Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA
Corso di Laurea in Informatica

Basi di Dati

Elaborato G33
Sistema informativo per cartelle cliniche di una divisione ospedaliera

Candidati:
Enrico Giordano
Matricola VR359169

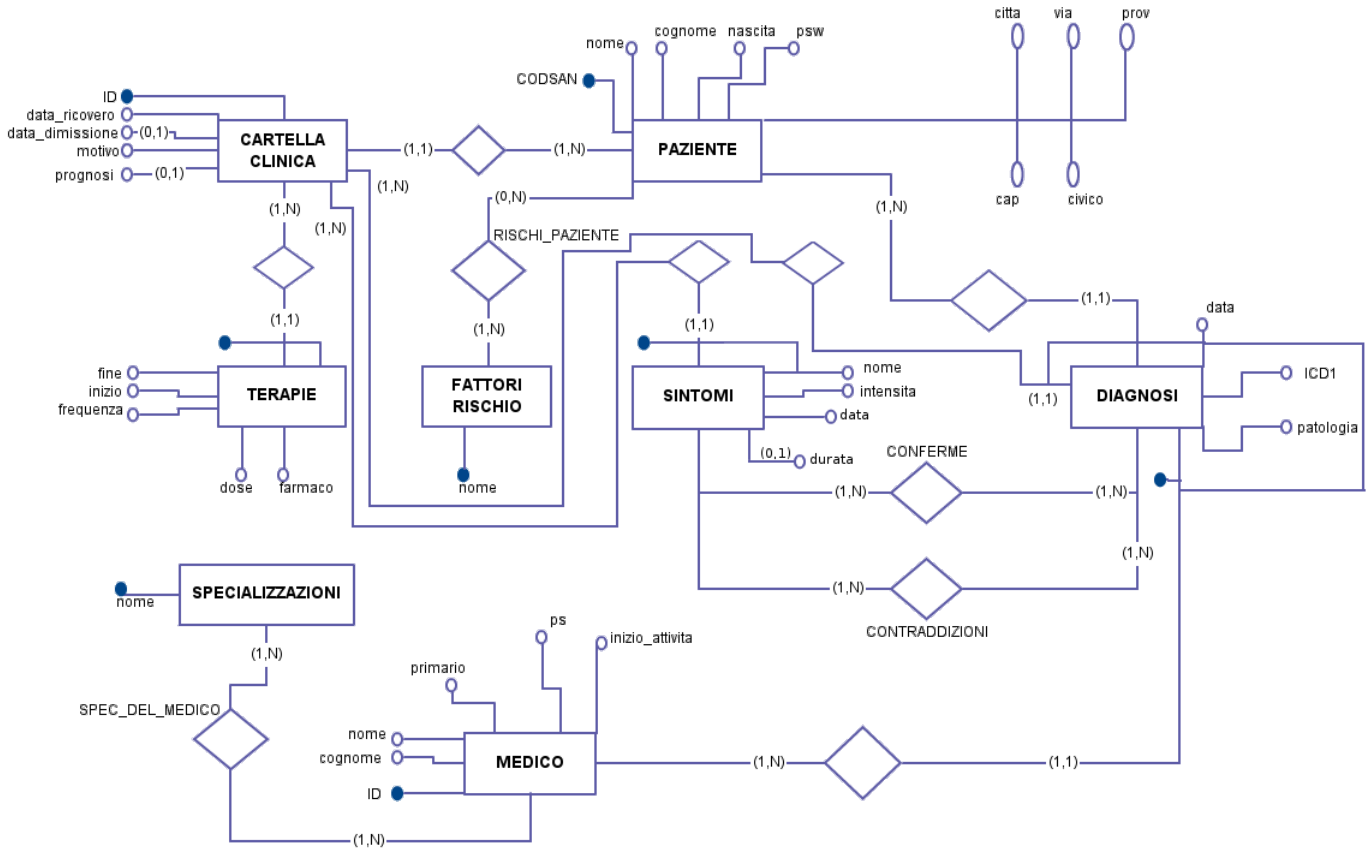
Cristian Pinna
Matricola VR361121

Anno Accademico 2013-2014

Indice

Parte I

Progettazione Concettuale

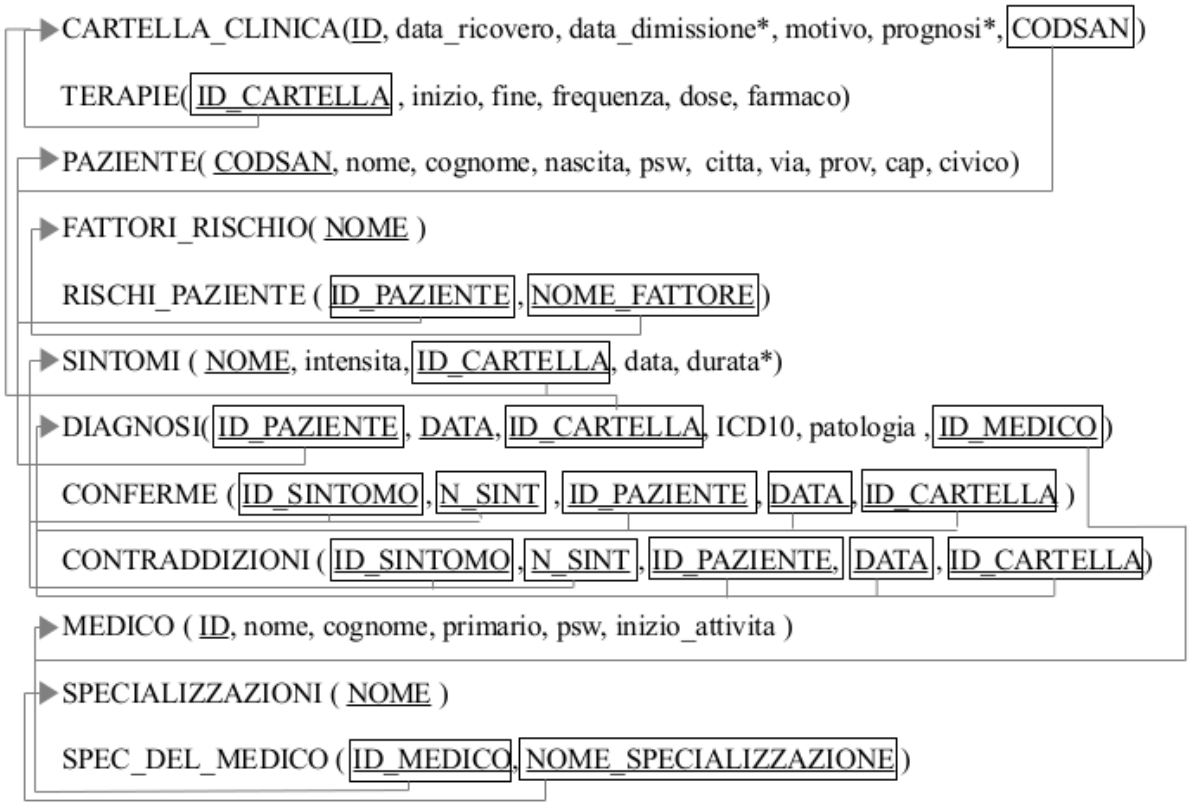


Elenco delle relazioni:

1. Relazione TERAPIE - CARTELLA CLINICA:
   - Cardinalità (1,1), una TERAPIA è associata univocamente ad una CARTELLA CLINICA
   - Cardinalità (1,N), ad una CARTELLA CLINICA può corrispondere più TERAPIE
2. Relazione CARTELLA CLINICA - PAZIENTE:
   - Cardinalità (1,1), una CARTELLA CLINICA è associata univocamente ad un PAZIENTE
   - Cardinalità (1,N), ad un PAZIENTE può corrispondere più CARTELLE CLINICHE
3. Relazione CARTELLA CLINICA - SINTOMI:
   - Cardinalità (1,N), ad una CARTELLA CLINICA può corrispondere uno o più SINTOMI
   - Cardinalità (1,1), un SINTOMO è associato univocamente ad una CARTELLA CLINICA
4. Relazione CARTELLA CLINICA - DIAGNOSI:
   - Cardinalità (1,N), ad una CARTELLA CLINICA può corrispondere una o più DIAGNOSI
   - Cardinalità (1,1), una DIAGNOSI è associata univocamente ad una CARTELLA CLINICA
5. Relazione PAZIENTE - FATTORI RISCHIO:
   - Cardinalità (0,N), ad un PAZIENTE può corrispondere nessuno o più FATTORI RISCHIO
   - Cardinalità (1,N), ad un FATTORE RISCHIO può corrispondere uno o più PAZIENTI
6. Relazione PAZIENTE - DIAGNOSI:
   - Cardinalità (1,N), ad un PAZIENTE può corrispondere una o più DIAGNOSI
   - Cardinalità (1,1), una DIAGNOSI è associata univocamente ad un PAZIENTE
7. Relazione SINTOMI - DIAGNOSI:
   - Cardinalità (1,N), ad un SINTOMO può corrispondere una o più DIAGNOSI
   - Cardinalità (1,N), ad una DIAGNOSI può corrispondere uno o più SINTOMI
8. Relazione DIAGNOSI - MEDICO:
   - Cardinalità (1,1), una DIAGNOSI è associata univocamente ad un MEDICO
   - Cardinalità (1,N), ad un MEDICO può corrispondere una o più DIAGNOSI
9. Relazione MEDICO - SPECIALIZZAZIONI:
   - Cardinalità (1,N), ad un MEDICO può corrispondere una o più SPECIALIZZAZIONI
   - Cardinalità (1,N), ad una SPECIALIZZAZIONE può corrispondere uno o più MEDICI

Per identificare il primario, è stato utilizzato l'attributo "primario" dell'entità "medico" come stringa contenente "si" o "no" a seconda che fosse primario o no.

Parte II
Schema Logico



Questo schema logico rappresenta una visione globale sull'elenco dei vari attributi, sulle chiavi primarie (attributi sottolineati) e sulle relazione tra di essi (i riquadri attorno al nome dell'attributo e la relativa freccia che punta alla relazione).

È stato tradotto nel file "database.sql" e popolato tramite il file "popola.sql". Quest'ultimo file esegue degli script che contengono molti insert per ogni tabella. Per generare questi script, sono stati creati dei programmi in grado di generare file .sql con un numero considerevole di insert in base al design del database, in modo da poter testare su grandi numeri il sito (e il comportamento del database).

Parte III

Page Schema

page-schema Homepage unique (

informazioni: link (info, *InfoPage.jsp);
personale_medico: link (personale, *PersonalePage.jsp);
patologie: link (patologie, *PatologiePage.jsp);
login: link (login, *Login.html);

);

page-schema InfoPage unique (

primario: String;
informazioni: text;
foto: list_of(data[]);

);

DB to page-schema InfoPage (

primario:
```
select *
from medico as m
where m.primario = 'si';
```

);

page-schema LoginPage unique (

login_paziente: form(
login: text;
pw: password;
invia: submit();
);

login_medico: form(
login: text;
pw: password;
invia: submit();
);

);

DB to page-schema LoginPage (

login_cliente:

```
if(     select *
        from paziente as p
        where p.codsan = ?codsan?
        and p.psw = ?psw?
)
then *PazientePage else *LoginPage
end;
```

login_medico:

```
if(     select m.*
        from medico as m
        where m.id = ?id?
        and m.psw = ?psw?
)
then *DiagnosiPage else *LoginPage
end;
```

);

page-schema PazientePage (

```
dati: (
        codice_sanitario: string;
        nome: string;
        cognome: string;
        data_nascita: string;
        via: string;
        civico: string;
        );
cartelle_cliniche: list of(
        link(cartella clinica, *CartellaPage);
        );

fattori_rischio: string;
elenco_medici: list of(
        nome: string;
        cognome: string;
);
```

);

