n). Un dipartimento, com'è ovvio, è obbligatoriamente associato all'unica istanza di FACOLTÀ, per cui card(DIPARTIMENTO, ORGANIZZATA_IN) = (1,1).

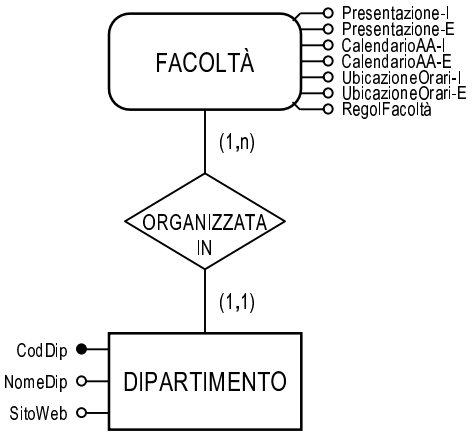


Figura 2.16: Entità DIPARTIMENTO

Passando ad aspetti maggiormente attinenti al settore didattico, occorre memorizzare le informazioni relative alle sessioni per gli esami di laurea e per gli esami di abilitazione professionale. Vengono cosìintrodotte due nuove entità, ESAME_LAUREA ed ESAME_STATO, mostrate in figura 2.17.

La prima è caratterizzata da due attributi: *Datalaurea* che riporta la data della seduta di laurea, e *Sessione*, che memorizza il nome della sessione a cui appartiene l'esame di laurea. Supponendo che vengano mantenute soltanto le istanze

degli esami di laurea relativi all'anno accademico corrente, è corretto scegliere *DataLaurea* come identificatore.

ESAME_STATO presenta invece l'unico attributo *DataEsame*, che funge anche da identificatore.

Queste entità sono collegate a FACOLTÀ dalle associazioni uno a molti FAC_LAUREA e FAC_ESAME. La scelta delle cardinalità è abbastanza ovvia; si vuole soltanto osservare che la cardinalità minima uguale a zero per FACOLTÀ implica che le due entità ESAME_LAUREA ed ESAME_STATO possano in certi momenti essere vuote (ad esempio nei periodi tra un esame e l'altro).

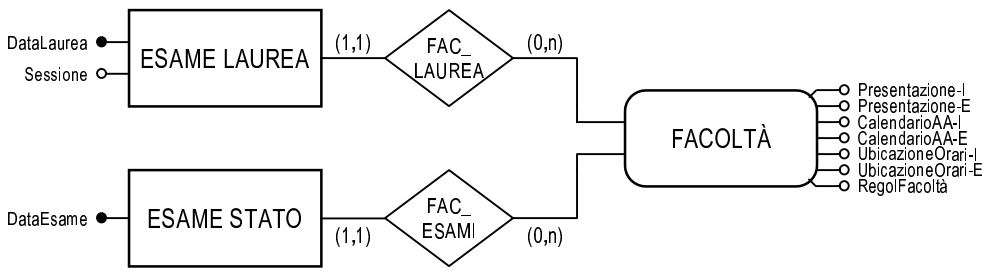


Figura 2.17: Entità ESAME_LAUREA ed ESAME_STATO

L'ultimo aspetto da considerare è legato alle notizie e agli avvisi che riguardano la Facoltà, e che vengono richiamati nella home page. Per memorizzare questi avvisi, è necessaria un'entità AVVISI_FAC. Gli attributi sono analoghi a quelli visti per AVVISI_CDS, con la differenza che in questo caso è richiesto che le notizie siano in versione bilingue. Si avranno quindi l'identificatore *CodAvvFac*, unitamente agli attributi *Data*, *DataScad*, *Autore*, *Descr_I*, *Descr_E*, *Testo_I* e *Testo_E*.

L'entità è legata a FACOLTÀ attraverso l'associazione FAC_AVV, del tutto simile alle associazioni analoghe viste in precedenza.

In figura 2.18 si può vedere il sottoschema completo.

2.1.7 Sottoschema di *Docente*

Apprestandoci ad esaminare la gestione delle informazioni che riguardano i docenti, appare subito evidente come non sia opportuno prescindere da una modellazione di più ampia portata che riguardi le diverse categorie di personale che hanno a che fare con la Facoltà. Se da una parte si è in qualche modo costretti a dedicare

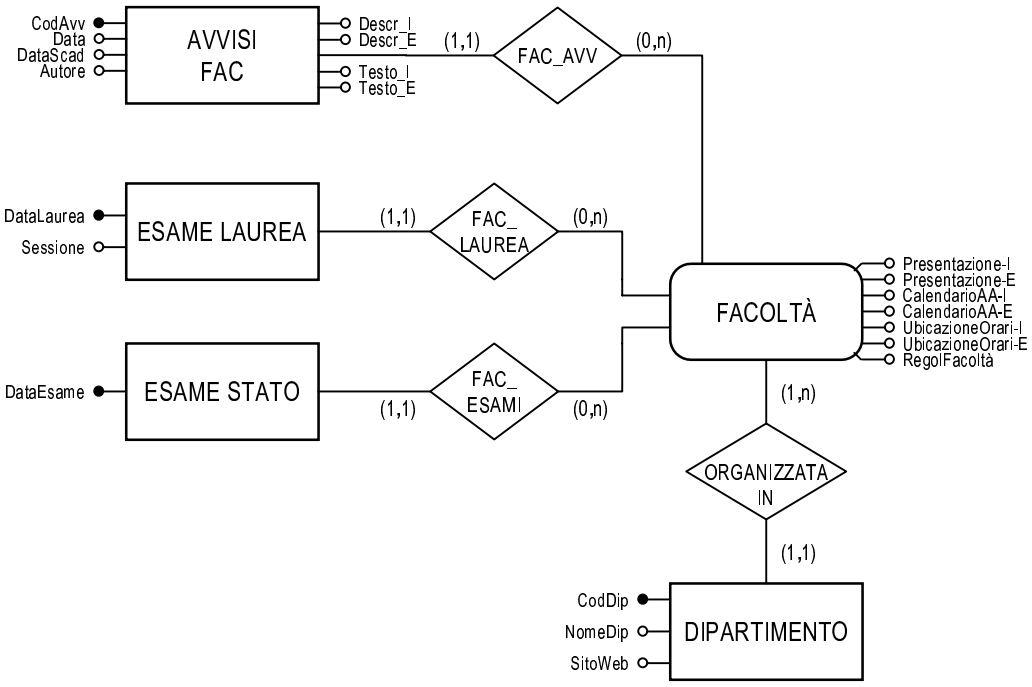


Figura 2.18: Sottoschema di FACOLTÀ

attenzione anche ad aspetti di scarsa attinenza con l'ambito specifico di questo lavoro di tesi, dall'altra occorre notare che una modellazione accurata e coerente del personale della Facoltà va a tutto vantaggio dello schema concettuale globale.

Le informazioni riguardanti i docenti verranno quindi gestite all'interno di un ambito più ampio che riguarda tutto il personale della Facoltà.

Per cominciare, viene definita un'entità denominata PERSONALE. Ad essa vengono associate le informazioni comuni a tutte le categorie di personale.

Applicando la primitiva *top-down* per la generazione di una gerarchia di generalizzazione, il personale viene suddiviso in tre categorie (2.19):

- il personale cosiddetto *interno* comprende tutto il personale alle dirette dipandenze della Facoltà
- La categoria *Docente Esterno* raggruppa i docenti non appartenenti alla Facoltà con l'incarico di gestire temporaneamente uno o più insegnamenti.
- La categoria *Altro Personale* racchiude il personale non riconducibile alle prime due suddivisioni.

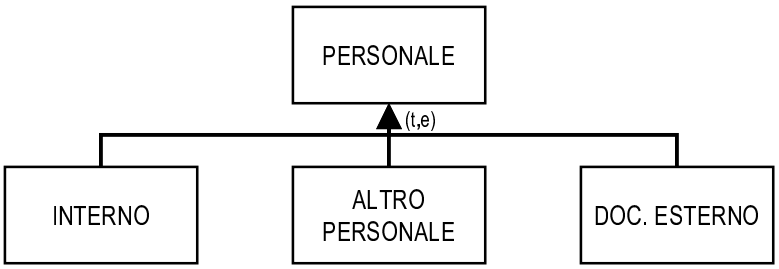


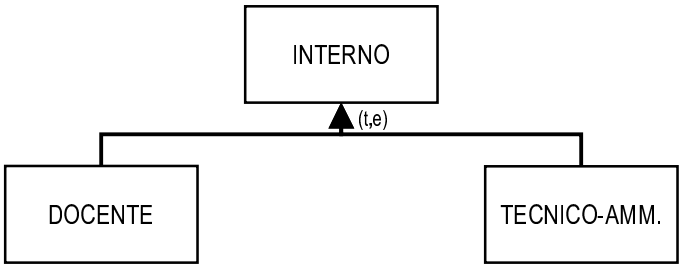
Figura 2.19: Specializzazione di PERSONALE

La gerarchia, essendo totale ed esclusiva, classifica tutto il personale avente un qualche genere di rapporto con la Facoltà e non prevede che un membro del personale possa appartenere a più categorie contemporaneamente.

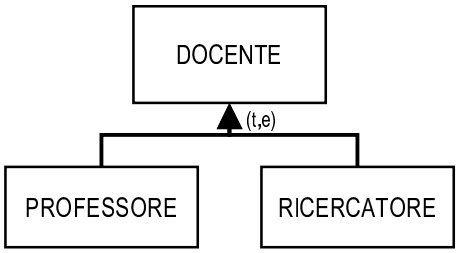
La classificazione può essere ulteriormente raffinata allo scopo di rappresentare con maggiore dettaglio le categorie di personale individuate.

Specializzazione del personale interno

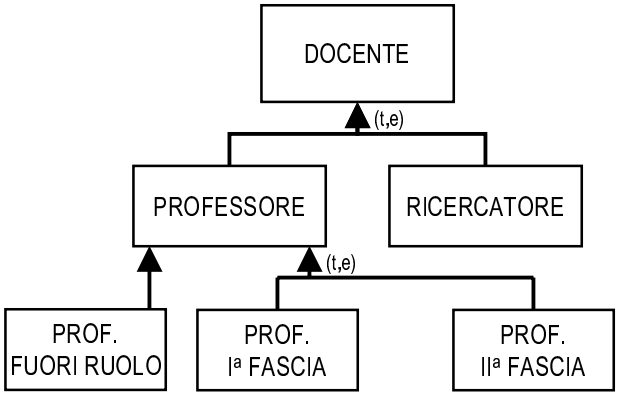
Si può operare una prima suddivisione fra il personale docente ed il personale tecnico-amministrativo; l'entità INTERNO viene quindi specializzata mediante l'introduzione di una gerarchia, di tipo totale ed esclusivo, formata dalle entità DOCENTE e TECNICO_AMM.



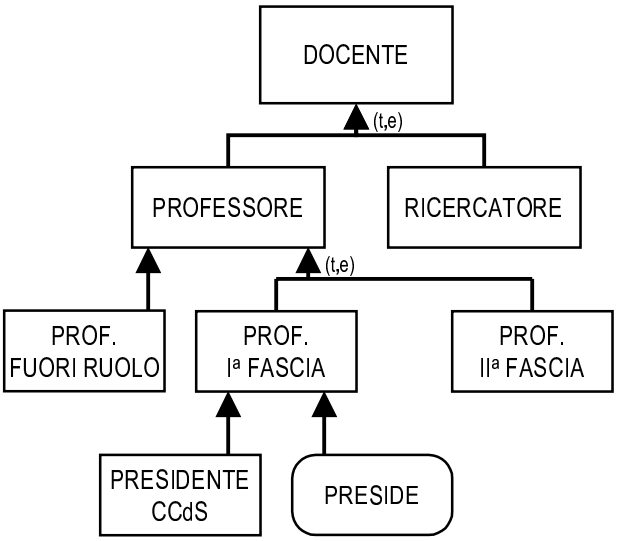
Proseguendo ora con l'analisi del personale docente, si effettua un'ulteriore distinzione fra professori e ricercatori tramite la definizione di una nuova gerarchia totale ed esclusiva, con le entità PROFESSORE e RICERCATORE.



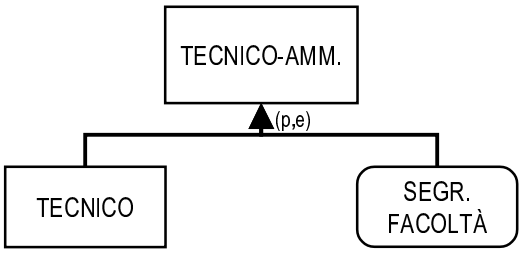
L'entità PROFESSORE viene ancora specializzata in PROF_II^a_FASCIA e PROF_II^a_FASCIA per mezzo di un' altra gerarchia, anche in questo caso di tipo totale ed esclusivo. Volendo poi evidenziare la presenza di professori fuori ruolo, i quali però possono appartenere sia alla I^a che alla II^a fascia, si introduce un subset dell'entità PROFESSORE.



Infine, si osserva che tra i professori di I^a fascia vengono eletti il Preside della Facoltà ed i Presidenti di Corso di Studio. Per esprimere queste informazioni conviene utilizzare due subset. si noti che il subset PRESIDE utilizza la notazione del rettangolo dai bordi arrotondati, poiché è popolato da un'unica istanza.



Per concludere l'analisi sul personale interno, rimane da esaminare il personale tecnico-amministrativo. Ai fini del progetto è necessario evidenziare le figure del tecnico e della segretaria di facoltà. Si specializza pertanto l'entità con l'utilizzo della seguente gerarchia:



La gerarchia è in questo caso parziale, perché esistono membri del personale tecnico-amministrativo che non rientrano nella classificazione introdotta, ed esclusiva.

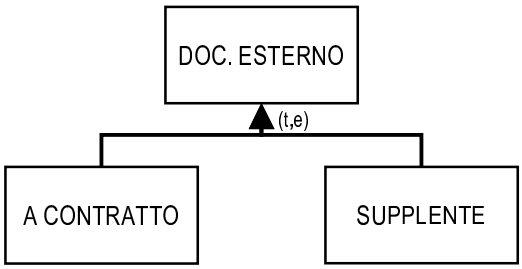
Specializzazione dei docenti esterni

I docenti che abbiamo denominato *esterni* possono essere di due tipi: docenti a contratto e docenti supplenti.

I docenti a contratto sono persone non appartenenti al mondo accademico, e che vengono incaricate, per mezzo di un apposito contratto, di tenere un insegnamento.

I docenti supplenti sono docenti chiamati da altre Facoltà o ad altri atenei, a cui viene affidato un insegnamento presso la Facoltà.

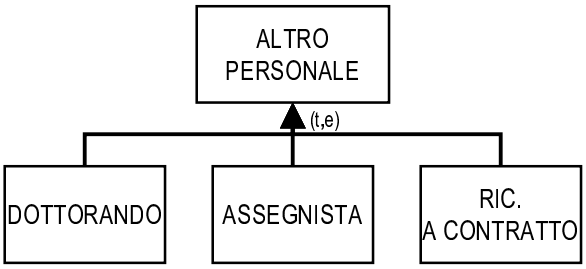
La gerarchia introdotta è quindi la seguente:



Specializzazione dell'entità *Altro_Personale*

Si definisce una gerarchia di generalizzazione che distingue fra le seguenti figure:

- Dottorandi
- Assegnisti
- Ricercatori a contratto



Queste tre categorie sono caratterizzate da contratti a tempo determinato e non sono assimilabili al personale direttamente dipendente dalla Facoltà.

Definizione degli attributi

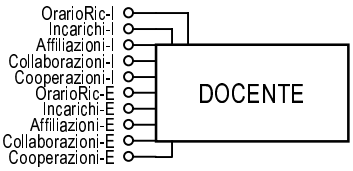
Si procede ora con l'ultimo raffinamento necessario per completare la modellazione della gerarchia del personale, introducendo gli attributi delle entità.

- *CF*: codice fiscale
- *Cognome*

- *Nome*
- *Username*: codice utente univoco, per l'identificazione da parte del sistema
- *Password*: da utilizzare assieme allo username
- *Ufficio*: informazioni sulla localizzazione dell'ufficio
- *Tel*: numero telefonico dell'ufficio
- *Fax*: numero di fax dell'ufficio
- *E-Mail*: indirizzo di posta elettronica
- *Foto*: fotografia (facoltativa) da inserire nella pagina personale
- *PagPers*: URL dell'eventuale pagina gestita a discrezione degli interessati, da visualizzare nella pagina personale "ufficiale" fornita dal sito
- *DataInizio*: data di inizio del rapporto di lavoro
- *DataFine(0,1)*: presente nel caso di contratti a tempo determinato, indica il termine previsto dal contratto.

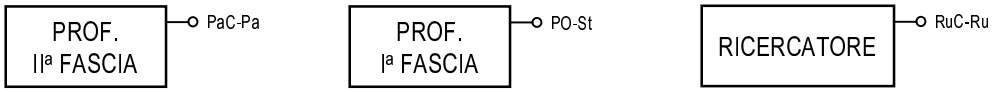
Si nota che sono evidenziati due identificatori. E' infatti possibile usare sia il codice fiscale che lo username, essendo entrambi univoci. Lo username, essendo normalmente generato a partire dal nome e dal cognome della persona che lo possiede, è di utilizzo più intuitivo rispetto al codice fiscale. La scelta definitiva tra le due possibilità viene rimandata alla fase di progetto logico .

- *Ti_Td*: attributo selettore che indica se il componente del personale è stato assunto a tempo indeterminato oppure a tempo determinato.



Tutti gli attributi di questa entità sono presenti in doppia versione, italiano (suffisso *-I*) e inglese (suffisso *-E*).

- *OrarioRic*: breve testo con l'indicazione dell'orario di ricevimento
- *Incarichi, Affiliazioni, Collaborazioni, Cooperazioni*: informazioni facoltative sulle diverse attività del docente.



Le entità PROF_I^a_FASCIA, PROF_II^a_FASCIA e RICERCATORE posseggono ognuna un attributo selettore, che permette di distinguere i diversi livelli di qualifica come illustrato di seguito:

- *PO-St*
  PO = Professore ordinario
  St = Professore straordinario
- *PaC-Pa*
  PaC = Professore associato confermato
  Pa = Professore associato non confermato
- *RuC-Ru*
  RuC = Ricercatore confermato
  Ru = Ricercatore non confermato

Gli attributi definiti per DOC_ESTERNO rispecchiano quelli visti per DOCENTE.

Per concludere, il sottoschema finale è rappresentato in figura 2.20.

2.1.8 Integrazione dei sottoschemi

In quest'ultima fase si procede alla connessione dei sottoschemi definiti nelle pagine precedenti, per arrivare alla produzione dello schema E/R completo.

L'integrazione avviene seguendo lo schema scheletro impostato inizialmente, e richiede ulteriori raffinamenti allo scopo di completare la definizione delle associazioni che collegano i sottoschemi.

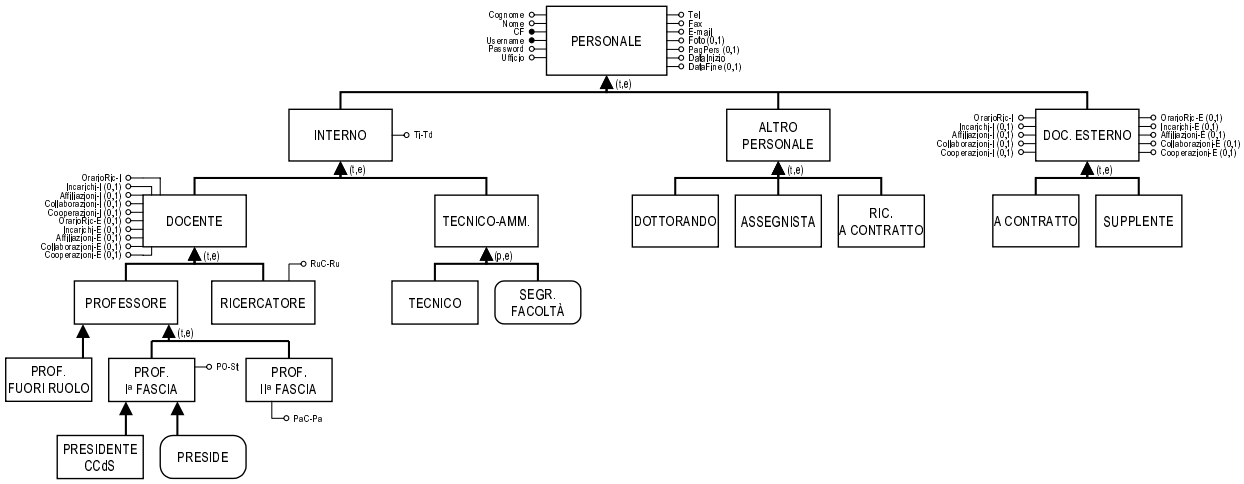


Figura 2.20: Sottoschema di PERSONALE

Raffinamento dell'associazione *Inserito_in*

L'associazione INSERITO_IN (figura 2.21)mette in relazione gli insegnamenti ai corsi di studio di appartenenza. Grazie ad essa è quindi possibile memorizzare i manifesti degli studi dei vari corsi.

Per poter svolgere questa funzione, è necessario definire i seguenti attributi:

- *AnnoAcc*: anno accademico del manifesto degli studi in cui la materia è presente. Grazie a questo attributo è possibile mantenere la serie storica dei manifesti degli studi, inserendo via via nuovi gruppi di associazioni caratterizzati da un anno accademico differente.

- *AnnoCorso*: anno di corso nel quale l'insegnamento è presente.

- *Periodo*: periodo didattico in cui la materia è collocata. La denominazione dell'attributo rispecchia l'assetto didattico introdotto con il nuovo ordinamento degli studi; nel caso del vecchio ordinamento, indica il semestre di appartenenza.

- *Obb_Opz*: indica se l'insegnamento è obbligatorio oppure opzionale (per dirla in altri termini, se è un fondamentale o un complementare).

L'associazione è di tipo molti a molti. Infatti uno stesso insegnamento può essere comune a più corsi di studio; d'altra parte, un corso di studio contempla ovviamente più materie. Non è stato fissato un limite minimo preciso al numero di insegnamenti inseriti in un corso di studio Un corso deve comunque avere almeno

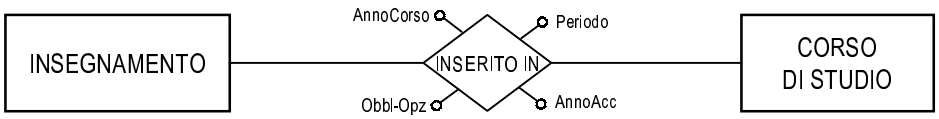


Figura 2.21: Raffinamento dell'associazione INSERITO_IN

un insegnamento (altrimenti la sua presenza non avrebbe senso), e ogni insegnamento deve appartenere ad almeno un corso di studio: entrambe le cardinalità minime sono quindi uguali ad 1.

Raffinamento dell'associazione *Dipende*

L'associazione DIPENDE, che collega i sottoschemi di FACOLTÀ e di PERSONALE, viene sdoppiata introducendo una nuova associazione *Afferisce_a*, che stabilisce il legame tra i membri del personale ed il dipartimento di appartenenza (figura 2.22).

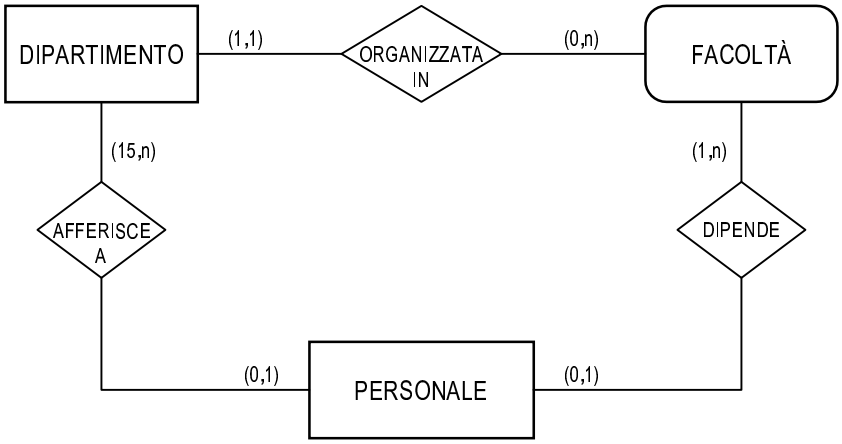


Figura 2.22: Raffinamento dell'associazione DIPENDE

L'associazione è facoltativa per PERSONALE, per tenere conto del fatto che esistono membri del personale (i docenti esterni, ad esempio) non appartenenti ad alcun dipartimento. Un membro del personale, inoltre, non può afferire a più dipartimenti. Quindi le cardinalità per PERSONALE sono (0,1).

Le cardinalità per le istanze di DIPARTIMENTO sono uguali a (15,n). La cardinalità minima esprime il vincolo (dedotto dallo Statuto di Ateneo [4]) secondo cui il numero minimo di docenti e ricercatori per la costituzione ed il mantenimento di un dipartimento è di 15 unità.

L'associazione DIPENDE, essendo del tipo uno a molti con partecipazione facoltativa per PERSONALE, presenta le cardinalità (0,1) verso PERSONALE e (1,n) verso FACOLTÀ.

Raffinamento dell'associazione *Insegna*

Come illustrato in figura 2.23, l'associazione INSEGNA viene sdoppiata nelle due associazioni INSEGNA_1 e INSEGNA_2, in modo da coinvolgere sia DOCENTE che DOC_ESTERNO.

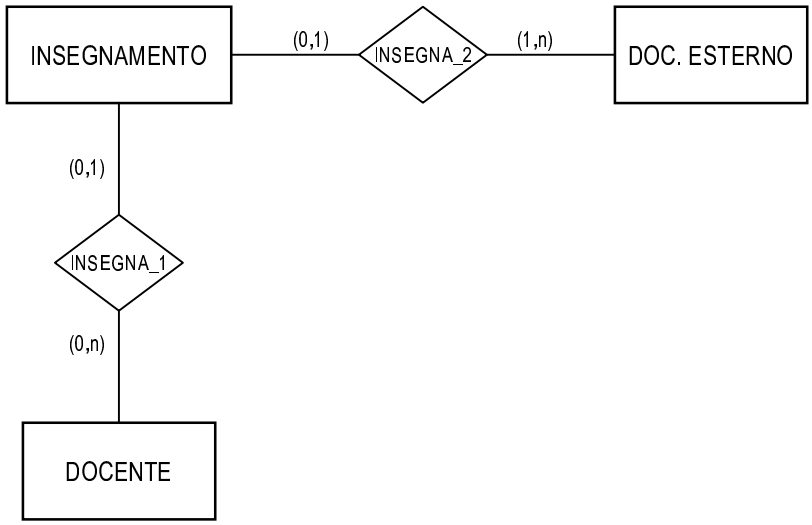


Figura 2.23: Raffinamento dell'associazione INSEGNA

Ogni insegnamento deve avere esattamente un docente titolare. Siccome l'associazione con il docente può avvenire sia tramite INSEGNA_1che tramite INSEGNA_2, INSEGNAMENTO partecipa ad entrambe le associazioni con cardinalità (0,1). Un docente può insegnare più materie, oppure essere privo di titolarità (come nel caso di alcuni ricercatori, oppure dei professori fuori ruolo): quindi card(DOCENTE, INSEGNA_1)=(0,n). L'associazione INSEGNA_2 è invece obbligatoria per DOC_ESTERNO, per cui card(DOCENTE, INSEGNA_1)=(1,n).

Raffinamento dell'associazione *Partecipa*

L'associazione PARTECIPA viene ridefinita per mezzo di due associazioni, ISCRIZIONE ed *Esito* (figura 2.24). La prima serve per modellare le liste d'esame: quando uno studente si iscrive per sostenere un appello od una verifica, viene creata una nuova associazione tra lo studente e l'appello scelto.

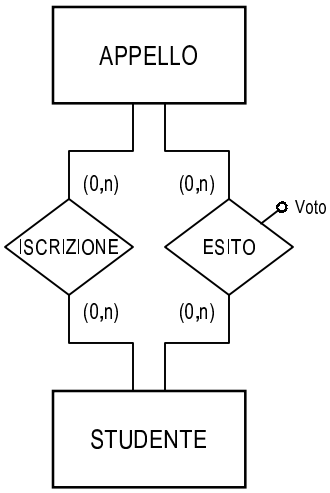


Figura 2.24: Raffinamento dell'associazione PARTECIPA

L'associazione ESITO memorizza invece i voti (tramite l'attributo *Voto*) ricevuti dagli studenti nelle verifiche la cui valutazione non è immediata (come nel caso delle prove scritte). Una volta che il docente ha concluso la valutazione delle verifiche e ha assegnato i voti, viene creato un nuovo insieme di associazioni.

Entrambe le associazioni sono di tipo molti a molti: uno studente può essere iscritto a zero o più appelli, e un appello può avere zero o più iscritti. La situazione per ESITO è analoga. Le cardinalità sono quindi tutte (0,n).

Introduzione della gerarchia *Avvisi*

Durante l'integrazione dei sottoschemi definiti in precedenza, si può notare che le entità AVVISI_CDS, AVVISI_INS AVVISI_FAC AVVISI_RAPP presentano fortissime analogie. Viene quindi naturale raggruppare tali entità in una gerarchia di generalizzazione totale ed esclusiva, introducendo la superentità AVVISI (figura 2.25).

Gli attributi comuni vengono fatti migrare sulla nuova entità. In particolare, il codice identificatore assume ora il più generico nome *CodAvv*. Gli altri attributi che si spostano su AVVISI sono *Data*, *DataScad*, *Descr_I* e *Testo_I*.

Introduzione dell'entità *Proposta_Tesi*

A questo punto rimane da considerare un'ultimo aspetto . E' necessario memorizzare le informazioni relative alle proposte di tesi effettuate dai docenti. A questo

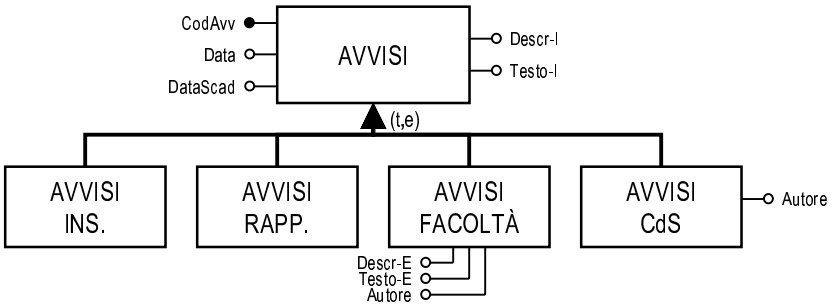


Figura 2.25: Creazione della gerarchia AVVISI

scopo viene introdotta un'entità PROPOSTA_TESI (figura 2.26), caratterizzata dai seguenti attributi:

- *CodProp*: codice univoco che funge da identificatore
- *Argomento*: argomento sul quale verte la proposta di tesi
- *Descr*: testo di descrizione
- *Immagine*: immagine facoltativa associata alla descrizione (memorizzata in un file esterno)
- *Durata*: tempo previsto (in mesi) di svolgimento della tesi
- *Tipo*: indica se la tesi ha carattere sperimentale oppure compilativo
- *DataInizio*: data prevista di inizio del lavoro di tesi (per progetti con una scadenza o non immediatamente disponibili)

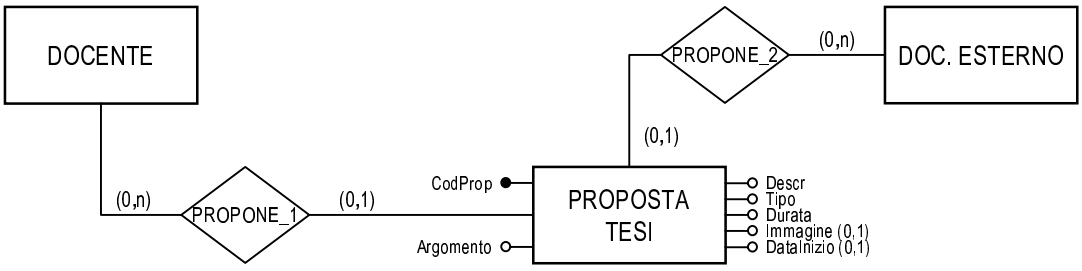


Figura 2.26: Entità PROPOSTA_TESI

Tra *Proposta_Tesi* e le entità DOCENTE e DOC_ESTERNO vengono stabilite le associazioni PROPONE_1 e PROPONE_2.

card(DOCENTE, PROPONE_1)=(0,n) perché un docente può proporre zero o più tesi.
card(PROPOSTA_TESI, PROPONE_1)=(0,1) perché una tesi è associata ad un solo docente (c'è lo 0 poiché l'associazione potrebbe avvenire tramite PROPONE_2).

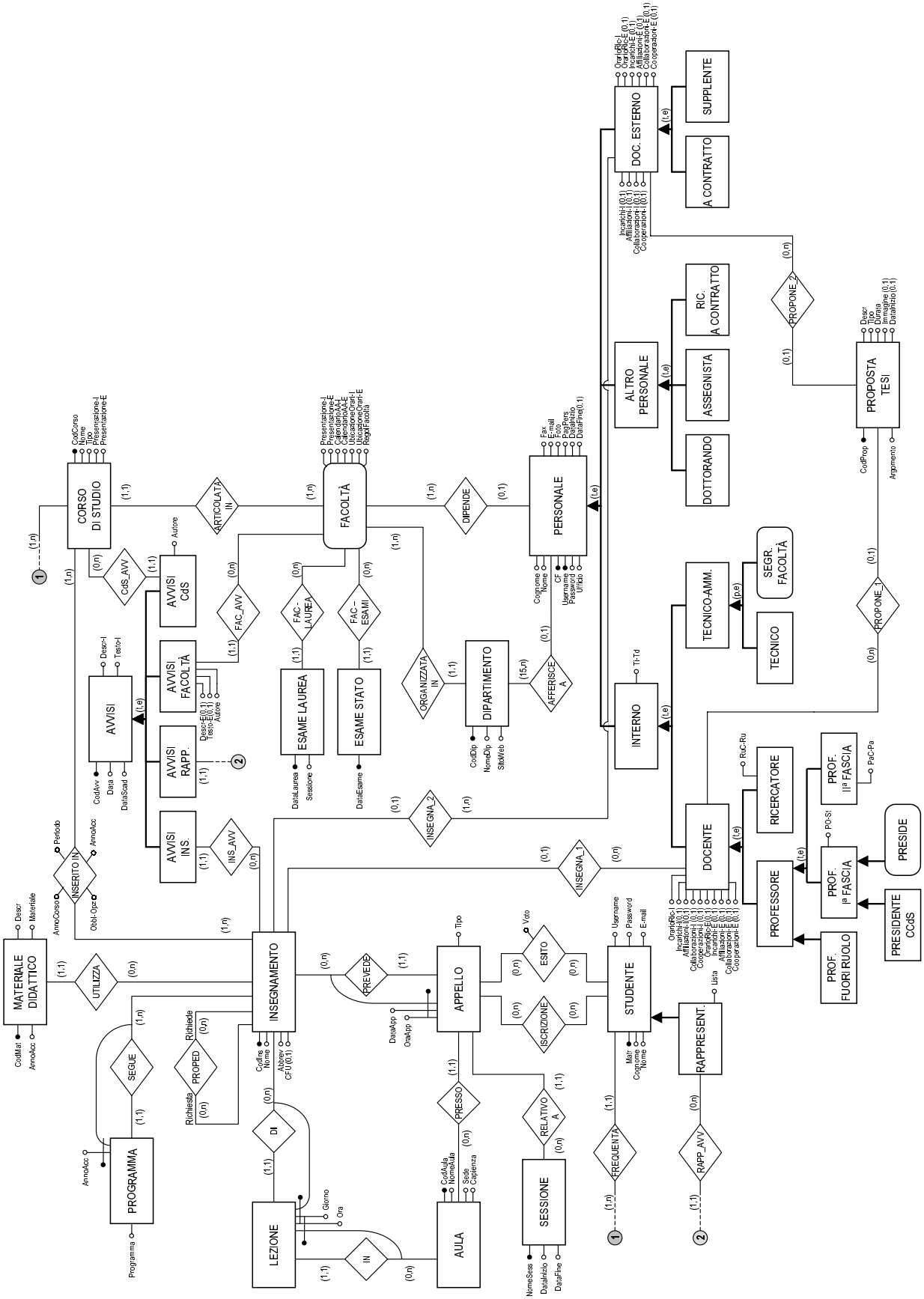
Le cardinalità per PROPONE_2 sono analoghe.

2.1.9 Schema E/R integrato

Al termine della fase di integrazione, la parte di progetto concettuale può dirsi conclusa. La figura della pagina seguente mostra lo schema E/R completo. Al fine di migliorare la leggibilità dello schema, per alcune associazioni dal percorso particolarmente lungo la connessione è sostituita da appositi cerchietti numerati corrispondenti.

E' stata effettuata inoltre l'integrazione tra lo schema prodotto in questo lavoro di tesi e lo schema elaborato nel lavoro di tesi parallelo [1]. Lo schema E/R globale, relativo al progetto nel suo complesso, è presentato nell'appendice A. È inoltre consultabile online al seguente indirizzo:

`http://sparc20.dsi.unimo.it/SitoFacolta/er.html`


CORSO DI STUDIO
ARTICOLATA IN
FACOLTÀ
DIPENDE
PERSONALE
ALTRO PERSONALE
ASSEGNISTA
DOTTORANDO
RIC. A CONTRATTO
DOC. ESTERNO
SUPPLENTE
A CONTRATTO
PROPONE_2
PROPOSTA TESI
CdS_AVV
AVVISI CdS
AVVISI
AVVISI FACOLTÀ
AVVISI RAPP.
AVVISI INS.
FAC_AVV
FAC-LAUREA
FAC-ESAMI
ESAME LAUREA
ESAME STATO
ORGANIZZATA IN
DIPARTIMENTO
AFFERISCE A
INSERITO IN
MATERIALE DIDATTICO
UTILIZZA
INSEGNAMENTO
PREVEDE
APPELLO
ESITO
ISCRIZIONE
STUDENTE
SEGUE
PROGRAMMA
PROPED
DI
LEZIONE
IN
PRESSO
AULA
RELATIVO A
SESSIONE
FREQUENTA
RAPPRESENT.
RAPP_AVV
INSEGNA_2
INSEGNA_1
INTERNO
TECNICO-AMM.
TECNICO
SEGR. FACOLTÀ
DOCENTE
RICERCATORE
PROFESSORE
PROF. I^a FASCIA
PROF. II^a FASCIA
PROF. FUORI RUOLO
PRESIDE
PRESIDENTE CCdS
PROPONE_1

2.2 Progetto logico

L'attività di progettazione logica è suddivisa in due fasi:

- **Ristrutturazione dello schema E/R**
  È indipendente dal modello logico e consiste in una semplificazione dello schema E/R, basata su criteri di ottimizzazione dello schema.

- **Progetto logico relazionale**
  Questa fase è riferita al modello logico relazionale e porta alla vera e propria trasformazione dello schema E/R semplificato nello schema relazionale.

2.2.1 Carico di lavoro

Prima di procedere,è importante effettuare una valutazione del carico di lavoro previsto sul database, considerando sia la dimensione dei dati da gestire che le caratteristiche delle operazioni che si stima saranno eseguite. Tale valutazione serve come ausilio in entrambe le fasi del procedimento di progettazione logica.

Volume dei dati

Per stimare il volume dei dati si deve tenere conto dei seguenti aspetti:

- numero medio di istanze di ogni entità ed associazione
- cardinalità e dimensioni di ciascun attributo
- percentuali di copertura delle gerarchie

Di seguito vengono esaminati i principali concetti presenti nello schema e vengono proposte le relative previsioni dei volumi. La lista completa delle entità e delle associazioni si trova nella tabella dei volumi a pag. 58. Ove possibile, le stime verranno effettuate tenendo in considerazione i cambiamenti che si verificheranno con l'introduzione del nuovo ordinamento degli studi.

Corso di Studio A regime, la Facoltà offrirà 6 corsi di laurea ed altrettanti corsi di laurea specialistica. Prevedendo un periodo di transizione durante il quale saranno attivati contemporaneamente corsi impostati secondo il vecchio e il nuovo ordinamento, i corsi di studio potrebbero diventare una ventina. Occorre inoltre considerare i corsi di specializzazione. Si prevedono quindi 25 istanze.

Insegnamento Possiamo considerare una media di 40 insegnamenti per corso di studio. Pertanto é ragionevole stimare 600 istanze, pur prevedendo che alcuni insegnamenti saranno comuni a più corsi di studio.

Inserito in Ipotizzando di mantenere la serie storica dei manifesti degli studi per gli ultimi 5 anni, a regime saranno memorizzate 5 associazioni per ogni insegnamento. Il totale viene maggiorato del 10% per tenere conto del fatto che alcuni insegnamenti possono essere comuni a più corsi di studio. Si hanno quindi $(600 \times 5) \times 1.1 = 3300$ istanze.

Appello La media attuale è di 9-10 appelli all'anno per ciascun insegnamento. Tenendo conto che non occorre memorizzare tutti gli appelli di un anno accademico, ma soltanto quelli delle sessioni più prossime, e che il nuovo ordinamento comporta uno sfoltimento degli appelli, conviene fissare una media di 5 appelli per insegnamento, e quindi 3000 istanze totali.

Aula La nuova sede comprende 10 aule; in aggiunta a queste, occorre considerare che attualmente vengono utilizzate parecchie aule di altri Dipartimenti. Perciò il numero di aule sarà 30.

Lezione Si memorizzano le singole ore di lezione che si tengono nell'arco del ciclo didattico corrente. E' sufficiente tenere conto delle lezioni di una settimana, dato che lo schema settimanale si ripete. Si ipotizza una media di 10 ore a settimana per insegnamento, e si tiene conto che le lezioni interessano soltanto gli insegnamenti attivi nel ciclo corrente (rispettivamente un quarto e la metà dei totali per il nuovo e il vecchio ordinamento). Calcolando quindi che le lezioni interessino solo un terzo circa degli insegnamenti complessivi, il numero di istanze è pari a $(600 : 3) \times 10 = 2000$.

Sessione Il nuovo ordinamento prevede 6 sessioni (una per ogni ciclo didattico più due di recupero). A queste vanno aggiunte le sessioni del vecchio ordinamento (altre 6), quindi in totale ci saranno 12 istanze.

Studente Partendo dal dato degli iscritti nell'a. a. 1998/99 (2089 iscritti, fonte: MURST), è ragionevole una proiezione di circa 2500 studenti.

2.2.2 Ristrutturazione dello schema E/R

Eliminazione delle gerarchie

Avvisi Come mostrato in Figura 2.27, la gerarchia degli avvisi viene collassata verso il basso, trasferendo gli attributi alle entità figlie. Questa trasformazione

Concetto	*Tipo*	*Volume*	*Motivazione*
Corso di Studio	E	25	
Insegnamento	E	600	
Appello	E	3000	
Aula	E	30	
Lezione	E	2000	
Prevede	R	3000	600×5
Presso	R	2500	
In	R	2000	
Di	R	2000	
Sessione	E	12	
Relativo a	R	3000	
Personale	E	250	
Insegna 1	R	500	
Insegna 2	R	100	
Propedeuticità	R	350	
Inserito in	R	3300	
Materiale didattico	E	6000	600×10
Utilizza	R	6000	
Studente	E	2500	
Iscrizione	R	2500	
Esito	R	2500	
Frequenta	R	2500	
Rappresentante	E	30	
Esame Laurea	E	5	
Fac-Laurea	R	2	
Esame Stato	E	2	
Fac-Stato	R	2	
Ha	R	250	
Articolata in	R	25	
Avvisi	E	280	
Avvisi Insegnamento	E	300	600×0.5
Avvisi CdS	E	10	
Avvisi Facoltà	E	10	
Avvisi Rappres.	E	10	
Ins-Avv	R	300	
CdS-Avv	R	10	
Fac-Avv	R	10	
Rapp-Avv	R	10	

Tabella 2.2: Tabella dei volumi

è conveniente poiché le operazioni prevedono (per lo più) l'accesso separato a ciascuna delle entità figlie.

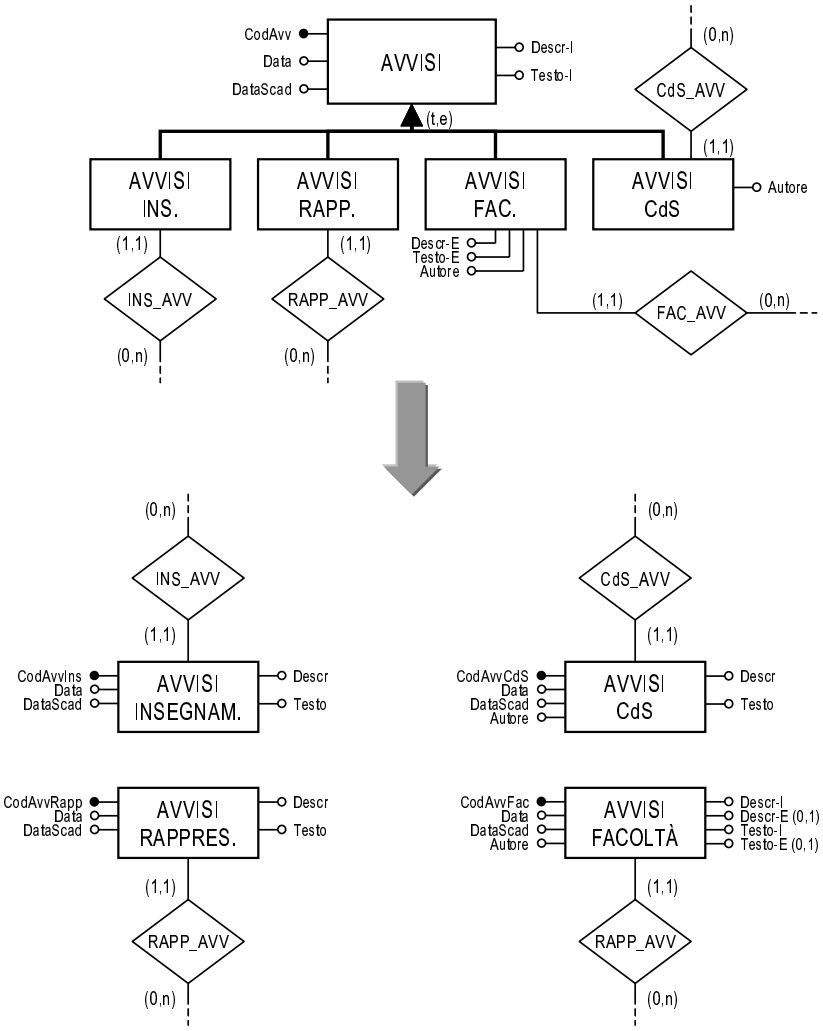


Figura 2.27: Eliminazione della gerarchia *Avvisi*

Rappresent. L'entità RAPPRESENT. è un subset di STUDENTE con una percentuale di istanze molto limitata (0.0012%). Sarebbe svantaggioso effettuare un collasso verso l'alto, dal momento che l'attributo selettore necessario assumerebbe quasi sempre valore nullo. Conviene quindi mantenere entrambe le entità, collegandole con un'associazione È. (Figura 2.28)

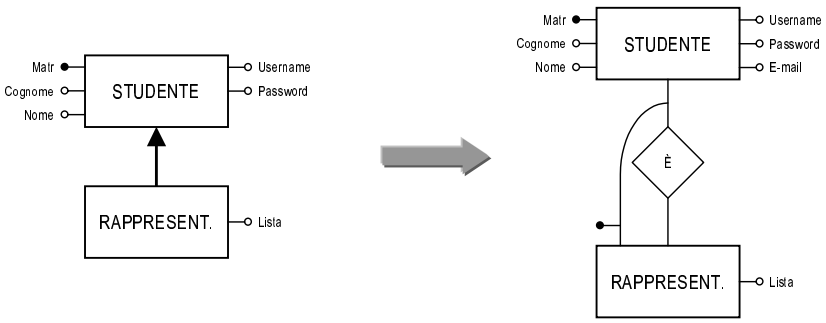


Figura 2.28: Eliminazione del subset *Rappresent.*

Personale Si decide di collassare completamente la gerarchia del personale in un'unica entità PERSONALE (figura 2.29). E' pertanto necessario introdurre l'attributo selettore *Qualifica*, che permette di distinguere fra le diverse categorie di personale classificate nella gerarchia. Occorre notare che *Qualifica* riassume in sè gli attributi selettori *PO-St*, *PaC-Pa* e *RuC-Ru*, che diventano quindi inutili e possono essere eliminati.