DB to page-schema PazientePage (

dati:
```
select p.*
from paziente as p
where p.codsan = ?codsan?
```

cartelle_cliniche:
```
select c.*
from cartella_clinica as c, paziente as p
where c.codsan = ?codsan?
and c.codsan = ?codsan?
```

);

fattori_rischio:
```
select r.*
from paziente as p, rischi_paziente as r
where p.codsan = (:codsan)
and p.codsan = r.id_paziente
```

elenco_medici:
```
select distinct m.
from paziente as p, medico as m, diagnosi as d
where p.codsan = ?codsan)?
and p.codsan = d.id_paziente
and m.id = d.id_medico
```

);

page-schema CartellaPage (

dati: (
- ID: string;
- dataRic: string;
- motivo: string;
- prognosi: string;

);

diagnosi: list of (
- medico: (
  - nome_medico: string;
  - cognome_medico: string;
-);
- data: date;
- patologia: string;
- icd10: string;
- conferme: list of (
  - sintomi: string;
  -);
- contraddizioni: list of (
  - sintomi: string;
  -);

);

terapie: list of (
farmaco: string;
dose: float;
posologia: string;
inizio cura: date;
fine cura: date;

);
);

DB to page-schema CartellaPage (

dati:

```
select c.*
from cartella_clinica as c, paziente as p
where c.id = ?id?
```

terapie:

```
select t.*
from cartella_clinica as c, terapie as t
where c.id = ?id?
and c.id = t.id_cartella
```

diagnosi:

```
select d.*, m.*
from cartella_clinica as c, diagnosi as d,
paziente as p, medico as m
where c.id = ?id?
and p.codsan = d.id_paziente
and p.codsan = c.codsan
and d.id_medico = m.id
```

terapie:

```
select t.*
from cartella_clinica as c, terapie as t
where c.id = ?id?
and c.id = t.id_cartella
```

);

page-schema PatologiePage (

patologie: text;
);

DB to page-schema PatologiePage (

patologie:

```
select d.*
from diagnosi as d
```

);

page-schema PersonalePage (

dati: text;
numero pazienti: int;
);

DB to page-schema PersonalePage (

dati:

```
select d.*
from diagnosi as d
```

numero pazienti:

```
select count(*)
from diagnosi
where patologia = (:patologia)
and icd10 = (:icd10)
group by icd10, patologia
```

);

page-schema DiagnosiPage (

new_diagnosi: form (
paziente: string;
cartella clinica: string;
data: date;
ICD10: string;
sintomi: list of (string);
tipologia: list of (string);
);
);

DB to page-schema DiagnosiPage (

new_diagnosi: *INSERT INTO DIAGNOSI ;*
new_sintomi: *INSERT INTO SINTOMI;*
new_conferme: *INSERT INTO CONFERME ;*
new_contraddizioni: *INSERT INTO CONTRADDIZIONI;*
);

Parte IV
Struttura dell'applicazione web

1 Moduli

L'applicativo è distribuito secondo lo standard mvc2:

- Model: comprende la classe DBMS.java e tutte le classi che descrivono i vari DataBean. DBMS.java si occupa di interrogare il DB e di manipolare i dati al suo interno. La comunicazione con Main.java avviene tramite DataBean in ambo le direzioni. Questa parte è distribuita nelle directory "bean" e "dbms";
- Control: comprende la sola classe Main.java, che non è altro che la servlet centrale che si occupa di ricevere tutte le richieste GET e POST, di richiedere a DBMS.java tutte le informazioni di cui si ha bisogno, e di inoltrare quest'ultime insieme alla richiesta HTTP alla pagina JSP appropriata.
- View: comprende tutte le pagine JSP, le librerie javascript usate, le immagini e le foto. Si trova nella cartella "WebContent".

Per java ogni interrogazione o manipolazione di dati SQL sono singole transizioni. Se si desidera effettuare una transizione composta da più query è necessario disabilitare l'autocommit ed impostare il livello di isolamento della transizione. Inoltre bisogna gestire esplicitamente il commit e il rollback. Se la transizione accede ad una risorsa non disponibile, allora java lanciera un eccezione ed è compito del programmatore far ricominciare dall'inizio l'intera transizione. Il livello di isolamento più rigido è SERIALIZABLE.