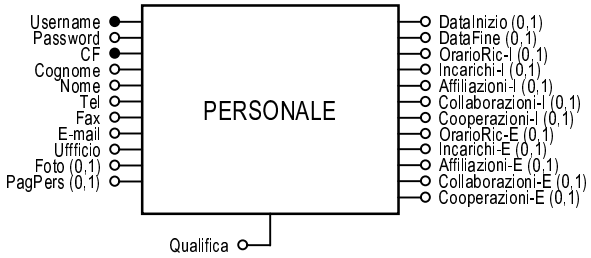


Figura 2.29: Eliminazione della gerarchia del personale

La tabella 2.3 riporta un esempio dei valori che l'attributo può assumere. Si può notare che, essendo la gerarchia di TECNICO_AMM a copertura parziale, è stato necessario introdurre un ulteriore valore, *amm*, per distinguere le entità che non appartengono alle due sottoentità.

Ulteriori trasformazioni

Partizionamento di entità In seguito al collasso verso l'alto della gerarchia del personale, l'entità PERSONALE presenta un gran numero di attributi. In particolare, un significativo sottoinsieme di questi riguarda esclusivamente il personale docente (è il caso degli attributi riguardanti l'orario di ricevimento e le diverse

Valore	*Significato*
PO	Professore ordinario
St	Professore straordinario
PaC	Professore associato confermato
Pa	Professore associato
RuC	Ricercatore confermato
Ru	Ricercatore
P	Preside
P-CCdS	Presid. Consiglio di Corso di Studio
PO-fr	Professore ordinario fuori ruolo
Pa-fr	Professore associato fuori ruolo
Cont	Docente a contratto
Suppl	Docente supplente
Tecn	Personale tecnico
Segr	Personale di segreteria
Amm	Personale amministrativo
Dott	Studente di dottorato
Ass	Assegnista
Ru-cont	Ricercatore a contratto

Tabella 2.3: Valori dell'attributo *Qualifica*

attività dei docenti), ed entra in gioco quasi esclusivamente nelle operazioni di gestione delle pagine personali. Per queste ragioni, si è pensato di operare un partizionamento di tipo verticale, separando gli attributi specifici per i docenti da quelli di utilità più generale (figura 2.30).

Accorpamento di associazioni La trasformazione della gerarchia del personale porta alla fusione delle associazioni INSEGNA_1 e INSEGNA_2 in un'unica associazione INSEGNA, come illustrato in figura 2.31. La partecipazione di PERSONALE è ovviamente facoltativa.

La stessa operazione viene effettuata per le associazioni PROPONE_1 e PROPONE_2.

Ruolo Riguardo al personale, rimane da considerare un aspetto importante. E' utile prevedere un meccanismo di personalizzazione dell'accesso al sito, finalizzato a presentare alle diverse tipologie di utenti soltanto i servizi di loro interesse ed a salvaguardare la sicurezza e riservatezza delle informazioni.

L'articolata gerarchia di generalizzazione del personale aveva, fra gli altri compiti, quello evidenziare le categorie di utenti abilitate ad interagire con il si-

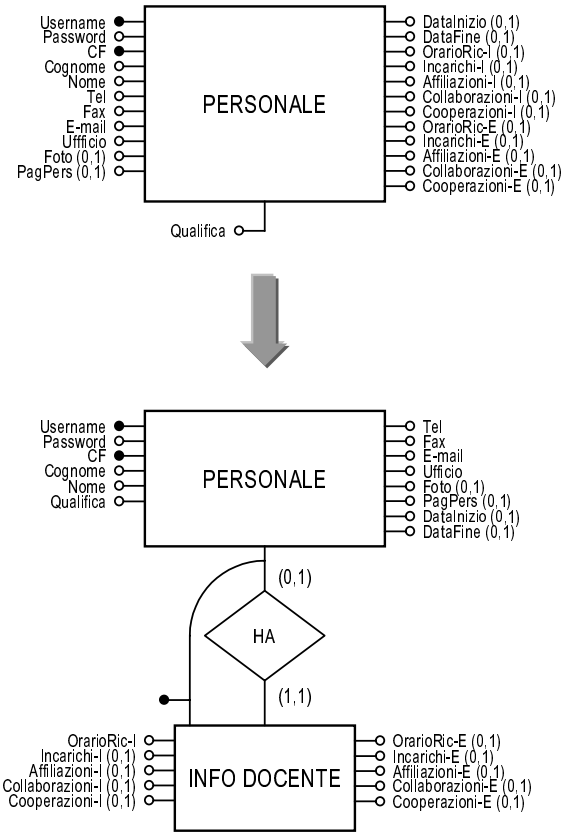


Figura 2.30: Partizionamento dell'entità PERSONALE

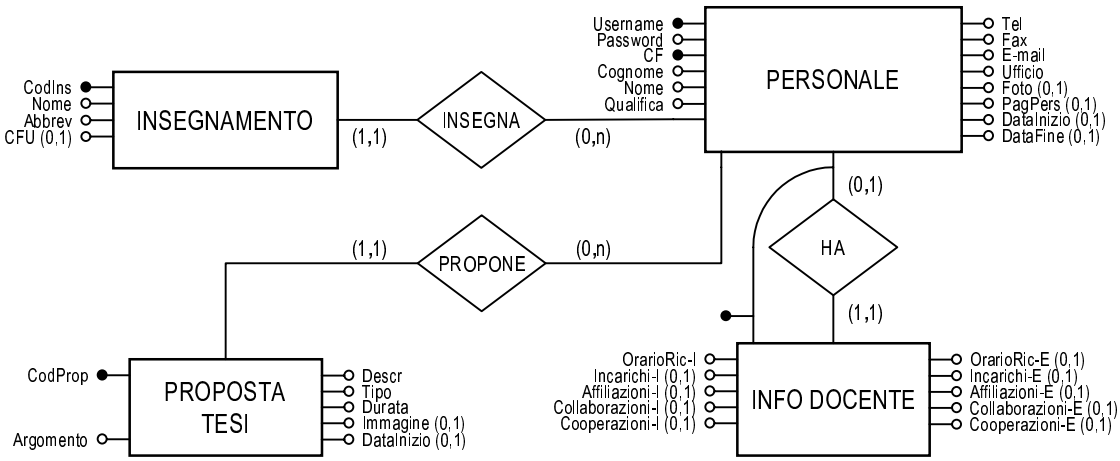


Figura 2.31: Accorpamento di associazioni

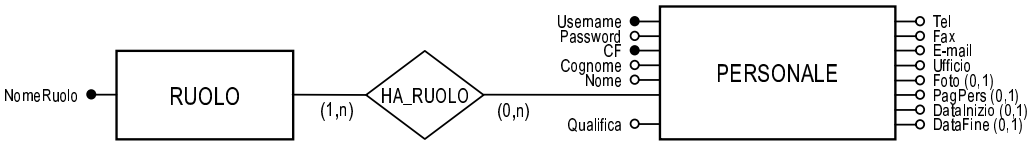


Figura 2.32: Inserimento dell'entità RUOLO

to allo scopo di modificarne i contenuti. Questa classificazione presenta però il difetto della scarsa flessibilità.

Una soluzione che presenta un buon grado di flessibilità è quella di abbinare chiunque debba effettuare interventi sui contenuti ad un *ruolo*, ognuno dei quali è contraddistinto da differenti permessi di modifica delle informazioni.

Ad esempio, chi ricopre il ruolo di *segreteria* ha la possibilità di operare sul calendario dell'anno accademico o sulla lista degli appelli, ma non è abilitato a modificare il materiale didattico relativo ad un insegnamento: tale operazione è permessa invece a chi è abbinato al ruolo di *docente*. I responsabili della gestione del sito, tramite il ruolo di *webmaster*, avranno la più ampia libertà di accesso.

Il meccanismo di autenticazione tramite *username* e *password* viene utilizzato per riconoscere l'utente quando questo vuole effettuare delle operazioni sui dati del sito. Il sistema, una volta identificato l'utente, riconosce il ruolo al quale è abbinato e può costruire una pagina di scelta con le operazioni effettivamente permesse.

Evitando in questa sede un'analisi più approfondita del meccanismo dei ruoli, occorre invece esaminare la modifica allo schema che questa soluzione comporta. Viene introdotta una nuova entità RUOLO, collegata a PERSONALE dall'associazione HA_RUOLO (figura 2.32). L'entità presenta un unico attributo, *NomeRuolo*, che ne costituisce anche l'identificatore.

L'associazione HA_RUOLO è di tipo molti a molti: ogni ruolo può essere abbinato a più persone e una persona può ricoprire più ruoli (con l'effetto di sommare i permessi di accesso dei singoli ruoli). L'associazione, inoltre, è obbligatoria per RUOLO (un ruolo, se esiste, deve essere ricoperto almeno da un membro del personale), mentre è facoltativa per PERSONALE: le cardinalità sono pertanto (1,n) e (0,n).

Eliminazione degli identificatori esterni

L'eliminazione degli identificatori esterni riguarda le entità LEZIONE, APPELLO, PROGRAMMA, RAPPRESENT e INFO_DOCENTE.

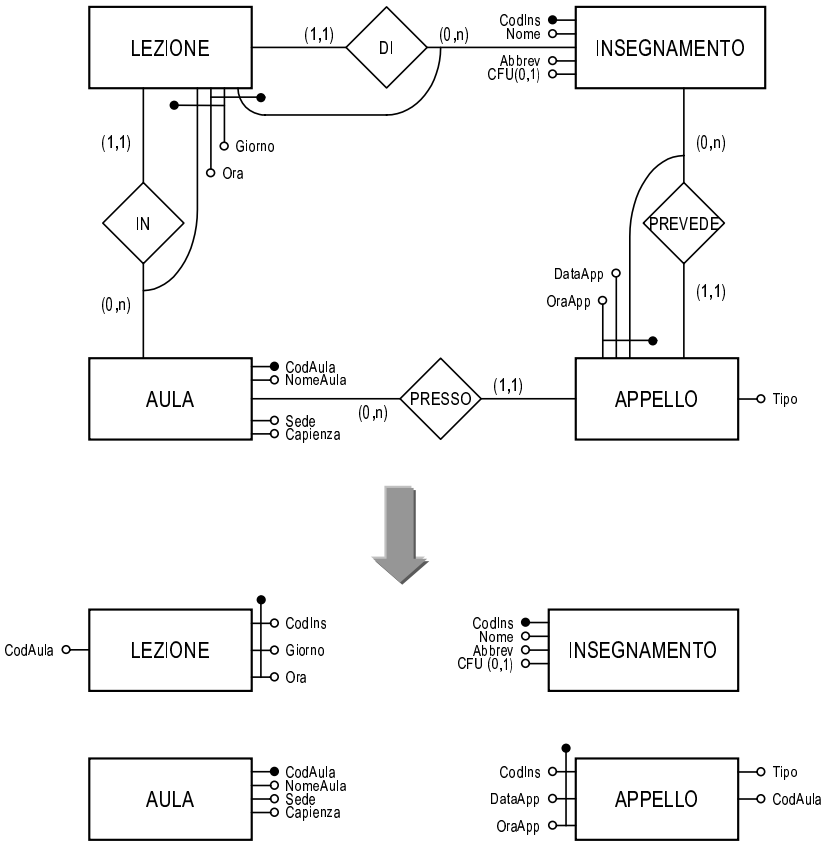


Figura 2.33: Eliminazione degli identificatori esterni di LEZIONE, APPELLO

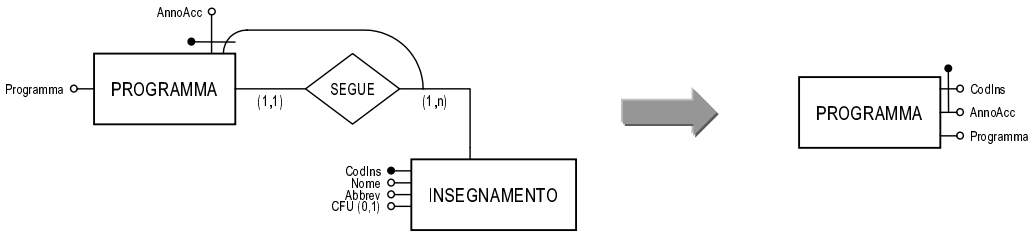


Figura 2.34: Eliminazione dell'identificatore esterno di PROGRAMMA

Schema E/R semplificato

A conclusione della fase di progetto logico di alto livello, in figura 2.37 si può vedere lo schema E/R semplificato risultante.

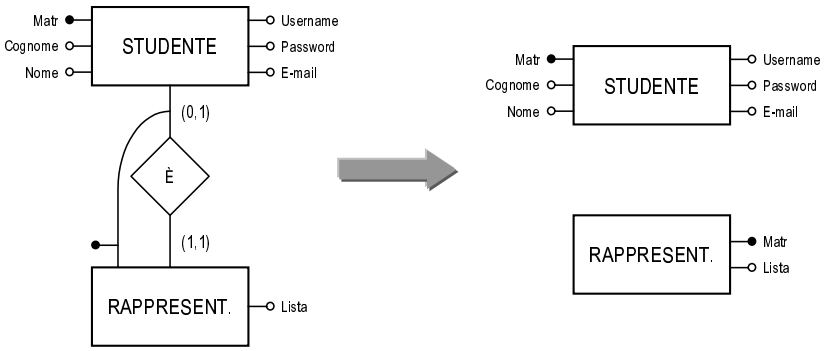


Figura 2.35: Eliminazione dell'identificatore esterno di RAPPRESENT

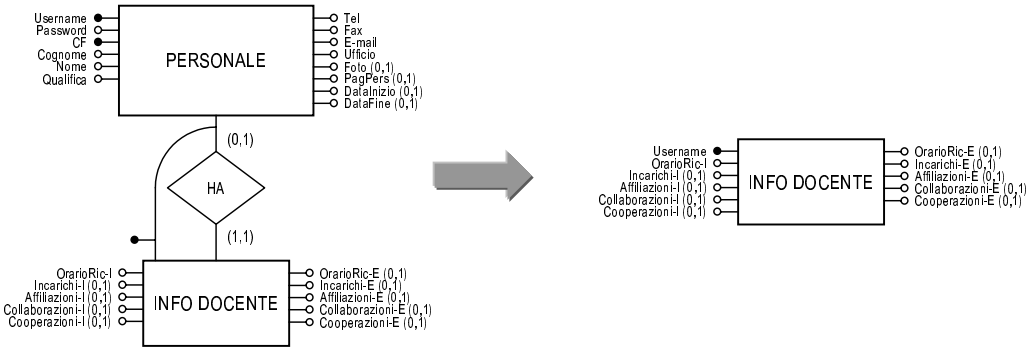
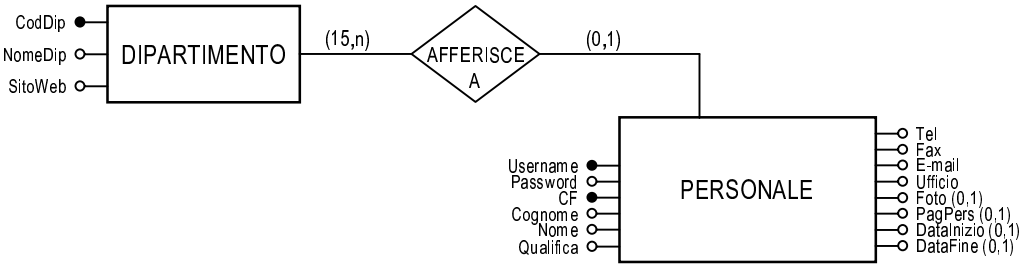


Figura 2.36: Eliminazione dell'identificatore esterno di DOCENTE_INFO

2.2.3 Progetto logico relazionale

Traduzione di entità ed associazioni



L'associazione AFFERISCE_A è di tipo uno a molti, pertanto si può tradurre la terna con due relazioni, compattando l'associazione nell'entità PERSONALE. Tra i due possibili identificatori di PERSONALE, inoltre, *Username* viene designato come chiave primaria; CF viene invece indicato come chiave alternativa.

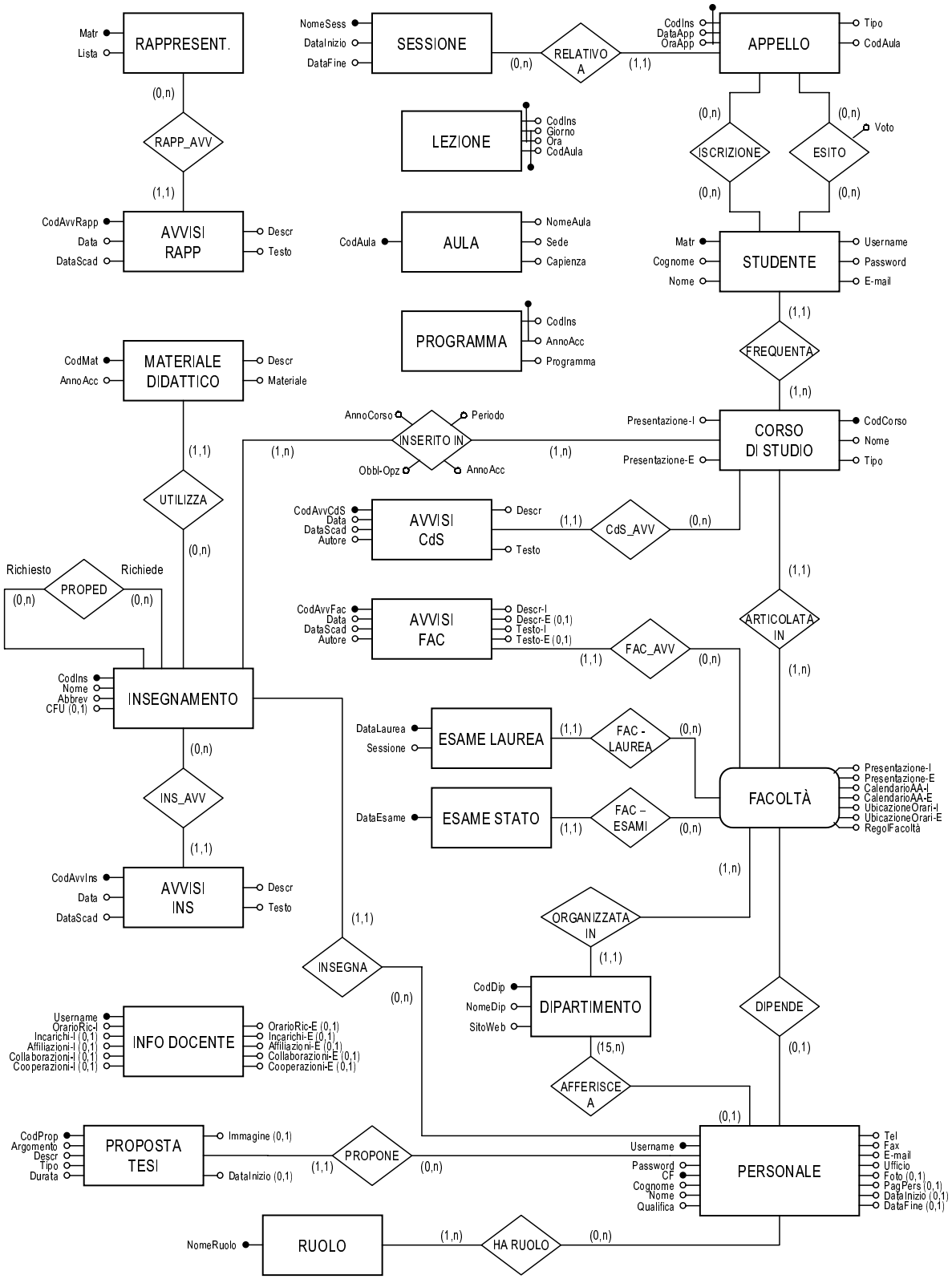
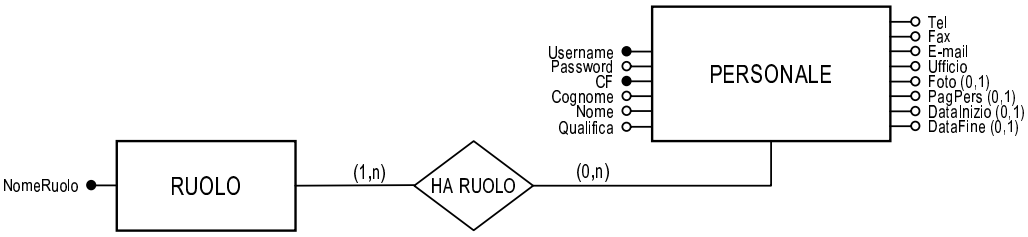


Figura 2.37: Schema E/R semplificato

```
Dipartimento(CodDip, NomeDip, SitoWeb)

Personale(Username, CF, Cognome, Nome, Qualifica, CodDip,
          Tel, Fax, Email, Ufficio, Foto, PagPers, Password,
          DataInizio, DataFine)
AK: CF
FK: CodDip REFERENCES Dipartimento
```

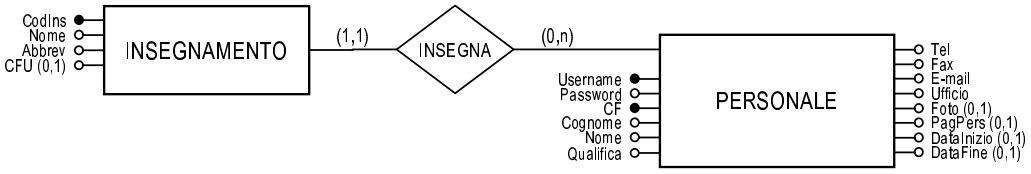

```
Info_Docente(Username, OrarioRic_I, OrarioRic_E, Incarichi_I,
       Incarichi_E, Affiliazioni_I, Affiliazioni_E,
       Collaborazioni_I, Collaborazioni_E, Cooperazioni_I,
       Cooperazioni_E)
FK: Username REFERENCES Personale
```



La partecipazione di entrambe le entità nell'associazione HA_RUOLO ha cardinalità massima maggiore di 1, per cui occorre tradurre l'associazione con una relazione. La traduzione di PERSONALE, effettuata in precedenza, non viene replicata.

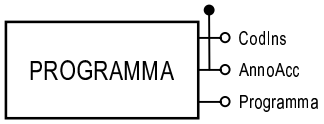
```
Ruolo(NomeRuolo)

Ha_Ruolo(Username, NomeRuolo)
FK: Username REFERENCES Personale
FK: NomeRuolo REFERENCES Ruolo
```

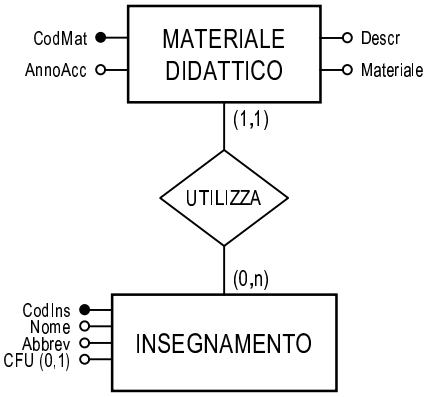


L'associazione INSEGNA è di tipo uno a molti, quindi viene inglobata nell'entità che partecipa con molteplicità unitaria.

```
Insegnamento(CodIns, Nome, Abbrev, CFU, Username)
FK: Username REFERENCES Docente
```

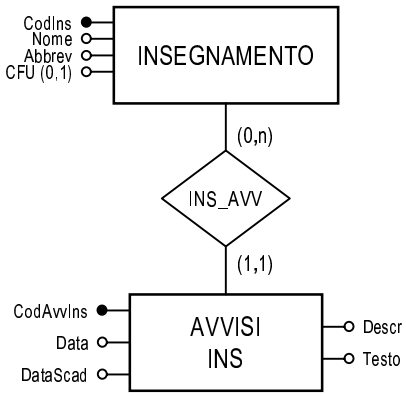



```
Programma(CodIns, AnnoAcc, Programma)
FK: CodIns REFERENCES Insegnamento
```



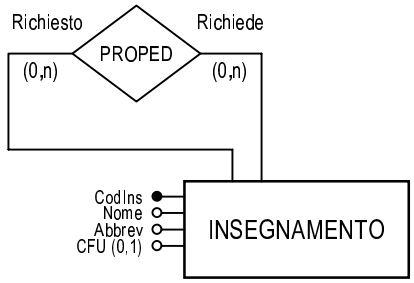
L'associazione UTILIZZA è di tipo uno a molti, pertanto si può tradurre la terna con due relazioni, compattando l'associazione nell'entità MATERIALE_DIDATTICO.

```
Materiale_Didattico(CodMat, CodIns, AnnoAcc, Descr, Materiale)
FK: CodIns REFERENCES Insegnamento
```



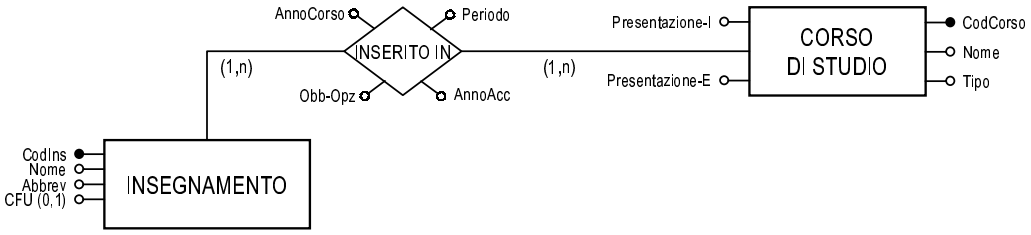
Anche in questo caso l'associazione INS_AVV viene inglobata dall'entità che partecipa con molteplicità unitaria.

```
Avvisi_Ins(CodAvvIns, CodIns, Data, DataScad, Descr, Testo)
FK: CodIns REFERENCES Insegnamento
```



L'anello molti a molti PROPED viene tradotto con una relazione, la cui chiave primaria è composta da due attributi che riflettono il diverso ruolo di INSEGNAMENTO nell'associazione. Entrambi gli attributi, inoltre, sono foreign key.

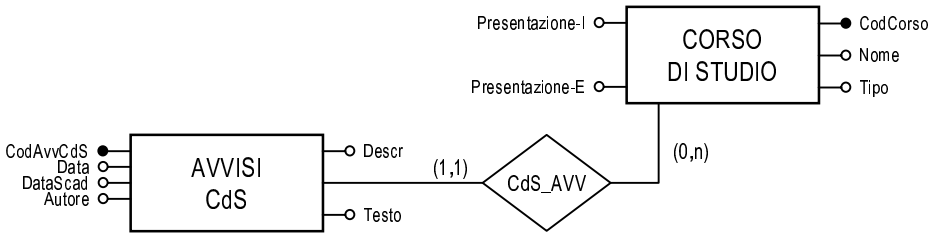
```
Proped(CodIns, CodIns_Richiesto)
FK: CodIns REFERENCES Insegnamento
FK: CodIns_Richiesto REFERENCES Insegnamento
```



In questo caso entrambe le entità partecipano ad INSERITO_IN con molteplicità maggiore di 1, per cui occorre seguire la traduzione standard. L'associazione viene quindi modellata con una relazione la cui chiave primaria è formata da *CodCorso* e *CodIns.*

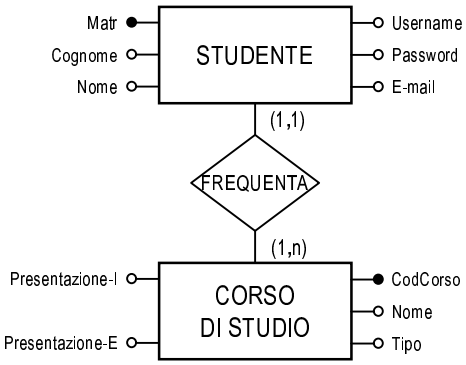
```
CdS(CodCorso, Nome, Tipo, Presentazione_I, Presentazione_E)

Inserito_in(CodCorso, CodIns, AnnoAcc, AnnoCorso, Periodo, Obb_Opz)
FK: CodCorso REFERENCES CdS
FK: CodIns REFERENCES Insegnamento
```



Si segue la traduzione tramite due relazioni.

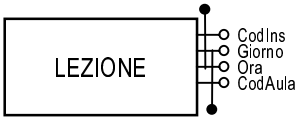
```
Avvisi_CdS(CodAvvCdS, CodCorso, Data, DataScad, Autore, Descr, Testo)
FK: CodCorso REFERENCES CdS
```



L'associazione FREQUENTA, di tipo uno a molti, viene collassata in STUDENTE.

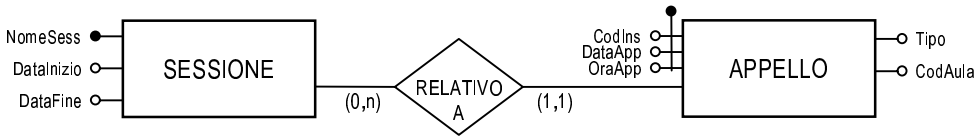
```
Studente(Matr, Cognome, Nome, Username, Password, Email, CodCorso)
FK: CodCorso REFERENCES CdS
```


```
Aula(CodAula, NomeAula, Sede, Capienza)
```



L'identificatore formato da *CodIns*, *Giorno* ed *Ora* viene designato come chiave primaria; l'altro identificatore viene indicato come chiave alternativa.

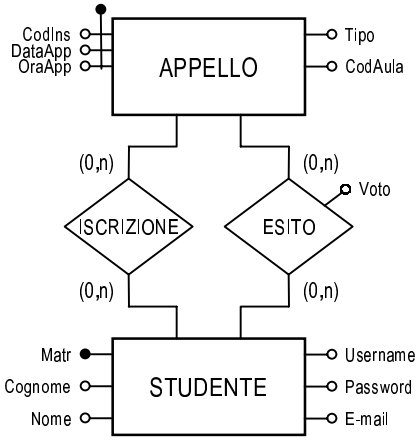
```
Lezione(CodIns, Giorno, Ora, CodAula)
AK: CodAula, Giorno, Ora
FK: CodIns REFERENCES Insegnamento
FK: CodAula REFERENCES Aula
```



Dal momento che RELATIVO_A è un'associazione uno a molti, conviene usare la traduzione con due relazioni.

```
Sessione(NomeSess, DataInizio, DataFine)

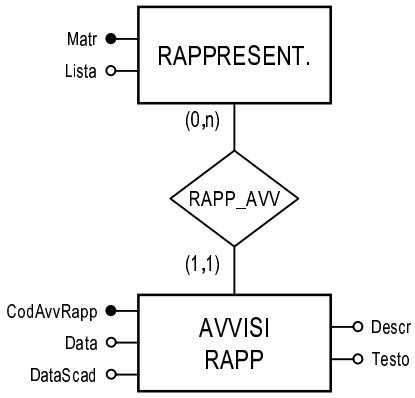
Appello(CodIns, DataApp, OraApp, NomeSess, CodAula, Tipo)
FK: CodIns REFERENCES Insegnamento
FK: NomeSess REFERENCES Sessione
FK: CodAula REFERENCES Aula
```



Le associazioni ISCRIZIONE ed ESITO vengono modellate tramite due relazioni.

```
Iscrizione(Matr, CodIns, DataApp, OraApp)
FK: Matr REFERENCES Studente
FK: CodIns, DataApp, OraApp REFERENCES Appello

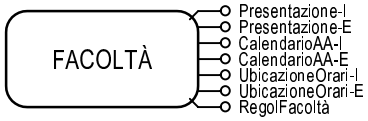
Esito(Matr, CodIns, DataApp, OraApp, Voto)
FK: Matr REFERENCES Studente
FK: CodIns, DataApp, OraApp REFERENCES Appello
```



Analogamente agli altri casi, RAPP_AVV viene collassata in AVVISI_RAPP.

```
Rappresentante(Matr, Lista)
FK: Matr REFERENCES Studente

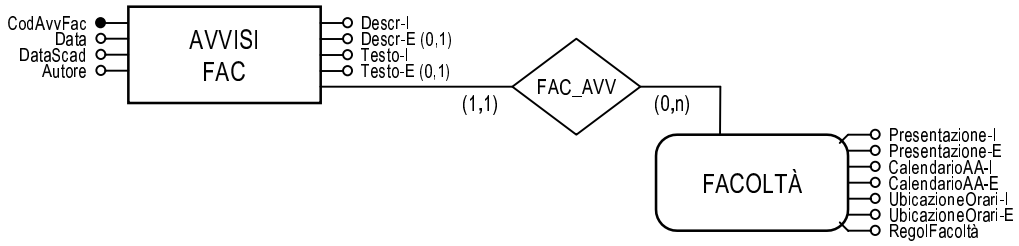
Avvisi_Rapp(CodAvvRapp, Data, DataScad, Descr, Testo, Matr)
FK: Matr REFERENCES Rappresentante
```



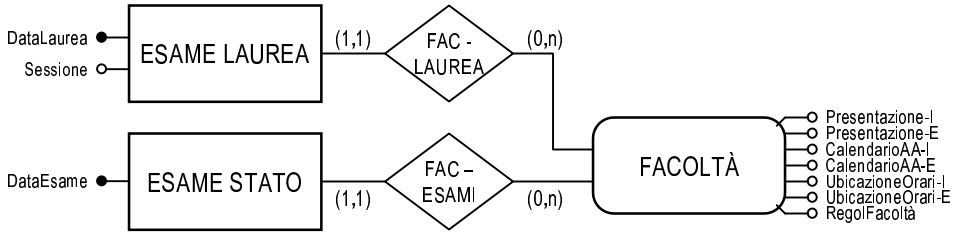
Come già visto in precedenza, FACOLTÀ è un'entità particolare, perché contiene una sola istanza. La scelta della chiave primaria è quindi del tutto libera ed ininfluente ai fini dell'utilizzo della relazione. Per coerenza con il resto dello schema, tuttavia, aggiungiamo un identificatore *NomeFac*, che ne diventa la chiave primaria.

```
Facolta(NomeFac, Presentazione_I, Presentazione_E,
        CalendarioAA_I, CalendarioAA_E, UbicazioneOrari_I,
        UbicazioneOrari_E, RegolFacolta)
```

A causa della particolare natura di FACOLTÀ, le varie associazioni che la collegano al resto dello schema non hanno altra utilità se non quella di migliorare la comprensione delle informazioni rappresentate. Tali associazioni possono pertanto essere tranquillamente ignorate.

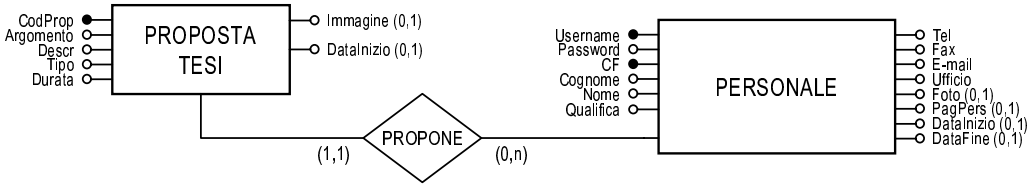



```
Avvisi_Fac(CodAvvFac, Data, DataScad, Autore, Descr_I,
           Descr_E, Testo_I, Testo_E)
```




```
Esame_Laurea(DataLaurea, Sessione)

Esame_Stato(DataEsame)
```



L'associazione PROPONE viene inglobata nell'entità PROPOSTA_TESI.

```
Proposta_Tesi(CodProp, Username, Argomento, Descr, Tipo, Durata,
              Immagine, DataInizio)
FK: Username REFERENCES Personale
```

2.2.4 Schema logico relazionale

Viene ora presentato lo schema relazionale completo.

```
Dipartimento(CodDip, NomeDip, SitoWeb)

Personale(Username, CF, Cognome, Nome, Qualifica, CodDip,
          Tel, Fax, Email, Ufficio, Foto, PagPers, Password,
          DataInizio, DataFine)
AK: CF
FK: CodDip REFERENCES Dipartimento

Info_Docente(Username, OrarioRic_I, OrarioRic_E, Incarichi_I,
             Incarichi_E, Affiliazioni_I, Affiliazioni_E,
             Collaborazioni_I, Collaborazioni_E, Cooperazioni_I,
             Cooperazioni_E)
FK: Username REFERENCES Personale

Ruolo(NomeRuolo)

Ha_Ruolo(Username, NomeRuolo)
FK: Username REFERENCES Personale
FK: NomeRuolo REFERENCES Ruolo

Insegnamento(CodIns, Nome, Abbrev, CFU, Username)
FK: Username REFERENCES Docente

Programma(CodIns, AnnoAcc, Programma)
FK: CodIns REFERENCES Insegnamento

Materiale_Didattico(CodMat, CodIns, AnnoAcc, Descr, Materiale)
FK: CodIns REFERENCES Insegnamento

Avvisi_Ins(CodAvvIns, CodIns, Data, DataScad, Descr, Testo)
FK: CodIns REFERENCES Insegnamento

Proped(CodIns, CodIns_Richiesto)
FK: CodIns REFERENCES Insegnamento
FK: CodIns_Richiesto REFERENCES Insegnamento
```

CdS(CodCorso, Nome, Tipo, Presentazione_I, Presentazione_E)

Inserito_in(CodCorso, CodIns, AnnoAcc, AnnoCorso, Periodo, Obb_Opz)
FK: CodCorso REFERENCES CdS
FK: CodIns REFERENCES Insegnamento

Avvisi_CdS(CodAvvCdS, CodCorso, Data, DataScad, Autore, Descr, Testo)
FK: CodCorso REFERENCES CdS

Studente(Matr, Cognome, Nome, Username, Password, Email, CodCorso)
FK: CodCorso REFERENCES CdS

Aula(CodAula, NomeAula, Sede, Capienza)

Lezione(CodIns, Giorno, Ora, CodAula)
AK: CodAula, Giorno, Ora
FK: CodIns REFERENCES Insegnamento
FK: CodAula REFERENCES Aula

Sessione(NomeSess, DataInizio, DataFine)

Appello(CodIns, DataApp, OraApp, NomeSess, CodAula, Tipo)
FK: CodIns REFERENCES Insegnamento
FK: NomeSess REFERENCES Sessione
FK: CodAula REFERENCES Aula

Iscrizione(Matr, CodIns, DataApp, OraApp)
FK: Matr REFERENCES Studente
FK: CodIns, DataApp, OraApp REFERENCES Appello

Esito(Matr, CodIns, DataApp, OraApp, Voto)
FK: Matr REFERENCES Studente
FK: CodIns, DataApp, OraApp REFERENCES Appello

Rappresentante(Matr, Lista)
FK: Matr REFERENCES Studente

Avvisi_Rapp(CodAvvRapp, Data, DataScad, Descr, Testo, Matr)
FK: Matr REFERENCES Rappresentante

```
Facolta(NomeFac, Presentazione_I, Presentazione_E,
        CalendarioAA_I, CalendarioAA_E, UbicazioneOrari_I,
        UbicazioneOrari_E, RegolFacolta)

Avvisi_Fac(CodAvvFac, Data, DataScad, Autore, Descr_I,
           Descr_E, Testo_I, Testo_E)

Esame_Laurea(DataLaurea, Sessione)

Esame_Stato(DataEsame)

Proposta_Tesi(CodProp, Username, Argomento, Descr, Tipo, Durata,
              Immagine, DataInizio)
FK: Username REFERENCES Personale
```

2.3 Implementazione su RDBMS Oracle8

Lo schema relazionale ottenuto è stato implementato su RDBMS Oracle8, versione 8.0.5.

E' stato scritto il codice SQL per la creazione delle tabelle e la contestuale introduzione dei vincoli di integrità. Successivamente sono stati realizzati diversi file SQL per il popolamento dello schema con un insieme di dati di ragionevoli dimensioni. Infine, sono stati creati alcuni script per una più facile gestione dello schema.

Complessivamente, la fase di implementazione ha richiesto la stesura di circa 910 linee di codice SQL commentato.

2.3.1 Vincoli di integrità

I vincoli di integrità definiti sono di cinque tipi:

- ***Not Null***
  Impone che un attributo di una tabella non possa contenere valori nulli.
- ***Unique***
  Stabilisce che un attributo od una combinazione di attributi si comporti come chiave, ossia che non possano esistere due tuple con gli stessi valori di questo attributo o gruppo di attributi.
- ***Primary Key***
  Definisce la chiave primaria di una tabella.
- ***Foreign Key* (integrità referenziale)**
  Stabilisce una relazione fra l'attributo o la combinazione di attributi definiti come foreign key e la chiave della tabella a cui si riferiscono.
- ***Check***
  Esprime un generico vincolo tramite un'espressione logico-relazionale. E' utile per effettuare controlli di vario tipo sui valori dei dati immessi.

I vincoli sopra descritti si possono presentare nella forma di *vincoli di tabella* oppure di *vincoli di colonna* (con l'esclusione del vincolo di tipo *not null*, che non può essere espresso nella prima forma).

Vincolo di tabella Fa parte della definizione di una tabella e impone una condizione su un qualunque insieme di colonne della tabella.

Vincolo di colonna Fa parte della definizione di una colonna e impone una condizione sull'attributo per il quale è stato introdotto.

2.3.2 Tipi di dati

Oracle mette a disposizione diversi tipi di dati, alcuni dei quali standard ed altri di tipo proprietario. Esaminiamo i principali tipi di dati utilizzati nella definizione degli attributi delle tabelle:

- `CHAR(`*n*`)`
  Stringa di caratteri di lunghezza fissa *n*.
- `VARCHAR2(`*n*`)`
  Stringa di caratteri di dimensione variabile con lunghezza massima *n*. È uguale al tipo di dato `VARCHAR`, ma l'uso di quest'ultimo viene sconsigliato poiché in future versioni di Oracle potrebbe essere ridefinito secondo una differente semantica di comparazione ([5]).
- `NUMBER(`*p*`,`*s*`)`
  Numero con *p* cifre a sinistra della virgola e *s* cifre a destra della virgola. Se *s* non viene fornito, la rappresentazione è di tipo intero.
- `DATE`
  Memorizza una data comprensiva di ora utilizzando un formato proprietario. In alcune situazioni è stato utilizzato per sostituire due attributi dedicati rispettivamente alla data e all'ora.

2.3.3 Definizione delle tabelle

In questo paragrafo viene presentato e descritto il codice SQL realizzato per la creazione delle tabelle dello schema. Tale codice rappresenta una valida guida di riferimento per illustrare i seguenti aspetti dell'implementazione:

- Nomi delle tabelle e dei loro attributi
- Tipi di dati associati agli attributi
- Vincoli di integrità

A tutti i vincoli creati viene assegnato un nome, per permettere una più comoda gestione dei vincoli stessi una volta che questi vengono memorizzati nel database.

Il nome viene costruito in maniera da risultare significativo. Esso comprende un prefisso, che specifica il tipo di vincolo, seguito dal nome della colonna (o della tabella) su cui agisce. Vediamo due esempi:

```
CONSTRAINT pk_Lezione PRIMARY KEY (CodIns, Giorno, Ora)
```

Questo vincolo, denominato `pk_Lezione`, definisce la chiave primaria della tabella `Lezione`.

```
Nome VARCHAR2(30)   CONSTRAINT nn_Nome NOT NULL
```

Vincolo di colonna che impedisce all'attributo `Nome` di contenere valori nulli.

In generale, si è scelto di usare la forma dei vincoli di colonna, per evidenziare in maniera più immediata le proprietà di ciascun attributo. Per i vincoli di tipo *check* ed i vincoli che coinvolgono più attributi contemporaneamente, invece, si è usata la forma dei vincoli di tabella.

Tenendo presente che l'implementazione del database effettuata nel presente lavoro di tesi ha valore di prototipo, si è scelto di rilassare, rispetto allo schema concettuale di riferimento, alcuni vincoli sull'obbligatorietà degli attributi, per permettere una maggiore flessibilità nell'inserimento dei dati.

Vengono ora presentate le definizioni delle singole tabelle.

Dipartimento

```
CREATE TABLE Dipartimento
(
CodDip   NUMBER(2)     CONSTRAINT pk_Dipartimento PRIMARY KEY,
NomeDip  VARCHAR2(60)  CONSTRAINT nn_NomeDip      NOT NULL,
SitoWeb  VARCHAR2(256)
);
```

Personale

```
CREATE TABLE Personale
(
Username   VARCHAR2(20)  CONSTRAINT pk_Personale PRIMARY KEY,
Cognome    VARCHAR2(30)  CONSTRAINT nn_Cognome   NOT NULL,
Nome       VARCHAR2(30)  CONSTRAINT nn_Nome      NOT NULL,
Qualifica  VARCHAR2(7)   CONSTRAINT nn_Qualifica NOT NULL,
CF         CHAR(16)      CONSTRAINT nn_CF        NOT NULL
                         CONSTRAINT ak_Personale UNIQUE,
Password   VARCHAR2(10)  CONSTRAINT nn_Password  NOT NULL,
CodDip                   CONSTRAINT fk_CodDip REFERENCES Dipartimento,
Tel        VARCHAR2(12),
```

```
Fax         VARCHAR2(12),
Email       VARCHAR2(30),
Ufficio     VARCHAR2(100),
Foto        VARCHAR2(256),  -- File esterno
PagPers     VARCHAR2(256),  -- Link alla pagina personale
DataInizio DATE,
DataFine    DATE
);
```

La chiave alternativa CF viene definita tramite i due vincoli di NOT NULL ed UNIQUE. Alternativamente, sarebbe stato possibile scrivere un unico vincolo del tipo:

```
    CF  CHAR(16)  NOT NULL UNIQUE
```

Una soluzione di questo tipo presenta una leggibilità migliore, ma comporta uno svantaggio. In fase di memorizzazione, Oracle scompone un vincolo del genere in due vincoli semplici (uno di tipo NOT NULL e l'altro di tipo UNIQUE), assegnando loro automaticamente due nomi poco comprensibili generati dal sistema. Si è quindi preferito scrivere due vincoli separati allo scopo di dare ad ognuno un nome significativo.