Il context tomcat del sito WEB è "medicina_hibernate", mentre il path relativo da context per raggiungere la servlet Main è "home". La porta per del server è 8080. Quindi l'URL per una richiesta HTTP al server sarà del tipo:

$http://localhost:8080/medicina\_hibernate/home$

Parte V
Gestione richieste

La servlet centrale identifica e gestisce le richieste HTTP sulla base che esse siano GET o POST e dipendentemente dal valore del parametro ps passato nella richiesta. Per le richieste GET, ps può assumere i seguenti valori:

- null, si rimanda alla HomePage;
- info, si rimanda alla pagina di informazioni;
- login, si rimanda alla pagina di login;
- cartella, si rimanda alla pagina CartellaPage;
- patologie, si rimanda alla pagina PatologiePage;
- personale, si rimanda alla pagina del personale medico (PersonalePage).

Per le richieste di tipo POST (invio valori da un form), ps può assumere i seguenti valori:

- paziente, si esegue il login per il paziente;
- medico, si esegue il login per il medico;

- diagnosi, si esegue l'insert per la diagnosi.

Parte VI
Pagine Web

Il sito è stato organizzato in maniera servlet-centric, in cui esiste un'unica servlet che gestisce l'apertura delle pagine web in base al parametro "ps" presente nella barra url. Per rendere più gradevole l'aspetto del sito, è stato creato un css unico per tutte le pagine del sito.

Per gestire il controllo errori, è stata creata una pagina chiamata "error.jsp", che viene richiamata in automatico ogni qual volta ci sia un'eccezione nel codice.

A differenza delle specifiche, in cui si diceva di inserire nella HomePage le informazioni della divisione ospedaliera (con le foto) e il login, è stato deciso di separare queste due sezioni in rispettive pagine web differenti (Info e Login), in modo da rendere tutto più ordinato, mentre la pagina HomePage funge solo da pagina di menù.

Non verranno riportate le query che hanno reso possibile l'acquisizione dei dati da parte del database in quanto sono presenti nel page-schema precedentemente presentato.

Infine, tramite una qualsiasi pagina si può raggiungere qualunque altra pagina (che non richieda il login) tramite la barra superiore di selezione (sotto il titolo, di colore viola).

2 Homepage HTML

Questa è la pagina principale tramite cui l'utente è in grado di selezionale ogni pagina che vuole consultare. È stata scritta in html in quanto è una pagina statica e non ha necessità di essere modificata tramite i dati del database.

3 Info JSP

Questa pagina presenta delle informazioni statiche e delle immagini linkate presenti nella directory *css/images*. L'unica informazione che ha reso necessario l'utilizzo di un'interrogazione al database è stata quella di ottenere il nome del primario. Per questo fatto, è stato necessario sviluppare una pagina di tipo JSP.

Passaggio di parametri: nessuno.

4 Login JSP

La pagina di Login è statica in quanto presenta sempre gli stessi form in grado di inviare alla servlet i dati per eseguire il login, però deve essere in grado di stampare un messaggio di errore quando il login non è andato a buon fine (quindi deve essere modificabile). Questi dati vengono inviati con metodo post alla servlet, in modo che non vengano visti sulla barra url; la servlet esegue una query per verificare che nome utente e password corrispondano ad almeno un'istanza nel database. Se nome utente e password corrispondono, si viene reindirizzati alla rispettiva pagina (se si esegue il login come paziente si viene reindirizzati a PazientePage, altrimenti a DiagnosiPage), altrimenti viene stampato un messaggio di errore e si viene reindirizzati nuovamente alla pagina di login.

Sono presenti quindi due form: uno per il paziente, in cui si inserisce il codice sanitario e la password, e uno per il medico, in cui si inserisce il suo identificatore (codice sanitario) e la password.