Info_Docente

```
CREATE TABLE Docente_Info
(
Username          CONSTRAINT pk_Docente_Info PRIMARY KEY
                  CONSTRAINT fk_Username     REFERENCES Personale,
OrarioRic_I       VARCHAR2(200)  CONSTRAINT nn_OrarioRic_I NOT NULL,
OrarioRic_E       VARCHAR2(200),
Incarichi_I       VARCHAR2(400),
Incarichi_E       VARCHAR2(400),
Affiliazioni_I    VARCHAR2(400),
Affiliazioni_E    VARCHAR2(400),
Collaborazioni_I VARCHAR2(400),
Collaborazioni_E VARCHAR2(400),
Cooperazioni_I    VARCHAR2(400),
Cooperazioni_E    VARCHAR2(400)
);
```

L'attributo *Username*, essendo una foreign key, non necessita dell'indicazione del tipo di dato, dal momento che assume quello della colonna a cui si riferisce.

Ruolo

```
CREATE TABLE Ruolo
(
NomeRuolo VARCHAR2(20)  CONSTRAINT pk_Ruolo PRIMARY KEY
);
```

Ha_Ruolo

```
CREATE TABLE Ha_Ruolo
(
Username     CONSTRAINT fk_Username  REFERENCES Personale,
NomeRuolo    CONSTRAINT fk_NomeRuolo REFERENCES Ruolo,

CONSTRAINT pk_Ruolo PRIMARY KEY (Username, Nomeruolo)
);
```

Insegnamento

```
CREATE TABLE Insegnamento
(
Codins       VARCHAR2(8)  CONSTRAINT pk_Insegnamento PRIMARY KEY,
Nome         VARCHAR2(60) CONSTRAINT nn_Nome_2 NOT NULL,
Abbrev       VARCHAR2(30) CONSTRAINT nn_Abbrev NOT NULL,
CFU          NUMBER(1),
Username     CONSTRAINT fk_Username_2 REFERENCES Personale,

CONSTRAINT ck_CFU CHECK (CFU>0)
);
```

CdS

```
CREATE TABLE CdS
(
CodCorso        CHAR(5)         CONSTRAINT pk_CdS PRIMARY KEY,
Nome            VARCHAR2(60)    CONSTRAINT nn_Nome_3 NOT NULL,
Tipo            VARCHAR2(2),    -- L, D, 1L, 2L, M, ...
Presentazione_I VARCHAR2(256),  -- File esterno
Presentazione_E VARCHAR2(256)   -- File esterno
);
```

Inserito_in

```
CREATE TABLE Inserito_In
(
CodIns    CONSTRAINT fk_CodIns    REFERENCES Insegnamento,
CodCorso  CONSTRAINT fk_CodCorso  REFERENCES CdS,
AnnoAcc   CHAR(9)    CONSTRAINT nn_AnnoAcc   NOT NULL,
AnnoCorso NUMBER(1)  CONSTRAINT nn_AnnoCorso NOT NULL,
Periodo   VARCHAR2(4),  -- 1, 2, 3, 4, 1Sem, 2Sem, Est
Obb_Opz   CHAR(3),      -- Obb, Opz

CONSTRAINT pk_Inserito_In PRIMARY KEY (CodIns, CodCorso),
CONSTRAINT ck_AnnoCorso CHECK ((AnnoCorso>=1) and (AnnoCorso<=5)),
CONSTRAINT ck_Obb_Opz   CHECK (Obb_Opz In ('Obb','Opz'))
);
```

Sono presenti due condizioni di controllo: la prima verifica che l'anno di corso sia compreso tra 1 e 5, mentre la seconda impone che il valore dell'attributo *Obb_Opz* sia "Obb" oppure "Opz".

Programma

```
CREATE TABLE Programma
(
CodIns    CONSTRAINT fk_CodIns_1  REFERENCES Insegnamento,
AnnoAcc   CHAR(9),
Programma VARCHAR2(256),  -- File esterno

CONSTRAINT pk_Programma PRIMARY KEY (CodIns, AnnoAcc)
);
```

Mat_Didattico

```
CREATE TABLE Mat_Didattico
(
CodMat    NUMBER(4) CONSTRAINT pk_Mat_Didattico PRIMARY KEY,
CodIns              CONSTRAINT fk_CodIns_2 REFERENCES Insegnamento,
AnnoAcc   CHAR(9),       -- es: 2000/2001
Descr     VARCHAR2(300),
Materiale VARCHAR2(256)  -- File esterno
);
```

Avvisi_Ins

```
CREATE TABLE Avvisi_Ins
(
CodAvvIns NUMBER(4) CONSTRAINT pk_Avvisi_Ins PRIMARY KEY,
CodIns              CONSTRAINT nn_CodIns     NOT NULL
                    CONSTRAINT fk_CodIns_3   REFERENCES Insegnamento,
Data      DATE      CONSTRAINT nn_Data       NOT NULL,
DataScad  DATE,
Descr     VARCHAR2(200)  CONSTRAINT nn_Descr NOT NULL,
Testo     VARCHAR2(2000) CONSTRAINT nn_Testo NOT NULL
);
```

Proped

```
CREATE TABLE Proped
(
CodIns            CONSTRAINT fk_CodIns_4 REFERENCES Insegnamento,
CodIns_Richiesto  CONSTRAINT fk_CodIns_5 REFERENCES Insegnamento,

CONSTRAINT pk_Proped PRIMARY KEY (CodIns, CodIns_Richiesto)
);
```

Avvisi_CdS

```
CREATE TABLE Avvisi_CdS
(
CodAvvCdS NUMBER(4)  CONSTRAINT pk_Avvisi_CdS PRIMARY KEY,
CodCorso             CONSTRAINT nn_CodCorso   NOT NULL
                     CONSTRAINT fk_CodCorso_2 REFERENCES CdS,
Data      DATE       CONSTRAINT nn_Data_2     NOT NULL,
DataScad  DATE,
Autore    VARCHAR2(30),
Descr     VARCHAR2(200)  CONSTRAINT nn_Descr_2 NOT NULL,
Testo     VARCHAR2(2000) CONSTRAINT nn_Testo_2 NOT NULL
);
```

Studente

```
CREATE TABLE Studente
(
Matr     NUMBER(4)     CONSTRAINT pk_Studente   PRIMARY KEY,
```

```
Cognome  VARCHAR2(20)  CONSTRAINT nn_Cognome_2  NOT NULL,
Nome     VARCHAR2(20)  CONSTRAINT nn_Nome_4     NOT NULL,
Username VARCHAR2(10)  CONSTRAINT nn_Username   NOT NULL
                       CONSTRAINT ak_Username   UNIQUE,
Password VARCHAR2(10)  CONSTRAINT nn_Password_2 NOT NULL,
Email    VARCHAR2(30),
CodCorso               CONSTRAINT nn_CodCorso_2 NOT NULL
                       CONSTRAINT fk_CodCorso_3 REFERENCES CdS
);
```

Aula

```
CREATE TABLE Aula
(
CodAula  VARCHAR2(7)   CONSTRAINT pk_Aula  PRIMARY KEY,
NomeAula VARCHAR2(60)  CONSTRAINT nn_Descr NOT NULL,
Sede     VARCHAR2(30)  CONSTRAINT nn_Sede  NOT NULL,
Capienza NUMBER(3)
);
```

Lezione

```
CREATE TABLE Lezione
(
CodIns             CONSTRAINT fk_CodIns_6 REFERENCES Insegnamento,
Giorno  NUMBER(1)  CONSTRAINT nn_Giorno   NOT NULL,
Ora     NUMBER(2)  CONSTRAINT nn_Ora      NOT NULL,
CodAula            CONSTRAINT nn_CodAula  NOT NULL
                   CONSTRAINT fk_CodAula  REFERENCES Aula,

CONSTRAINT pk_Lezione PRIMARY KEY (CodIns, Giorno, Ora),
CONSTRAINT ak_Lezione UNIQUE (CodAula, Giorno, Ora),
CONSTRAINT ck_Giorno  CHECK ((Giorno>=1) and (Giorno<=6)),
CONSTRAINT ck_Ora     CHECK ((Ora>=8) and (Ora<=19)),
);
```

L'attributo *Giorno*, che rappresenta il giorno della settimana, codificato tramite un valore numerico, che può variare da 1 (lunedì) a 6 (sabato). Sono stati inseriti due vincoli di controllo sul range di valori assumibili dalle colonne *Giorno* ed *Ora*.

Sessione

```
CREATE TABLE Sessione
(
NomeSess   VARCHAR(30)  CONSTRAINT pk_Sessione PRIMARY KEY,
DataInizio DATE,
DataFine   DATE
);
```

Appello

```
CREATE TABLE Appello
(
CodIns             CONSTRAINT fk_CodIns_7  REFERENCES Insegnamento,
DataOra DATE,      -- memorizza sia la data che l'ora
Tipo    CHAR(1),   -- s, o
CodAula            CONSTRAINT nn_CodAula_2 NOT NULL
                   CONSTRAINT fk_CodAula_2 REFERENCES Aula,
NomeSess           CONSTRAINT fk_NomeSess  REFERENCES Sessione,

CONSTRAINT pk_Appello PRIMARY KEY (CodIns, DataOra),
);
```

Iscrizione

```
CREATE TABLE Iscrizione
(
Matr       CONSTRAINT fk_Matr REFERENCES Studente,
CodIns,
DataOra,

CONSTRAINT pk_Iscrizione PRIMARY KEY (Matr, CodIns, DataOra),
CONSTRAINT fk_CodIns_DataOra
    FOREIGN KEY (CodIns, DataOra) REFERENCES Appello(CodIns, Dataora)
);
```

Esito

```
CREATE TABLE Esito
(
Matr        CONSTRAINT fk_Matr_2 REFERENCES Studente,
CodIns,
```

```
DataOra,
Voto    Number(2),

CONSTRAINT pk_Esito PRIMARY KEY (Matr, CodIns, DataOra),
CONSTRAINT fk_CodIns_DataOra_2
    FOREIGN KEY (CodIns, DataOra) REFERENCES Appello(CodIns, Dataora)
);
```

Rappresentante

```
CREATE TABLE Rappresentante
(
Matr      CONSTRAINT pk_Rappresentante PRIMARY KEY
          CONSTRAINT fk_Matr_3         REFERENCES Studente,
Lista     VARCHAR2(30)
);
```

Avvisi_Rapp

```
CREATE TABLE Avvisi_Rapp
(
CodAvvRapp NUMBER(4)  CONSTRAINT pk_Avvisi_Rapp PRIMARY KEY,
Matr                  CONSTRAINT fk_Matr_4      REFERENCES Studente,
Data       DATE       CONSTRAINT nn_Data_3      NOT NULL,
DataScad   DATE,
Descr      VARCHAR2(200)  CONSTRAINT nn_Descr_3 NOT NULL,
Testo      VARCHAR2(2000) CONSTRAINT nn_Testo_3 NOT NULL
);
```

Facoltà

```
CREATE TABLE Facolta
(
Presentazione_I VARCHAR2(256),  -- File esterno
Presentazione_E VARCHAR2(256),  -- File esterno
CalendarioAA_I  VARCHAR2(256),  -- File esterno
CalendarioAA_E  VARCHAR2(256),  -- File esterno
InfoGen_I       VARCHAR2(256),  -- File esterno
InfoGen_I       VARCHAR2(256),  -- File esterno
RegolFacolta    VARCHAR2(256)   -- File esterno
);
```

Avvisi_Fac

```
CREATE TABLE Avvisi_Fac
(
CodAvvFac NUMBER(4)  CONSTRAINT pk_Avvisi_Fac PRIMARY KEY,
Data      DATE       CONSTRAINT nn_Data_4 NOT NULL,
DataScad  DATE,
Autore    VARCHAR2(40),
Descr     VARCHAR2(200)  CONSTRAINT nn_Testo_4 NOT NULL,
Testo     VARCHAR2(2000) CONSTRAINT nn_Testo_4 NOT NULL
);
```

Esame_Laurea

```
CREATE TABLE Esame_Laurea
(
DataLaurea  DATE  CONSTRAINT pk_Esame_Laurea PRIMARY KEY,
NomeSess          CONSTRAINT fk_NomeSess_2   REFERENCES Sessione
);
```

Esame_Stato

```
CREATE TABLE Esame_Stato
(
DataEsame  DATE   CONSTRAINT pk_Esame_Stato PRIMARY KEY
);
```

Proposta_Tesi

```
CREATE TABLE Proposta_Tesi
(
CodProp    NUMBER(4)      CONSTRAINT pk_Proposta_Tesi PRIMARY KEY,
Username   CONSTRAINT nn_Username_2  NOT NULL
           CONSTRAINT fk_Username_2  REFERENCES Insegnamento,
Argomento  VARCHAR2(200)  CONSTRAINT nn_Argomento     NOT NULL,
Descr      VARCHAR2(3000) CONSTRAINT nn_Descr         NOT NULL,
Tipo       CHAR(12)       CONSTRAINT nn_Tipo          NOT NULL,
Durata     NUMBER(2),
Immagine   VARCHAR2(256), -- File esterno
DataInizio DATE,
CONSTRAINT ck_Durata CHECK ((Durata>=1) and (Durata<=18))
);
```

Un vincolo di controllo forza la colonna *Durata* ad assumere valori compresi tra 1 e 18 (ricordando che questo attributo rappresenta la durata prevista in mesi, il limite imposto sembra ragionevole).

Capitolo 3

Architettura funzionale del sito Web

In questo capitolo viene illustrata la proposta architetturale per l'implementazione del sito web.

La scelta della soluzione architetturale da adottare per la costruzione di un sito web complesso, qual'è quello di una Facoltà universitaria, è una fase molto importante che va ponderata con attenzione. Essa, infatti, condiziona fortemente le successive attività di implementazione, aggiornamento ed evoluzione del sito.

La decisione di utilizzare un'architettura o una tecnologia non deve essere effettuata solamente in base a criteri di efficienza e velocità nel gestire le richieste degli utenti. E' necessario prestare particolare attenzione al raggiungimento dei seguenti obiettivi:

- efficacia della fase di implementazione
- facilità di manutenzione ed aggiornamento dei contenuti
- flessibilità nel supporto di evoluzioni future.

L'architettura che si propone deve necessariamente essere in grado di gestire contenuti dinamici.

Per un trattazione più approfondita delle tecnologie qui descritte, è necessario fare riferimento all'appendice B.

3.1 L'evoluzione dello sviluppo di siti web

Con la sorprendente esplosione di Internet che si è verificata negli ultimi anni, il World Wide Web è diventato uno degli strumenti principali per la diffusione delle informazioni.

Il continuo fiorire di nuove tecnologie e la rapida evoluzione di quelle già esistenti hanno consentito alle applicazioni web, prima limitate al ristretto ambito della navigazione ipertestuale di documenti di vario genere (accademico/scientifico, commerciale/pubblicitario, sociale e di intrattenimento), di acquisire caratteristiche tali da permettere loro di confrontarsi sul vasto terreno dei sistemi informativi tradizionali.

Il concetto di sito web sta evolvendo da semplice collezione di documenti verso un vero e proprio sistema informativo integrato con altri sistemi, primi fra tutti i DBMS.

Questa evoluzione ha spinto le applicazioni web verso livelli di complessità paragonabili a quelli delle grosse applicazioni desktop di tipo tradizionale, ed ha influito notevolmente sulle attività di sviluppo e manutenzione dei siti web.

In un passato neanche troppo lontano, l'attività di sviluppo e gestione di un sito web poteva essere condotta con tecniche di tipo artigianale. I siti web, di dimensioni limitate e di aspetto semplice, erano costituiti da pagine HTML statiche, con l'aggiunta tutt'al più di qualche script CGI per la aumentare l'interazione con l'utente. I compiti di preparazione dei contenuti, di design grafico, di stesura del codice HTML e di programmazione potevano essere assolti da un'unica persona.

La nascita di nuove tecnologie, l'integrazione con i DBMS per l'accesso a basi di dati, la sempre crescente importanza dell'aspetto grafico e le grandi dimensioni delle applicazioni hanno reso inadeguato un approccio di questo tipo, portando contemporaneamente alla ribalta nuove figure professionali.

Lo sviluppo dei siti web moderni è infatti un'attività che coinvolge diverse discipline. Se ne possono citare alcune:

- Designer grafici
- Programmatori
- Amministratori di database
- Autori di contenuti

Il ruolo del designer grafico (il cosiddetto *web artist*), per esempio, ha assunto una notevole importanza, in considerazione del fatto che l'aspetto estetico è una componente molto rilevante di un'applicazione web. Il lavoro di un *web artist* è quello di produrre pagine HTML graficamente accattivanti, e non deve necessariamente avere competenze di programmazione.

Analogamente, il programmatore dovrebbe potersi idealmente concentrare sugli aspetti implementativi, senza preoccuparsi di questioni a lui estranee quali

l'aspetto finale delle pagine o la modifica dei contenuti.

Coordinare il contributo delle diverse competenze non è un compito semplice. I contenuti statici, i contenuti dinamici, la logica di interazione con l'utente e l'aspetto grafico devono funzionare tutti insieme.

Questa riflessione sulle peculiarità dello sviluppo di siti web evidenzia chiaramente come sia importante ricercare un alto grado di separazione fra *presentazione*, *contenuto* e *logica applicativa*.

3.2 Presentazione, contenuto e logica applicativa

Per chiarezza, vediamo una sintetica spiegazione di questi tre importanti concetti:

- **Presentazione** Rappresenta la forma con la quale i contenuti vengono pubblicati, e più in generale tutto ciò che riguarda l'aspetto estetico del sito.
- **Contenuto** Costituisce essenzialmente l'insieme delle informazioni e dei documenti che devono essere gestiti dal sito web; in pratica, *cosa* deve essere pubblicato.
- **Logica applicativa** Detta anche *business logic*, è l'insieme delle regole che governano l'interazione con l'utente e il funzionamento dell'applicazione.

La separazione fra questi tre aspetti rappresenta un fattore chiave per migliorare le attività di sviluppo e manutenzione di un sito web complesso quale è il sito di una Facoltà.

3.3 Limiti delle attuali tecnologie

L'efficienza nello sviluppo di siti web complessi è spesso condizionata dai limiti delle tecnologie utilizzate. Il principale fra questi limiti, riscontrabile in molte delle soluzioni tecnologiche di maggior diffusione, è sicuramente la mancanza di separazione fra contenuto, presentazione e logica applicativa.

CGI

La tecnica CGI (Common Gateway Interface), una delle prime soluzioni per l'interazione tra client e server sul web (e ancora oggi molto usata), rappresenta un esempio emblematico di commistione fra contenuto, stile e logica.

Uno script CGI non è altro che un programma, eseguito dal server, che risponde ad una specifica richiesta di un utente producendo una pagina web contenente i risultati della particolare elaborazione richiesta. Le istruzioni HTML necessarie alla preparazione della pagina di risposta sono ”affogate” nel codice del programma CGI.

Questa caratteristica ha conseguenze facilmente immaginabili sulla facilità delle operazioni di manutenzione. Una semplice variazione dell'estetica delle pagine richiede la modifica del codice sorgente dei programmi CGI e la loro eventuale ricompilazione (nel caso di linguaggi compilati). Un'operazione di questo genere richiede necessariamente delle competenze di programmazione, ed è quindi fuori della portata di una figura come il *web artist*.

Servlet

Le Servlet Java sono diventate una piattaforma di primaria importanza per lo sviluppo web sul lato server. Esse riscuotono grande successo per diversi motivi:

- sono decisamente più efficienti degli script CGI
- offrono un'ottima integrazione con le sorgenti di dati presenti nel back-end
- la Servlet API è disponibile su tutte le maggiori piattaforme hardware, per cui le applicazioni godono di un'ottima portabilità.

Con l'approccio comunemente usato per la generazione del contenuto HTML, però, il problema di fondo rimane lo stesso: le Servlet incoraggiano ad inserire il codice per la presentazione (HTML) direttamente nel codice Java responsabile della logica applicativa. La creazione del markup HTML tramite l'utilizzo di istruzioni `out.println()` diventa un compito lungo e oneroso per pagine web di grosse dimensioni, e rende l'aggiornamento del sito un'operazione altamente dispendiosa, a causa della necessità di far intervenire dei programmatori per una qualsiasi modifica al contenuto od alla presentazione.

Pagine web contenenti script

Questo tipo di soluzione sta incontrando un vasto consenso. Sono diverse le tecnologie appartenenti a questo filone. Tra le principali si possono citare:

- ASP (Active Server Pages) di Microsoft
- JSP (Java Server Pages) di Sun Microsystems

- Cold Fusion di Allaire

- PHP

Nel seguito si farà riferimento alle due soluzioni più rappresentative, e cioè JSP e ASP.

Il principio su cui si basano i prodotti sopra elencati è il medesimo per tutti: si creano delle pagine web contenenti frammenti di codice. Al momento della richiesta della pagina, un apposito modulo installato sul server interpreta questo codice e restituisce la pagina HTML comprensiva dei risultati delle elaborazioni effettuate dagli script.

Le differenze stanno nelle piattaforme e nei linguaggi supportati. ASP, disponibile soltanto per il mondo Windows, permette l'utilizzo di diversi linguaggi di scripting, con la prevalenza di VBScript. JSP utilizza invece il linguaggio Java, ed è una tecnologia più aperta essendo indipendente dalla piattaforma e dal server.

L'approccio adottato da queste soluzioni facilita sicuramente il ricorso alla programmazione e consente un buon grado di flessibilità.

Anche se queste tecnologie rendono più rapido ed accessibile lo sviluppo di siti web con contenuti dinamici, difficilmente però possono rappresentare un passo in avanti nella separazione fra logica, presentazione e contenuto.

Come si è visto, infatti, il principio di base è opposto a quello adottato per gli script CGI e le servlet: invece di inserire la pagina HTML dentro il codice, si inserisce il codice dentro la pagina HTML. Una soluzione di questo genere non costituisce certo un grande miglioramento rispetto alla precedente.

JSP e ASP, in realtà, offrono dei meccanismi che sulla carta permettono un certo grado di separazione fra logica applicativa e presentazione. È possibile spostare le parti di codice "pesante" al di fuori delle pagine, creando dei componenti esterni (JavaBeans per JSP o ActiveX per ASP), lasciando nelle pagine soltanto il codice necessario per accedere questi oggetti.

Questi metodi, però, non forniscono la separazione pulita che promettono. Rimane pur sempre necessario inglobare nelle pagine web porzioni di codice. L'eventuale creazione dei componenti esterni è inoltre lasciato alla buona volontà dello sviluppatore, che, in mancanza di espliciti vincoli, sarà sempre tentato di utilizzare il codice direttamente nelle pagine.

3.4 La soluzione proposta

Si è visto come le tecnologie per l'integrazione tra web e database presentino, in diverse misure, un'indesiderata commistione fra i tre aspetti della presentazione, del contenuto e della logica applicativa.

Esiste un approccio molto interessante che permette di risolvere questo problema in maniera efficiente. Tale soluzione si basa sulla combinazione di di tre elementi:

- il linguaggio XML
- i fogli di stile XSL
- le servlet Java

Prima di procedere con l'analisi dell'architettura, è necessario esaminare un po' più da vicino i primi due "ingredienti" citati, allo scopo di capire meglio l'utilizzo che ne verrà fatto.

3.4.1 XML

XML (eXtensible Markup Language) è un linguaggio di markup relativamente nuovo, sviluppato dal World Wide Web Consortium (W3C) e recentemente adottato come standard dallo stesso organismo.

XML nasce come un linguaggio di descrizione e si pensa che possa diventare, per le sue caratteristiche, un formato universale per lo scambio di dati su Internet.

Nato per operare all'interno del World Wide Web, esso si pone in contrapposizione con l'attuale standard per la distribuzione e la visualizzazione di documenti sul web, cioè l'HTML. Lo scopo è infatti quello di costituire uno standard per lo scambio di dati in forma *machine readable*, al contrario di HTML, orientato allo scambio di documenti *human readable*.

Analogamente ad un documento HTML, un documento XML contiene del testo annotato con tag. Tuttavia, diversamente dal linguaggio HTML, lo standard XML non possiede un insieme di tag precostituito: i tag vengono creati in base alle necessità, ed è possibile definirne un numero illimitato.

Mentre l'HTML possiede molti tag per indicare l'aspetto che deve assumere un elemento testuale, i tag creati in XML vengono utilizzati per descrivere la struttura logica dei dati.

Vediamo ora un esempio. Il seguente frammento di codice XML rappresenta alcune informazioni riguardanti una scuderia di Formula 1:

```
<scuderia>
  <nome>Ferrari</nome>
  <direttore-sportivo>Jean Todt</direttore-sportivo>
  <pilota id="1">Michael Schumacher</pilota>
  <pilota id="2">Rubens Barrichello</pilota>
</scuderia>
```

Per quanto si è detto, il linguaggio XML costituisce un ottimo mezzo per descrivere i contenuti, statici o dinamici, del sito, prescindendo completamente dall'aspetto finale con cui questi contenuti verranno presentati. La capacità di descrivere in maniera estremamente naturale informazioni di tipo strutturato, inoltre, rende questo standard perfettamente idoneo per rappresentare record di dati provenienti da un database.

Un documento XML non possono essere visualizzati cos'ıcome sono, ma deve essere associato ad un foglio di stile che ne descriva le regole relative alla rappresentazione.

3.4.2 XSL

XSL (eXtensible Stylesheet Language) è il linguaggio per i fogli di stile creato appositamente per lo standard XML. Esso è costituito da due parti:

- **XSL Transformations (XSLT)**
  Consente di definire le regole tramite le quali un documento XML viene trasformato in un altro documento XML. In documento trasformato può conformarsi al markup e al DTD del documento originale, oppure può usare un insieme di tag completamente diverso. In particolare, può usare i tag definiti dalla seconda parte di XSL, i Formatting Objects.

- **XSL Formatting Objects**
  Consiste in un *vocabolario XML* che specifica la formattazione in maniera più dettagliata e ad un livello più basso di quanto possibile con lo standard CSS

Essenzialmente, quindi, XSL è formato da due linguaggi: il primo è un linguaggio di trasformazione, il secondo è un linguaggio di formattazione. Entrambi sono delle applicazioni di XML.

Il linguaggio di trasformazione (XSLT) è utilizzabile indipendentemente dal linguaggio di formattazione. Per questo motivo, molte implementazioni di XSL si concentrano esclusivamente sulla parte di trasformazione ed ignorano i Formatting Objects.

L'utilizzo più importante di XSLT consiste nel trasformare un documento XML in un documento XHTML, oppure in un documento HTML per garantire la compatibilità con i browser meno recenti.

La trasformazione avviene tramite un apposito modulo software, detto *XSLT processor*.

3.4.3 Analisi dell'architettura

La soluzione considerata prevede che il contenuto dinamico di una pagina venga prodotto tramite l'utilizzo di XML, senza quindi preoccuparsi della presentazione. Il contenuto statico, lo stile e l'aspetto finale della pagina stessa viene delegato ad un apposito foglio di stile XSLT, che contiene il necessario codice di markup per la produzione di una pagina web finita.

Un'impostazione di questo genere permette di ottenere una convincente separazione fra presentazione, logica applicativa e contenuto. Vediamo quindi come si possono dividere le responsabilità nella realizzazione del sito:

- Il programmatore lavora sulle servlet Java, che gestiscono la logica dell'applicazione e producono il contenuto dinamico sotto forma di XML (invece di generare direttamente le pagine HTML, come succede nell'approccio classico).
- Il web designer si occupa esclusivamente della presentazione, producendo i fogli di stile XSLT responsabili della generazione delle pagine web complete. Tutto quello che occorre conoscere è la struttura (o meglio la DTD, Document Type Definition) del documento XML che si deve trasformare.

In figura 3.1 è possibile vedere l'impostazione generale del sistema, che presenta la configurazione tipica di un'architettura client-server a tre livelli.

- Al primo livello si trova l'utente, il *client*, che interagisce per mezzo di un *web browser*.
- Il livello intermedio comprende il *Web server* ed il software necessario alla gestione della logica applicativa. Il *Web server* riceve le richieste del client e può rispondere con l'invio di pagine statiche oppure richiedere la generazione di pagine dinamiche. Il modulo della logica applicativa (che, come si è detto, sarà composto principalmente da servlet Java) interagisce con il database e con l'insieme dei fogli di stile XSLT per produrre pagine web con contenuto dinamico.
- Il livello di back-end è costituito infine dal DBMS che gestisce i dati di interesse per il sito.

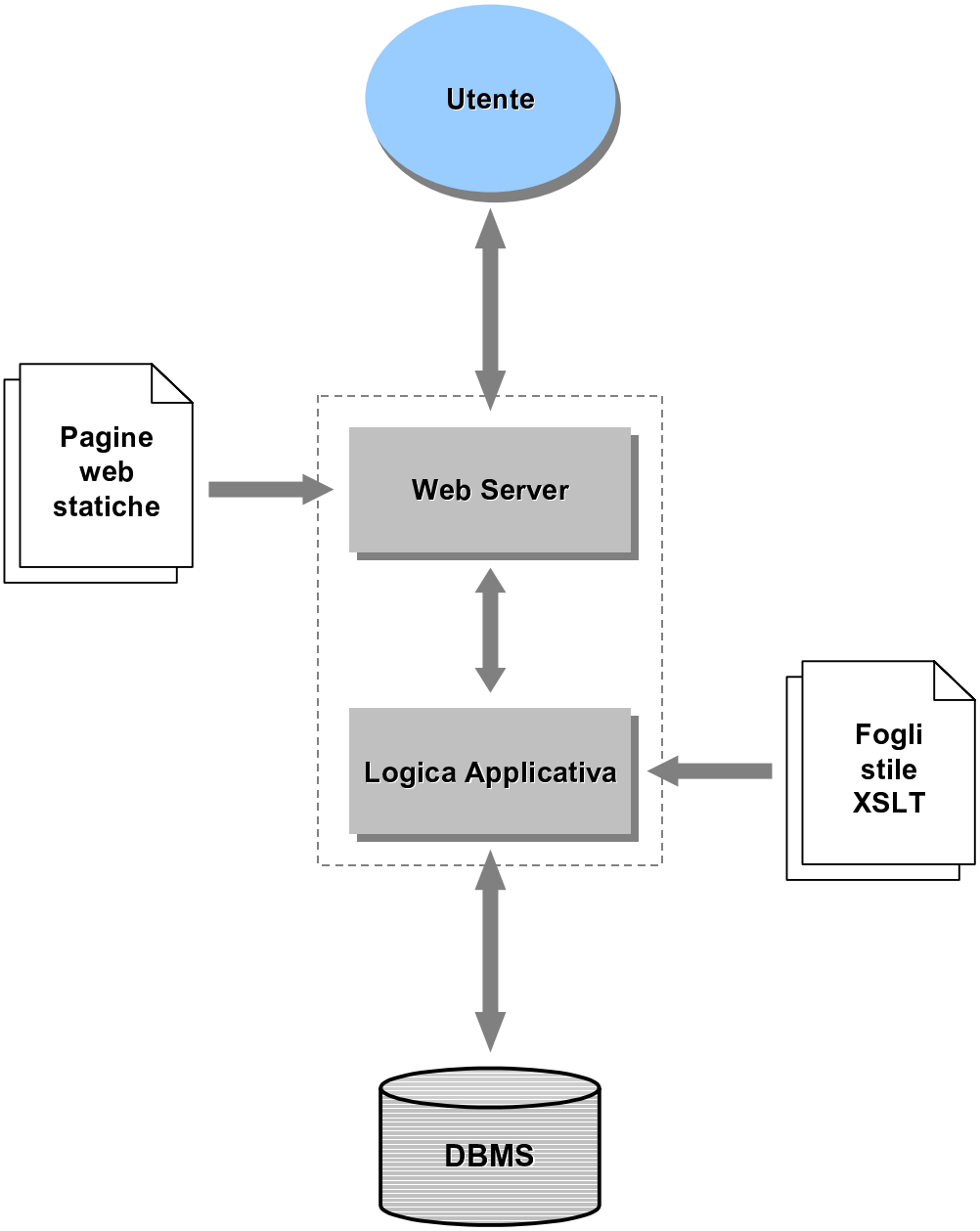


Figura 3.1: Impostazione generale

Passiamo ora ad analizzare con maggiore dettaglio la soluzione architetturale proposta. La figura 3.2 illustra l'articolazione del modulo della logica applicativa e descrive in modo particolareggiato la sequenza di operazioni necessarie per la produzione di una pagina dinamica. Tale sequenza viene descritta nel seguito.

1. La richiesta provoca l'invocazione dell'apposita servlet, la quale, interagendo con il database attraverso il protocollo JDBC, recupera le informazioni che andranno a formare il contenuto dinamico della pagina.

2. Il risultato dell'interrogazione viene formattato in XML e viene passato all'XSLT Processor.

3. L'XSLT Processor applica al documento XML la trasformazione XSLT appropriata e produce cosìla pagina web finita, pronta per essere inviata all'utente attraverso il Web server.

Per individuare il corretto foglio di stile da applicare, l'applicazione riceve tale indicazione come parametro passato dal client nell'URL di richiesta. È bene inoltre prevedere la possibilità di definire degli stylesheet di default, che vengono utilizzati nel caso in cui non venga effettuata una scelta.

La trasformazione con i fogli di stile non deve necessariamente avvenire sul server: esistono browser che già ora supportano le specifiche XSL, sia pure in modo incompleto.

In futuro, la totalità dei web browser sarà in grado di trattare i file XML direttamente. Attualmente, soltanto alcuni browser di ultima generazione supportano questa caratteristica, come le versioni 5.0 e 5.5 di Microsoft Internet Explorer ed il nuovo Netscape Navigator 6.

In presenza di browser *XML-enabled*, quindi, si può pensare di consegnare loro direttamente il documento XML, delegando al client il compito di effettuare la trasformazione XSLT richiesta.

Tale soluzione presenta però alcuni limiti che la rendono per il momento sconsigliabile.

Innanzitutto, occorre tenere presente che la tecnologia XSL, pur essendo avviata verso una standardizzazione definitiva è ancora in fase di sviluppo, e la specifica di XSLT è stata standardizzata soltanto di recente tramite la sua promozione a *W3C Recommendation* dal World Wide Web Consortium ([6]). Per questo motivo, i diversi prodotti software attualmente disponibili forniscono implementazioni parziali, basate su differenti versioni delle bozze di lavoro dello standard.

Appare evidente quindi la difficoltà di creare fogli di stile compatibili con le diverse implementazioni. La trasformazione effettuata sul server, utilizzando

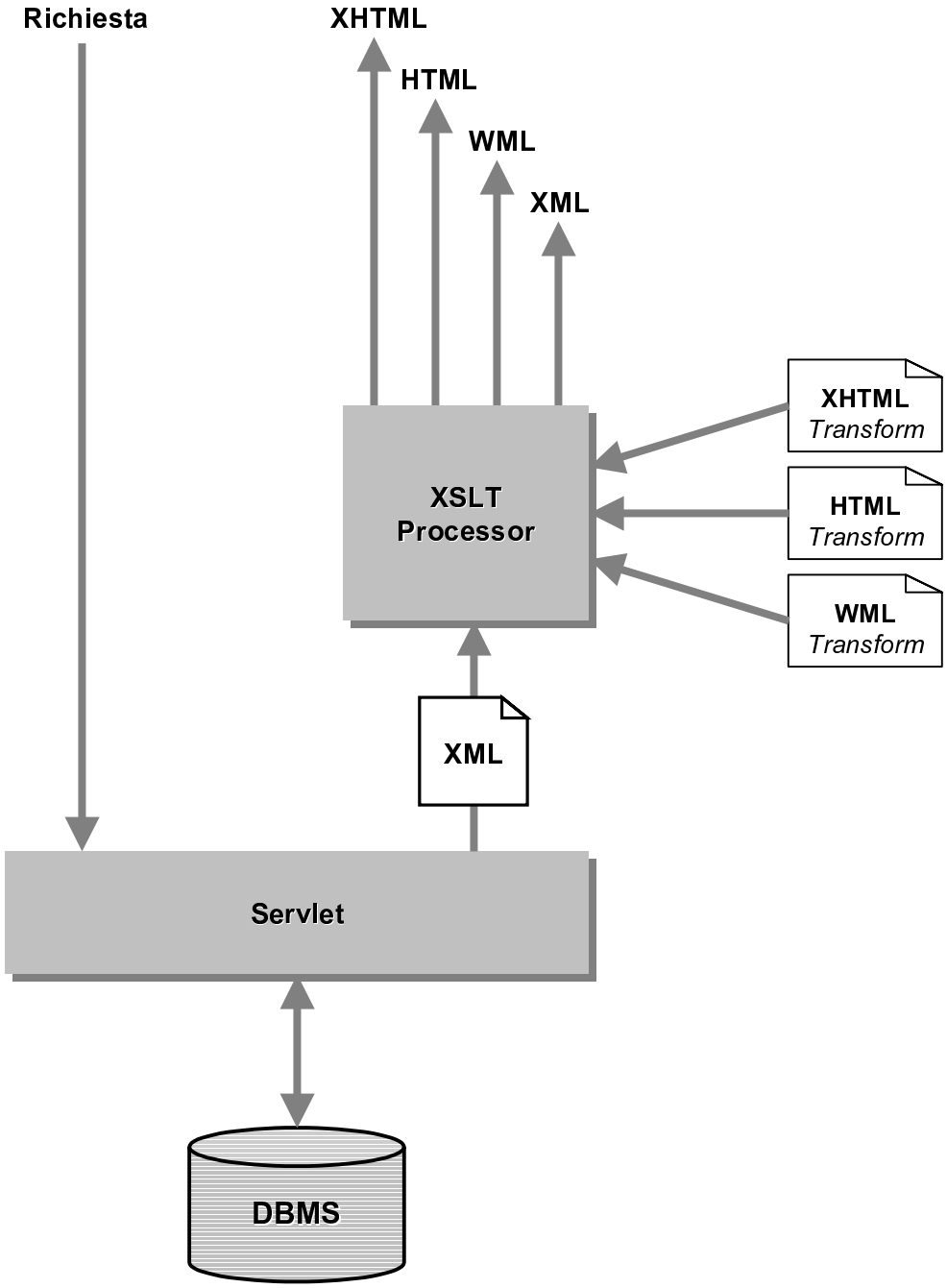


Figura 3.2: L'architettura nel dettaglio

un'implementazione di XSLT nota, resta dunque la soluzione per il momento più logica.

In secondo luogo, lo sgravio sull'occupazione di CPU del server che si ottiene spostando la trasformazione sul client viene controbilanciato dal maggior consumo di larghezza di banda necessario al trasferimento verso il client dell'accoppiata documento XML + foglio di stile.

3.4.4 Vantaggi della soluzione architetturale proposta

Dopo aver già discusso dei vantaggi principali, esaminiamo altri pregi di questa soluzione.

Più visualizzazioni dei dati

Si nota immediatamente un grande pregio di questa architettura. Grazie alla separazione fra contenuto e presentazione, è possibile ottenere più visualizzazioni diverse a partire dallo stesso insieme di dati, variando solamente il foglio di stile. Le visualizzazioni possono quindi essere create a piacere, in base alle esigenze.

Supporto di diversi tipi di output

La notevole flessibilità garantita da questo approccio rende inoltre molto agevole il supporto di linguaggi di markup diversi. Per realizzare una versione del sito compatibile con il protocollo WAP adottato dai telefoni cellulari, ad esempio, basta creare un apposito insieme di fogli di stile che trasformino il contenuto in pagine WML.

Personalizzazione della presentazione

La caratteristica sopra enunciata può anche essere usata per attuare una personalizzazione della presentazione più spinta, applicando trasformazioni diverse in base al browser usato. La servlet che gestisce la richiesta è infatti in grado di riconoscere il tipo di browser del client e quindi un client che utilizza Internet Explorer può ricevere una presentazione diversa rispetto ad un client che utilizza Netscape Navigator. Con le continue guerre tra le diverse implementazioni di HTML, DHTML e JavaScript supportate dai due concorrenti, questa caratteristica può risultare comoda.

Portabilità e standardizzazione

La scelta di utilizzare il linguaggio Java non vincola l'implementazione ad una specifica piattaforma, ma ne consente la portabilità su diverse architetture hardware.

L'adozione di XML, poi, costituisce una scelta importante: significa utilizzare uno standard aperto, flessibile, universalmente accettato e supportato, destinato ad assumere una sempre maggiore rilevanza sia per il Web che per lo scambio di dati.

Come considerazione finale, si fa notare che l'utilizzo del linguaggio XML come parte integrante dell'architettura del sito apre possibilità interessanti nel campo della ricerca e dell'integrazione di informazioni provenienti da sorgenti eterogenee. La possibilità di interagire con il sito ricevendo direttamente un output XML rappresenta sicuramente una porta privilegiata per l'analisi delle informazioni contenute nel sito web stesso.

Capitolo 4

Realizzazione del prototipo

Questo capitolo illustra la fase di implementazione del prototipo del sito web che è stato realizzato.

La prima parte si occupa della descrizione degli strumenti utilizzati per lo sviluppo del prototipo.

Nella seconda parte viene illustrata la creazione della struttura di base dell'applicazione.

La terza parte descrive, infine, l'implementazione di una parte delle funzionalità del sito web relative agli aspetti didattici.

4.1 Gli strumenti utilizzati

Dovendo realizzare un'applicazione web dotata di servlet Java, gli strumenti software che si rendono necessari sono i seguenti:

- Un Web server
- Un *servlet engine* che permetta al Web server di eseguire le servlet
- L'ambiente di sviluppo Java

Per quanto riguarda lo sviluppo in Java, è stato utilizzato il JDK (Java Development Kit) 1.2.2, corredato dell'ambiente di sviluppo dedicato alle servlet, il JSDK (Java Servlet Developmente Kit) versione 2.1.

Considerando il fatto che la realizzazione che si vuole effettuare ha carattere di prototipo, si è scelto di usare il web server fornito con il JSDK, il quale dispone ovviamente di supporto per le servlet. Tale strumento, oltre ad essere

perfettamente adeguato alle necessità, presenta inoltre il grande vantaggio di essere configurabile a piacere senza richiedere l'intervento dell'amministratore di sistema della macchina sulla quale si lavora.

Per funzionare, il Web server del JSDK richiede l'assegnazione di una porta di comunicazione. Tale porta non deve essere quella di default del protocollo HTTP (8080), per non entrare in conflitto con il Web server permanente installato sulla macchina. Si è quindi scelta, in maniera del tutto arbitraria, la porta 8078. La prima parte delle URL indirizzate al prototipo dovrà quindi avere la seguente forma:

```
http://nomedelserver.it:8078
```

4.2 Implementazione della struttura di base

4.2.1 XSLT Processor

La prima fase dell'attività di creazione dell'architettura è consistita nella definizione del modulo responsabile delle trasformazioni XSLT. Dal momento che il progetto e la realizzazione di un software cosìcomplesso esulano dagli scopi del presente lavoro di tesi, si è deciso di utilizzare una delle numerose implementazioni esistenti. Dopo una fase di confronto, la scelta è caduta su XT, di James Clark [7] (coautore dello standard XSL.)

Scritto totalmente in Java, XT è un prodotto molto diffuso nella comunità XML, con buone performances, una buona stabilità ed un costante supporto da parte del suo autore.

XT è liberamente scaricabile; la sua licenza d'uso, decisamente generosa, permette di utilizzare e modificare liberamente il codice, a patto di mantenere le informazioni di copyright.

La versione utilizzata, la 19991105, implementa le specifiche XSLT nella loro versione PR-xslt-1991008. Questa informazione è importante per la compatibilità con i fogli di stile che verranno realizzati.

Per funzionare, XT necessita la presenza di un parser XML scritto in Java, che supporti l'API standard SAX. La scelta più naturale è stata quella di utilizzare il parser realizzato dallo stesso autore, ovvero XP. [8]

Una volta che le distribuzioni di XT e XP sono state inserite in una directory appartenente al CLASSPATH, il processore XSLT è operativo.

XT può essere utilizzato tramite invocazione da linea di comando. Un tale tipo

di utilizzo può servire per testare od effettuare il debugging dei fogli di stile. Il comando da usare è il seguente:

```
java com.jclark.xsl.sax.Driver pippo.xml pippo.xsl pippo.html
```

dove *`pippo.xml`* è il documento XML da trasformare, *`pippo.xsl`* è il foglio di stile e *`pippo.html`* è il risultato della trasformazione.

Dopo aver completato il setup dell'XSLT processor, il passo successivo consiste nell'integrazione di questo modulo nell'architettura dell'applicazione.

4.2.2 La servlet `XsltServlet`

Affinchè le tutte le servlet dell'applicazione possano usufruire della trasformazione XSLT, occorrerebbe invocare manualmente XT dall'interno di ognuna di esse. Il codice necessario all'invocazione, però, è abbastanza complicato e rischia di appesantire il codice delle servlet, peggiorandone la leggibilità. Inserire queste porzioni di codice sempre uguale in ogni servlet, poi, costituisce una seccante ripetizione.

Per queste ragioni, si è cercato un approccio più elegante al problema [9]. Si è deciso di creare una servlet speciale, denominata `XsltServlet`, che contiene il codice necessario all'invocazione di XT. Tutte le altre servlet vengono dichiarate come sottoclassi di `XsltServlet`, invece di `HttpServlet` come sarebbe usuale.

In questo modo, le servlet ereditano i metodi necessari all'invocazione di dell'XSLT processor. I metodi stssi, inoltre, sono organizzati in modo da rendere il postprocessing effettuato da XT del tutto trasparente alle servlet stesse.

Il diagramma in figura 4.1 illustra la gerarchia che si viene a creare ed evidenzia i principali metodi delle classi.

Esaminiamo ora il funzionamento della servlet.

La servlet accetta richieste di tipo GET e di tipo POST, che vengono trattate rispettivamente dai due metodi `doGet()` e `doPost()`. Tali metodi, in realtà, non fanno altro che inoltrare le richieste al metodo `doGetPost()`.

`doGetPost()` esiste in due versioni:

- ```
  protected void doGetPost(HttpServletRequest req,
                           HttpServletResponse resp)
  ```
  ```

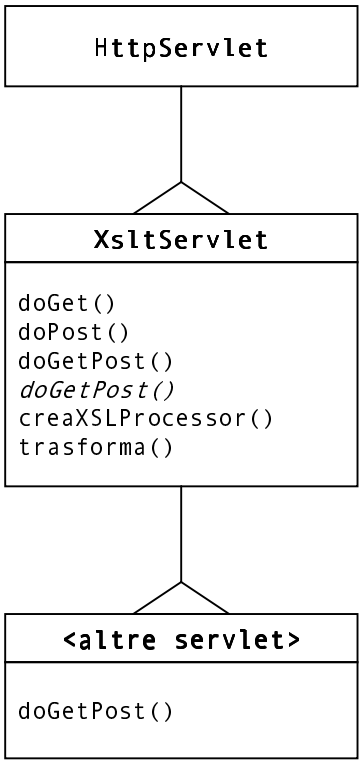


Figura 4.1: Gerarchia delle servlet

- `protected abstract String doGetPost(HttpServletRequest req, HttpServletResponse resp, Writer w)`

La prima versione, a due parametri, è implementata direttamente in `XstlServlet`.

Il secondo `doGetPost()`, a tre parametri, è dichiarato come *abstract* e deve essere obbligatoriamente ridefinito dalle servlet discendenti (nel diagramma è evidenziato in corsivo).

Le richieste vengono passate al `doGetPost` a due parametri, che è il vero regista dell'esecuzione. Questo metodo effettua, in sequenza, le seguenti operazioni:

- Innanzitutto, recupera il nome dello stylesheet eventualmente passato come parametro nell'URL. È possibile infatti scegliere dall'esterno il foglio di stile da usare per la trasformazione, usando un URL del tipo:

`http://www.nomeserver.it/NomeServlet?xsl=pippo.xsl`

dove `pippo.xsl` rappresenta in nome del foglio di stile da usare.

- Viene poi invocato il metodo `doGetPost` ridefinito dalla servlet. È qui che vengono effettuate le operazioni vere e proprie della servlet. Tipicamente, la servlet apre una connessione con il database, effettua una query, formatta i dati risultanti in XML e si disconnette dal database.

  Al metodo vengono passati tre parametri. I primi due sono gli usuali oggetti di tipo `HttpServletRequest` e `HttpServletResponse`, per gestire la richiesta e la corrispondente risposta. Il terzo è un oggetto di tipo `StringWriter` (uno stream di caratteri), nel quale il metodo deve inserire l'output in XML affinché sia processato con il foglio di stile. L'output, quindi, non deve essere passato direttamente nell'oggetto di tipo `HttpServletResponse`, a differenza di quanto avviene con l'utilizzo tradizionale delle servlet.

  Questo metodo restituisce il nome del foglio di stile di default, che viene utilizzato qualora non si sia utilizzato il parametro `xsl`.

- Una volta che il metodo `doGetPost()` della servlet ha finito, il controllo ritorna al `doGetPost()` di `XsltServlet`, che provvede a lanciare la trasformazione XSLT allo scopo di produrre la pagina web finale.

  Per fare questo, il metodo si avvale dei metodi `creaXSLProcessor()` e `trasforma()` i quali contengono il codice necessario all'invocazione di XT.

Come si è visto, quindi, il processo di trasformazione XSLT viene reso del tutto trasparente. Nello scrivere una servlet non occorre fare altro che sovrascrivere il metodo `doGetPost()`, compiere tutte le operazioni necessarie all'interazione con il database, e restituire infine l'output XML tramite lo stream di caratteri passato come parametro.

La servlet incorpora un'ulteriore caratteristica. Impostando il valore del parametro `xsl` a `none`, il passaggio di trasformazione XSLT viene saltato e viene restituito direttamente il documento XML.

4.2.3 La classe `DbOps`

Questa classe è stata creata per raccogliere alcune operazioni di frequente utilizzo relative all'interazione con il database. Usufruendo dei metodi di tale classe, il codice delle servlet si mantiene più pulito e più comprensibile.

Vediamo i metodi che la classe presenta:

- `openDbConnection()`

  Questo metodo serve per creare una connessione JDBC con il DBMS Oracle. Restituisce un oggetto `Connection`, che rappresenta la connessione creata.

- `closeDbConnection()`

  Questo metodo chiude la connessione con il database, ripulendo nel contempo gli oggetti `Statement` e `ResultSet` creati durante l'interazione con il database stesso.

- `toXML()`

  Metodo che serve per effettuare la formattazione in XML di un insieme di tuple risultanti da una query. Lo schema del documento XML prodotto è fisso e rispecchia la struttura tabellare del risultato della query. Questo metodo è quindi indicato per un uso generico; per costruire documenti XML più articolati occorre effettuare una trasformazione ad hoc. L'aspetto dell'XML che si ottiene è il seguente:

```
<?xml version ="1.0"?>
<nome-tabella>
  <riga1>
    <colonna1>valore</colonna1>
    <colonna2>valore</colonna2>
    ⋮
    <colonnan>valore</colonnan>
  </riga1>
  <riga2>
    <colonna1>valore</colonna1>
    <colonna2>valore</colonna2>
    ⋮
    <colonnan>valore</colonnan>
  </riga2>
  ⋮
  <rigam>
    <colonna1>valore</colonna1>
    <colonna2>valore</colonna2>
    ⋮
    <colonnan>valore</colonnan>
  </rigam>
<nome-tabella>
```

Parametri del metodo:

- `ResultSet rs`
  Oggetto contenente il risultato della query
- `Writer w`
  Stream di caratteri sul quale viene scritto l'output XML
- `String nometab`
  Nome da utilizzare per la coppia di tag che racchiude l'intero documento prodotto
- `String nomeitem`
  Nome da utilizzare per i tag che definiscono le singole tuple

I nomi delle colonne vengono ricavati dal *ResultSet*.

4.3 Implementazione delle funzionalità del sito

In questa sezione si esaminano le funzionalità del sito web che sono state realizzate.

4.3.1 Pagina principale della didattica

Figura 4.2: Pagina principale della didattica

Accedendo all'area del sito relativa alla didattica, si trova la pagina di figura 4.2. Tale pagina presenta una visione d'insieme dell'offerta didattica della Facoltà, visualizzando l'insieme dei corsi di studio, dei master, delle scuole di specializzazione e dei corsi di dottorato disponibili.

I corsi di studio sono divisi in base al tipo di ordinamento.

4.3.2 Pagine dei Corsi di Studio

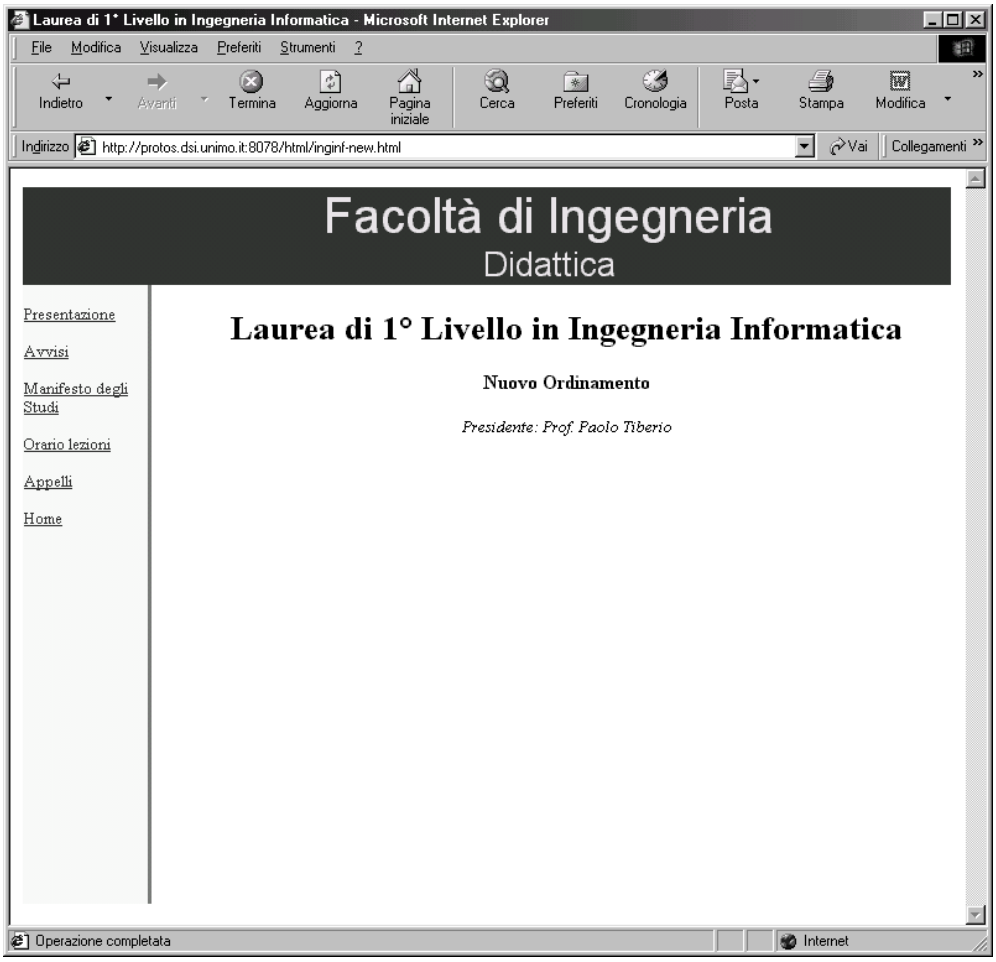


Figura 4.3: Pagina di un Corso di Studio

Dalla pagina principale, si accede alle sezioni relative ai singoli corsi di studio. In figura 4.3 si vede ad esempio la pagina dedicata al Corso di Laurea di 1^o Livello in Ingegneria informatica.

Il menu a sinistra della pagina consente di accedere alle diverse opzioni disponibili.

4.3.3 Presentazione di un Corso di Studio

Selezionando il link “Presentazione”, si arriva alla pagina che mostra un testo di presentazione. Il contenuto dinamico della pagina viene generato dalla servlet `PresentazioneCorso`.

Il fatto che la presentazione del corso sia disponibile sia in italiano che in inglese costituisce un ottimo esempio per dimostrare la flessibilità dell’architettura.

La servlet recupera dal database i nomi dei file contenenti le due versioni (italiana ed inglese) del testo di presentazione relativo al corso di studio in questione. Questi file vengono letti e inseriti in un unico documento XML, che contiene cosìentrambe le versioni del testo. A questo punto, è compito del foglio di stile selezionare la versione da visualizzare. Si creano quindi due fogli di stile XSL, uno relativo alla pagina in italiano e l’altro relativo alla pagina in inglese. Le figure 4.4 e 4.5 illustrano il risultato.

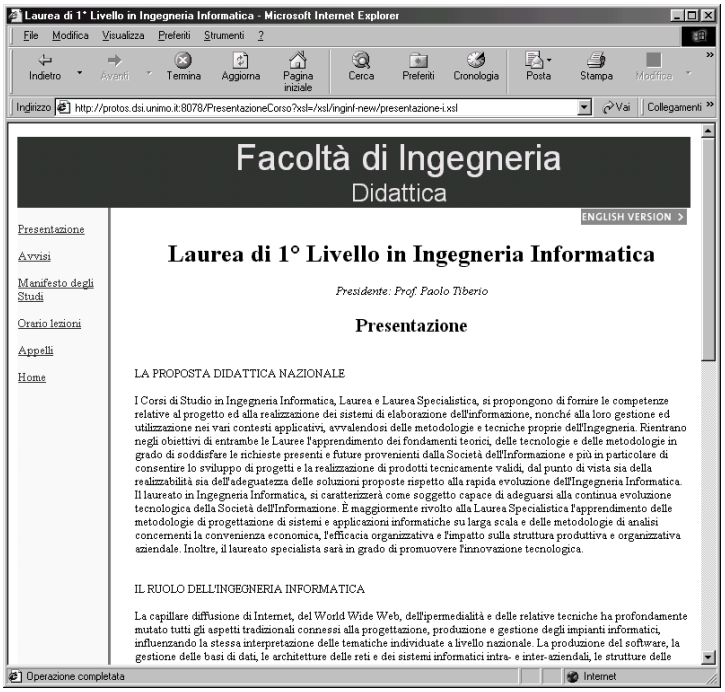


Figura 4.4: Presentazione di un corso di studio (italiano)

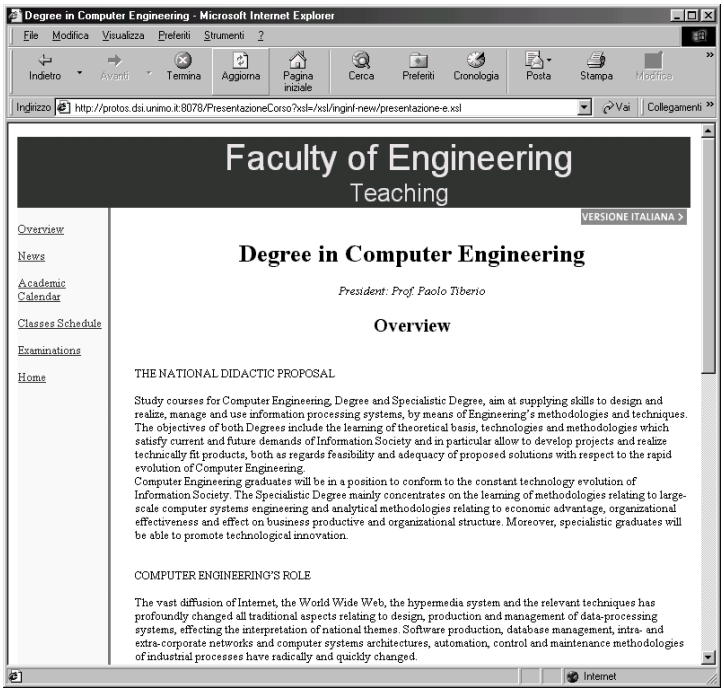


Figura 4.5: Presentazione di un corso di studio (inglese)

4.3.4 Manifesto degli studi

La pagina visualizza, di default, il Manifesto degli Studi dell'anno accademico in corso. E' possibile comunque scegliere, da una lista, gli anni accademici precedenti.

Per visualizzare il Manifesto degli Studi, la servlet `ManifestoStudi`, che gestisce questa operazione, deve ricevere due parametri (passati nell'URL):

- `cds` identifica il Corso di Studio al quale il Manifesto si riferisce
- `annoacc` seleziona l'anno accademico prescelto

Grazie a questi due elementi, la servlet può interrogare il database ed ottenere le informazioni volute. La query che viene effettuata interessa le due tabelle INSEGNAMENTO ed INSERITO_IN:

```
SELECT AnnoCorso, Periodo, Nome, Obb_Opz, CFU
FROM Insegnamento i1, Inserito_In i2
```

```
WHERE i1.CodIns = i2.CodIns
AND i2.CodCorso = codcorso
AND i2.AnnoAcc = annoacc
ORDER BY AnnoCorso, Periodo
```

Il risultato di quest'interrogazione è una serie di tuple, ognuna riguardanti un insegnamento, ordinate per anno di corso e per periodo didattico.

Si deve ora formattare questo risultato in XML. In questo caso non viene utilizzato il metodo `toXML()`, dal momento che si vuole ottenere un output XML più articolato. Di seguito è riportato un frammento del risultato prodotto dall'algoritmo di formattazione della servlet:

```
<manifesto annoacc="2000/2001">
<anno id="1">
  <periodo id="1">
    <insegnamento>
      <nome>Fondamenti di Informatica A</nome>
      <obb_opz>Obb</obb_opz>
      <cfu>3</cfu>
    </insegnamento>
    <insegnamento>
      <nome>Analisi Matematica A</nome>
      <obb_opz>Obb</obb_opz>
      <cfu>3</cfu>
    </insegnamento>
    <insegnamento>
      <nome>Inglese A</nome>
      <obb_opz>Obb</obb_opz>
      <cfu>3</cfu>
    </insegnamento>
  </periodo>
  <periodo id="2">
    <insegnamento>
      <nome>Fondamenti di Informatica B</nome>
      <obb_opz>Obb</obb_opz>
      <cfu>6</cfu>
    </insegnamento>
    <insegnamento>
      <nome>Analisi Matematica B</nome>
      <obb_opz>Obb</obb_opz>
      <cfu>6</cfu>
```

```
    </insegnamento>
    <insegnamento>
      <nome>Chimica</nome>
      <obb_opz>Obb</obb_opz>
      <cfu>6</cfu>
    </insegnamento>
  </periodo>
  ⋮
</anno>
<anno id="2">
⋮

</manifesto>
```

Vediamo ora il foglio di stile che converte questo output XML in una pagina web completa:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/TR/REC-html40">

<xsl:output method="html"/>

<xsl:template match="/">
<html>
<head>
  <title>Laurea di 1&#176; Livello in Ingegneria Informatica</title>
  <meta name="GENERATOR" />
  <meta http-equiv="Content-Type" content="text/html" />
</head>

<body bgcolor="#FFFFFF">
<!-- Qui c'è il codice XHTML responsabile del contenuto
     statico della pagina -->
 ⋮
        <xsl:apply-templates/>
 ⋮
</body>
</html>
```

```
</xsl:template>

<xsl:template match="anno">
   <br /><br />
   <table cellpadding="10" width="100%">
      <tr>
         <td bgcolor="#DEDEDE" width="100%">
            <center><b>
               <xsl:value-of select="@id"/>&#176; Anno
            </b></center>
         </td>
      </tr>
      <xsl:apply-templates/>
   </table>
</xsl:template>

<xsl:template match="periodo">
   <tr>
      <td>
         <b>Periodo <xsl:value-of select="@id"/></b> <br />
         <xsl:apply-templates/>
      </td>
   </tr>
</xsl:template>

<xsl:template match="nome">
   <xsl:apply-templates/>
</xsl:template>

<xsl:template match="obb_opz">
   (<xsl:apply-templates/> -
</xsl:template>

<xsl:template match="cfu">
   <xsl:apply-templates/> CFU) <br />
</xsl:template>

</xsl:stylesheet>
```

La pagina web risultante è mostrata in figura 4.6.

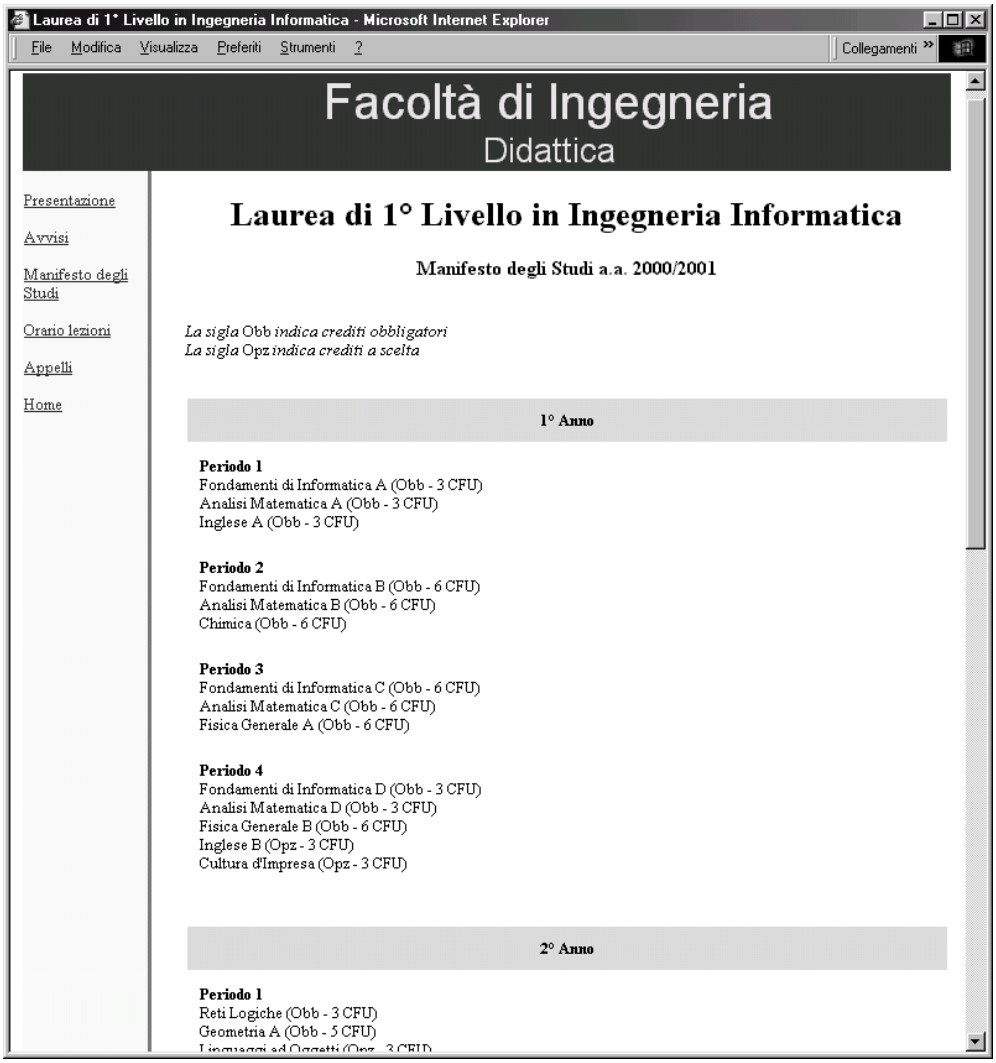


Figura 4.6: Manifesto degli Studi

Conclusioni e sviluppi futuri

La presente tesi si è svolta nell'ambito del progetto del sito Web della Facoltà di Ingegneria dell'Università di Modena e Reggio Emilia.

Gli obiettivi proposti sono stati raggiunti:

- è stato portato a termine il progetto del database necessario alla gestione dei dati riguardanti il supporto della didattica. L'attività di progettazione è stata inoltre seguita dall'implementazione dello schema realizzato;
- è stata proposta una soluzione architetturale innovativa e flessibile, basata sull'utilizzo dello standard XML insieme alla tecnologia dei fogli di stile XSL e delle Servlet Java. Si è visto come l'architettura proposta riesca a realizzare una netta separazione fra contenuto, presentazione e logica applicativa, con benefici effetti sulla facilità di manutenzione, sull'espandibilità e sul supporto dei nuovi standard del Web senza che questo vada a discapito della massima compatibilità con i browser oggi maggiormente diffusi;
- è stato realizzato un prototipo che implementa alcune delle funzionalità previste per il nuovo sito e che mette in luce le qualità della soluzione architetturale utilizzata.

Lo standard XML è assai potente e flessibile ed offre numerosi vantaggi. L'interesse ed il supporto verso questo linguaggio da parte della comunità degli sviluppatori sono decisamente notevoli, in virtù delle sue innegabili qualità. Gli utilizzi possibili sono innumerevoli e non legati soltanto al mondo della pubblicazione di documenti sul Web.

Il linguaggio per fogli di stile XSL, pur non avendo ancora raggiunto la piena stabilità per la mancanza di una standardizzazione definitiva da parte del World Wide Web Consortium, ha dimostrato di avere delle grandi potenzialità. Esso fornisce, insieme ad XML, quella separazione vera tra contenuto e presentazione che può rivoluzionare il mondo del Web.

Gli sviluppi futuri che si propongono riguardano essenzialmente due direzioni.

Forse l'unico vero, grande svantaggio dell'architettura proposta è costituito dalle prestazioni. Il processo di trasformazione XSLT è indubbiamente pesante dal punto di vista computazionale, considerando anche il fatto che il software che lo realizza è scritto in Java, notoriamente debole sotto questo punto di vista. Occorre dire che durante lo sviluppo del prototipo le prestazioni osservate sono sempre state molto buone, cosa naturale dal momento che l'architettura non doveva servire richieste concorrenti. Sarebbe quindi utile effettuare una valutazione sulla scalabilità delle prestazioni dell'architettura, con riferimento ad un utilizzo reale in cui il server Web deve smaltire molte richieste contemporanee.

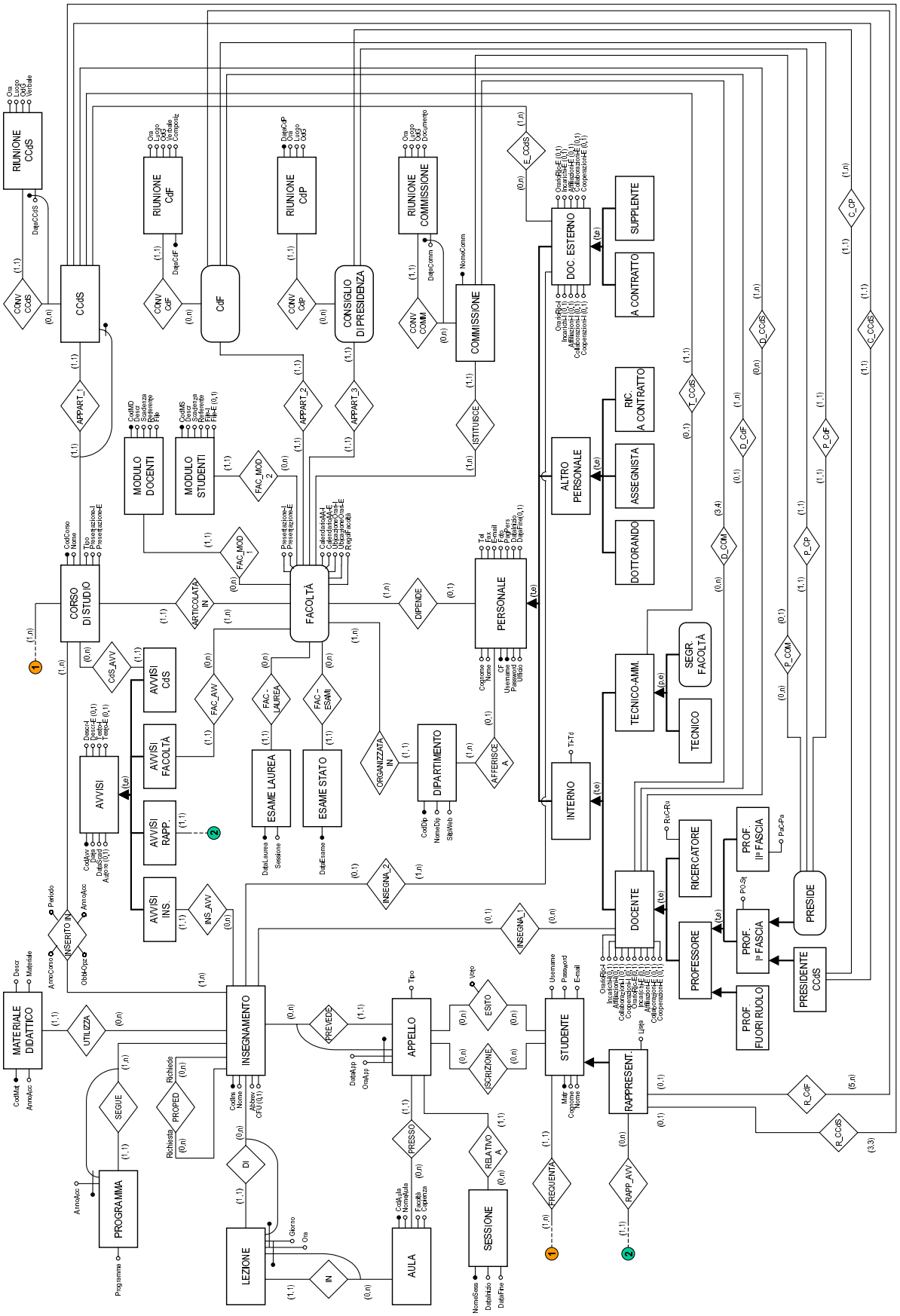
Un'altro interessante sviluppo potrebbe infine essere quello di realizzare una versione del sito in WML, allo scopo di sperimentare la flessibilità offerta dall'architettura nel supporto dei vari linguaggi di markup.

Appendice A

Schema E/R globale

Viene presentato lo schema E/R globale del database, risultato del lavoro di integrazione dei due schemi concettuali prodotti in questo lavoro di tesi e nella tesi [1].

Lo schema globale è consultabile anche online all'indirizzo: `http://sparc20.dsi.unimo.it/SitoFacolta/er.html`


RIUNIONE CCdS
CONV CCdS
CCdS
APPART_1
RIUNIONE CdF
CONV CdF
CdF
APPART_2
RIUNIONE CdP
CONV CdP
CONSIGLIO DI PRESIDENZA
APPART_3
RIUNIONE COMMISSIONE
CONV COMM
COMMISSIONE
ISTITUISCE
E_CCdS
DOC. ESTERNO
SUPPLENTE
A CONTRATTO
C_CdP
C_CCdS
MODULO DOCENTI
MODULO STUDENTI
FAC_MOD_2
FAC_MOD_1
CORSO DI STUDIO
ARTICOLATO IN
FACOLTÀ
DIPENDE
PERSONALE
ALTRO PERSONALE
ASSEGNISTA
DOTTORANDO
RIC. A CONTRATTO
T_CCdS
D_CCdS
D_CdF
D_COM
P_COM
P_CdF
P_CP
CdS_AVV
AVVISI CdS
FAC_AVV
AVVISI FACOLTÀ
AVVISI RAPP.
AVVISI INS
INS_AVV
AVVISI
FAC-LAUREA
FAC-ESAMI
ESAME LAUREA
ESAME STATO
ORGANIZZATA IN
DIPARTIMENTO
AFFERISCE A
TECNICO-AMM.
TECNICO
SEGR. FACOLTÀ
INTERNO
DOCENTE
RICERCATORE
PROFESSORE
PROF. I^a FASCIA
PROF. FUORI RUOLO
PRESIDE
PRESIDENTE CCdS
INSERITO IN
MATERIALE DIDATTICO
UTILIZZA
INSEGNAMENTO
INSEGNA_1
INSEGNA_2
PREVEDE
APPELLO
ESITO
ISCRIZIONE
STUDENTE
RAPPRESENT.
FREQUENTA
RAPP_AVV
R_CdF
R_CCdS
SEGUE
PROGRAMMA
PROPED
DI
LEZIONE
IN
AULA
PRESSO
RELATIVO A
SESSIONE

Appendice B

Tecnologie per l'integrazione Web - Database

Un'applicazione Web-database può essere creata in molti modi differenti; in tutti però possiamo riconoscere degli elementi comuni. Questi componenti (o strati) costituiscono l'architettura dell'applicazione.

In generale possiamo individuare quattro strati:

- Strato del browser
- Strato logico dell'applicazione
- Strato di connessione al database
- Strato del database

Il livello del browser. Il browser è il Client di un'applicazione Web-database. Il browser gestisce il rendering e la visualizzazione del codice HTML, l'esecuzione delle funzionalità estese come JavaScript, ActiveX, e Java. Entrambi i browser più diffusi, Netscape Navigator e Microsoft Internet Explorer supportano JavaScript, anche se Explorer solo parzialmente (nelle ultime versioni si arriva ad un supporto del 99%). Solo Explorer supporta ActiveX, tecnologia proprietaria Microsoft. Ciascuno dei due browser supporta Java, anche se le implementazioni differiscono leggermente, poiché solo Netscape ha sposato la campagna della Sun Microsystems "100% Pure Java".

Livello logico dell'applicazione. In un'applicazione Web-database, questo livello può esistere sotto forma di programma CGI, programma Server API, moduli lato Server, browser Plug-In, Servlet Java o Applet Java. La logica dell'applicazione è la componente con la quale lo sviluppatore spende la maggior parte del

tempo. Questo livello riceve e processa i dati, li formatta e spedisce la query al database, recupera i risultati e genera il codice HTML di cui il browser si occupa del corrispondente rendering.

Livello della connessione al database. Lo strato di connessione a DB fornisce il collegamento fra il livello logico ed il DBMS. Esistono diverse forme di connessioni: protocolli di rete per DBMS, librerie API e librerie di classi, programmi che sono essi stessi i Client del DataBase. I prodotti che forniscono connettività a DB devono soddisfare varie esigenze: devono essere facili da usare, flessibili, veloci, robusti ed affidabili. Soluzioni diverse si concentrano su aspetti differenti: l'obiettivo principale di un sistema basato sulla coppia HTML/SQL Template e Parse, è una prototipazione rapida e facile per lo sviluppo di siti che accedono dinamicamente a DB. La tecnologia dei Template è però, più lenta rispetto a soluzioni con API native, poiché si evitano gli overhead del parsing del Template. D'altro canto, queste ultime guadagnano in velocità e flessibilità , rinunciando ad avere Tool che permettono uno sviluppo facilitato e ad alto livello.

Livello database. Questo livello si occupa di tutte le classiche funzionalità di un DBMS: memorizzazione fisica dei dati su disco, gestione intelligente ed ottimizzazione delle query definite dall'utente, gestione della sicurezza. Spesso il DataBase, più che una scelta è un vincolo iniziale. Molte aziende desiderano trasformare le loro applicazioni legacy, magari con DB di diversi venditori e piattaforme, in applicazioni Web-based. La capacità di combinare dati eterogenei in un'unica interfaccia, è l'essenza del Web, e l'apertura di questa tecnologia offre numerose soluzioni per le più diverse esigenze.

B.1 Architetture per l'integrazione Web-database

Differenti soluzioni architetturali si distinguono a seconda del metodo attraverso il quale le applicazioni Web si interfacciano con il DataBase. Ciascuna presenta peculiarità proprie, che devono essere note allo sviluppatore per sfruttare al meglio la tecnologia più adatta.

B.1.1 Soluzioni *client-side*

Le soluzioni lato client comprendono due tipi di tecnologie: le estensioni del browser e le applicazioni esterne.

Estensioni del browser

Le estensioni del browser sono delle aggiunte (add-on) al nucleo del Web browser che ne migliorano ed aumentano le funzionalità standard di interprete HTML. Questa categoria comprende interpreti per JavaScript, ActiveX control, Java, e plug-in proprietari di varia natura. Talune di queste tecnologie sono proprietarie, quindi vincolano l'utente all'uso di uno specifico browser.

L'uso di queste tecnologie permette un aumento delle performance, in termini di diminuzione dei tempi di attesa e traffico di rete (occupazione di banda). Si può mantenere lo stato con l'uso di variabili, eliminando cosìla necessità di trasferire dati fra browser e Web Server. Si possono effettuare elaborazioni sui dati, eliminando ancora una volta il bisogno di spedire i dati al Server. Si possono creare GUI per interazioni particolarmente sofisticate e gradevoli.

L'aumento delle performance è pagato con lo scotto di avere applicazioni meno robuste: se un Client esce accidentalmente o deliberatamente dall'applicazione, lo stato è perso. Le elaborazioni a lato Client fanno diminuire, spesso drasticamente, il numero di contatti al Server: questo tecnicamente è sicuramente un grosso vantaggio, però a volte è visto come una debolezza dagli esperti di politiche marketing di un'azienda.

I Plug-In sono librerie linkate dinamicamente che permettono la gestione di MIME-Type non standard. Essendo compilati nativamente per una data piattaforma, offrono performance ragguardevoli, ma richiedono di essere installati. Questa considerazione è estremamente pesante nel caso di Intranets di grandi dimensioni: il tempo di scarico del codice e installazione si moltiplica per gli n utenti, e con esso i costi associati. I Plug-In sono dipendenti dalla piattaforma: ciò significa che gli sviluppatori sono costretti a scrivere versioni differenti (porting) per ciascun ambiente target, e che una modifica deve poi essere riversata su ciascuna piattaforma supportata. Le nuove versioni di Explorer e Navigator presentano lo stesso tipo di gestione dei Plug-In, in questo modo gli sviluppatori si devono preoccupare di fare il porting solo per piattaforme diverse, evitando quello per browser diversi sulla stessa piattaforma.

Applicazioni esterne

Le applicazioni esterne sono tipicamente Client dei DataBase tradizionali che sonolanciati dal browser sulla macchina Client durante l'esecuzione di una specifica applicazione Web (è necessario configurare il browser registrando un nuovo MIME- Type associato al nome dell'applicazione). L'uso di applicazioni esterne è un modo facile e veloce per portare on-line un legacy DataBase, ma il risultato che se ne ottiene non è né aperto, né portabile.

B.1.2 Soluzioni *server-side*

La maggior parte delle soluzioni oggi disponibili sono soluzioni Server-Side. ciò comporta che questa categoria di DataBase-gateway sia quella che presenta la più vasta varietà. Questo squilibrio di compiti fra Client e Server si spiega con la natura stessa del Web: un'architettura con un Fat Server ed un Thin Client. In quest'ottica, uno sviluppatore di DataBase-gateway non dovrebbe concentrarsi sulla macchina Client: questa in generale non possiede grandi risorse; si evitano procedure di installazione, processi di aggiornamento di quanto già installato e complesse attività di configurazione.

Ciascuno di questi sgradevoli inconvenienti, rinnega i punti di forza del mondo Web: una tecnologia che può essere sfruttata senza grandi sforzi, senza installazione di nuovo software, necessità di configurazioni, l'aggiornamento delle risorse avviene in un unico punto (e quindi diffuso in modo trasparente e senza costi aggiuntivi), con un'unica interfaccia gradevole e facile da usare per tutte le applicazioni Web.

CGI

Il protocollo CGI (Common Gateway Interface) stabilisce come vengono scambiate le informazioni fra browser e Web Server, che le rende disponibili ad un programma eseguito sulla macchina host. Essendo un protocollo, non una libreria di funzioni scritta per uno specifico Web Server, i programmi CGI sono indipendenti dal linguaggio: così, a patto di rispettare le specifiche del protocollo, possono essere scritti in C/C++, Perl, Java, ...

CGI rimane un modo molto diffuso per accedere a programmi residenti sul Server, per mezzo di un browser. La grossa spinta che ne decretò il loro successo fu proprio la possibilità di connessione a DataBase. Punti di forza della tecnologia CGI sono l'indipendenza dall'architettura del Web Server, indipendenza dal linguaggio, è uno standard aperto, offre una separazione del processo e soprattutto la semplicità del protocollo. Proprio quest'ultimo paradossalmente porta ad avere risultati non molto brillanti dal punto di vista delle performance, e ciò ha portato alla ricerca di soluzioni alternative. Un programma CGI può leggere i dati spediti tramite form HTML, formattarli in una query SQL, ed inviarli ad un DBMS a cui si è connesso. In modo analogo può costruire al volo delle pagine HTML inserendovi dati prelevati da DB.

Una tecnica comune per pubblicare il contenuto di un DataBase è quella dei Template: un Template è una pagina HTML in cui vi sono annegati dei tag non standard che permettono di prelevare dati da un DB. Ad un Template si accede tramite un URL del tipo: `http://nomehost/cgi-bin/Parser/ExtraPath/`

`TemplateFile.html?ExtraInfo` dove Parser è il programma CGI che legge il file Template e sostituisce dinamicamente i dati prelevati dal DB con le query specificate producendo un file in puro HTML.

HTTP Server API e Server Module

Le API e imoduli rappresentano l'equivalente lato Server delle estensioni al browser.

La prima tecnologia di accesso a DB via Web sono stati i gateway basati su CGI: si prelevano i dati spediti dalle pagine HTML e li si rende disponibili ad un programma a lato Server che garantisce la connessione al DB ed è esterno allo stesso Web Server. CGI richiede la generazione di un processo ogni qualvolta che viene fatta una richiesta, portando con sé i relativi overhead: spreco tempo di CPU, memoria aggiuntiva, ...

Il tema centrale di questo approccio invece, è che i programmi che accedono a DB coesistono con il Server HTTP, migliorando l'accesso a DB dal Web per mezzo di incremento di velocità, condivisione di risorse, e l'insieme di funzioni a disposizione. I programmi che usano le Server API girano come librerie caricate dinamicamente o come moduli, anziché come eseguibili separati. Un programma Server API viene di solito caricato alla prima volta che viene richiesta tale risorsa, in questo modo solo il primo utente che richiede la risorsa sarà costretto ad un'attesa più lunga. Per evitare questo inconveniente sulla prima instanziazione, si può forzare l'HTTP Server a caricare la risorsa all'istante del suo start-up (preloading).

In un programma CGI, ciascuna istanza possiede un proprio spazio di memoria, quindi talune risorse condivise da tutte le istanze (dati o codice di funzioni) vengono in realtà duplicate n volte. Un programma Server API condivide lo spazio con altre istanze di se stesso (thread) e con l'HTTP Server. Ciò significa che dati comuni possono essere richiesti da tutte le istanze che ne hanno bisogno, ma esistere solo in unico posto.

Questa tecnologia porta ad ottenere grossi vantaggi nelle prestazioni, ma al costo di un aumento della complessità: gli sviluppatori di programmi Server API devono agire sempre con molta cautela quando effettuano cambiamenti a variabili globali, usando se possibili chiamate ad opportune API a questo preposte.

Un programma CGI, accede ad una transazione Web solo in un numero limitato di punti: recupera i dati di una form dopo che il Server HTTP li ha ricevuti ed emette il codice HTML dopo che il processo di elaborazione è terminato. Nessun tipo di intervento è ad esempio possibile sullo schema di autenticazione, poiché l'autenticazione del protocollo HTTP avviene fuori dal reame del programma CGI. AL contrario, un programma Server API è strettamente collegato al Server, esiste in congiunzione o come parte di esso. Tali programmi possono modifica-

re, personalizzandolo, i metodi di autenticazione, di trasmissione crittografata, di login, rispetto a quelli di default.

I moduli dei costruttori di Web Server (Web Server Vendor Modules), sono moduli precostruiti scritti usando le Server API. Gli sviluppatori spesso acquistano Tool commerciali che aiutano o sostituiscono lo sviluppo di particolare funzionalità richiesta dall'applicazione. Esistono numerosi moduli che assistono differenti necessità, come particolari esigenze di sicurezza, transazioni economiche, verifica degli accessi, ...

Più le Server Api e i moduli sono strettamente legati ad un certo Server, più efficienti ed agevoli risulteranno le transazioni Web. Questo naturalmente si scontra con l'indipendenza dalla piattaforma, a meno che l'HTTP Server sia anch'esso scritto in Java, ciò lo renderebbe *cross-platform*, ma con un decadimento ancora una volta delleperformance.

Server HTTP proprietari

Un Server HTTP proprietario è un'applicazione Server che gestisce richieste HTTP efornisce ulteriori funzionalità che non sono comunemente disponibili tra gli altri Server HTTP. I Server Web proprietari combinano la capacità di soddisfare il classico meccanismo delle transazioni *request/response* con altre caratteristiche chiave come l'integrazione di applicazioni nate in ambiente legacy DataBase ed il mondo WWW.

Nonostante la comunità informatica reclami standard aperti e soluzioni non proprietarie, rimane il fatto che le soluzioni native o proprietarie spesso offrono le migliori prestazioni possibili per una specifica piattaforma, a parità di tecnologia ed esigenze delle applicazioni.

Il ventaglio dei Server Web proprietari non è molto ampio e comprende ad esempio *Lotus Domino*, *Oracle Web Listener*,e *Hyper-G*. La ragione principale che porta a scegliere soluzioni proprietarie è che nessun Server standard soddisfa in modo immediato bisogni di applicazioni specifiche. Apache, Netscape Enterprise Server e Microsoft Information Server non possono integrarsi in modo immediato con le applicazioni legacy Lotus Notes, motore delle intranet di molte aziende. Sviluppare un Server che traduce da Lotus Notes ad un generico Server HTTP e uno sforzo di grossa entità, cosìl'avere un vasto installato di applicazioni Notes giustifica l'adozione di un Server Web proprietario Lotus Domino.

La Servlet API di Java

Le *servlet* sono applicazioni Java che vengono eseguite come parti di una pagina HTML, ma a differenza delle *applet*, essi risiedono sul lato server. Per essere più precisi, la loro definizione generale è *componenti da lato server per l'estensione*

di server Java-enabled.

Per scrivere una generica servlet HTML è sufficiente ereditare dalla classe `HttpServlet` e definire due appositi metodi `doGet()` e `doPost()`), che vengono richiamati quando viene richiesto un servizio; esistono speciali metodi per accedere ai parametri che provengono da una form HTML, ed anche uno stream di output predefinito che ci consente di generare la pagina di risposta.

Le servlet vengono associati a URL speciali, proprio come avviene per i programmi CGI. A differenza dei programmi CGI, l'esecuzione di una servlet non richiede la creazione di un nuovo processo, dal momento che server WWW servlet-enabled contengono una Java virtual machine che, semplicemente, carica dinamicamente una nuova classe. In pratica, la servlet diventa parte integrante dello stesso server WWW. Questo è ancora più evidente se si considerano i server WWW interamente scritti in Java, come Jigsaw del W3C o il Java Server (ex Jeeves) di Sun. Per ottenere tempi di risposta ancora più rapidi, si possono configurare situazioni in cui il codice del servlet viene precaricato all'avvio.

Riassumendo: le servlet sono un potente meccanismo per scrivere applicazioni server di Internet o Intranet, che possono avvantaggiarsi dei classici punti di forza di Java, come portabilità, riuso effettivo del codice, robustezza e cosìvia. In particolare è da sottolineare la sicurezza dell'approccio: i servlet vengono controllati da un Security Manager analogo a quello degli applet, e questo dà agli sviluppatori la libertà di scrivere estensioni ai loro server senza aprire pericolosi "buchi" nella sicurezza della propria rete. Vale la pena di ricordare, infatti, che una parte degli attacchi alla sicurezza dei server WWW viene eseguita sfruttando bug presenti nei programmi CGI o, in generale, in estensioni del server.

Bibliografia

[1] Marzia Da Como. Progetto del sito Web della Facoltà di Ingegneria di Modena: parte prima. Tesi di laurea, Università di Modena e Reggio Emilia, Facoltà di Ingegneria, corso di laurea in Ingegneria Informatica, 1999-2000. http://sparc20.dsi.unimo.it/tesi/index.html.

[2] Database & Web. *Dev*, luglio/agosto, 1998. Ed. Infomedia.

[3] D. Beneventano S. Bergamaschi M. Vincini. *Progetto di Basi di Dati Relazionali - lezioni ed esercizi*. Pitagora Editrice, Bologna, 1999.

[4] Statuto dell'Università degli Studi di Modena e Reggio Emilia. http://www.unimo.it/ateneo/docs/statuto/.

[5] Oracle8 Database documentation.

[6] WorldWide Web Consortium. XSL Transformations (XSLT) - version 1.0. http://www.w3.org/TR/1999/REC-xslt-19991116, November 1999.

[7] James Clark. XT - an XSLT processor in Java.
available at: http://www.jclark.com/xt/xt.html.

[8] James Clark. XP - an XML parser in Java.
available at: http://www.jclark.com/xp/.

[9] Benoit Marchal. Servlet programming for teams. View Source web magazine, September 1999.
http://developer.iplanet.com/viewsource/marchal_xml.htm.

[10] S. Abiteboul P. Buneman D.Suciu. *Data on the Web*. Morgan Kaufmann Publishers, 1999.

[11] Madhu Siddalingaiah. Perspective on technology: Java Servlet API and XML, August 2000. http://java.sun.com.

[12] Jon Bosak. Four myths about XML. *IEEE Computers*, 31(10):120–122, October 1998.

[13] Jon Bosak. XML, Java, and the future of the Web. http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html, 1997.

[14] Benoit Marchal. More servlet programming for teams. View Source web magazine, December 1999. http://developer.iplanet.com/viewsource/marchal_xml2.htm.

[15] Ronald Bourret. XML and databases. http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases.htm, 1999.

[16] World Wide Web Consortium. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml, 1998.

[17] World Wide Web Consortium. Extensible HyperText Markup Language (XHTML) 1.0. http://www.w3.org/TR/xhtml1/, January 2000.

[18] 4xt.org - risorse per utenti di XT. http://www.4xt.org.