Passaggio di parametri:

```
if (request.getAttribute("error") != null)
        error = ((Integer)request.getAttribute("error"));
```

5 PazientePage JSP

La pagina personale del paziente contiene tutti i dati del paziente memorizzati nel database. La parte sinistra della pagina contiene i dati personali, mentre la parte destra contiene i dati relativi ai medici che lo hanno diagnosticato e alle sue cartelle cliniche. Le cartelle cliniche presentate rimandano alla pagina corrispondente che presenta tutti i dati associati alla cartella clinica. Per ottenere le cartelle cliniche del paziente è stata utilizzata una query (vedere page-schema) ordinando il result set per id.

Passaggio di parametri:

```
if (request.getParameter("user") != null)
        paziente = (String)request.getParameter("user");
```

6 CartellaPage JSP

Questa pagina contiene tutti i dati importanti di una cartella clinica che possono interessare al paziente. Vengono presentati in ordine: la data ricovero, la data dimissione, il motivo del ricovero, la prognosi (che può non essere presente), l'elenco di terapie somministrate durante il ricovero (farmaco prescritto, dose, posologia e inizio e fine cura) e l'elenco di tutte le diagnosi effettuate dai medici, riportando il nome del medico che l'ha effettuata, la data, la patologia diagnosticata e i sintomi che confermano o contraddicono la patologia.

Anche se non era stato specificato, nelle diagnosi sono stati riportati i sintomi che confermano e contraddicono la diagnosi.

Passaggio di parametri:

```
if (request.getParameter("cartella") != null)
        cartella = (String) request.getParameter("cartella");
```

7 PersonalePage JSP

Questa pagina contiene tutti i dati del personale medico della divisione ospedaliera.

In primo piano viene presentato il primario e le sue specializzazioni (questo è stato ottenuto da una query); successivamente, in una sezione della pagina in verde (camice del chirurgo) sono presentati i medici, con la data di inizio attività, il numero di diagnosi effettuate e l'elenco delle loro specializzazioni. Ogni dato è presentato su riquadro azzurro (camice del medico di reparto). Questi dati sono ottenuti iterando sul result set ottenuto dalla query descritta nel page-schema.

Passaggio di parametri: nessuno.

8 PatologiePage JSP

In questa pagina vengono presentate tutte le patologie diagnosticate in reparto, riportando il codice ICD10 e il numero di pazienti a cui è stato diagnosticata la patologia considerata. Per ottenere questi dati, è stata eseguita la query descritta nel page-schema; ottenuto il result set, è stato utilizzato un ciclo for per presentare ogni istanza del database proveniente dal risultato della query.

Passaggio di parametri: nessuno.

9 DiagnosiPage JSP Javascript Ajax

Questa è la pagina che contiene più codice di programmazione delle altre, poiché sono state aggiunte delle features che permettono al medico di non commettere errori nell'inserimento di una diagnosi e di rendere facile l'utilizzo di questa pagina.

Per inserire una diagnosi con i relativi sintomi, è necessario fare almeno 3 insert: uno per la diagnosi, uno per i sintomi diagnosticati e uno o più per indicare se il sintomo conferma o contraddice la diagnosi.

Riguardo alla diagnosi, è necessario sapere il codice sanitario del paziente e della cartella clinica associata. Per rendere più facile la manipolazione di questi dati, è stato utilizzato un campo del form generale di tipo select tramite cui si può selezionare il paziente; appena si seleziona il paziente, si attiva un sottoprogramma basato su tecnologia Ajax che esegue una query per sapere l'identificatore di tutte le cartelle cliniche associate al paziente e inserire altrettanti option del successivo campo select della form. Questo rende possibile la selezione della cartella clinica senza scriverla a mano (e quindi commettere probabilmente errori).

Oltre a questo, è possibile inserire dinamicamente più sintomi per la diagnosi e anche eliminarli qualora ci fossero stati errori di inserimento. Per sviluppare questa utile feature, sono stati utilizzati varie funzioni Javascript che rendessero trasparente la memorizzazione dei dati, l'aggiunzione e l'eliminazione di questi senza creare problemi all'utilizzatore.

Parametri passati:

```
if (request.getAttribute("error") != null)
    error = Integer.parseInt((String)request.getAttribute("error"));

if (request.getParameter("user") != null)
        medico = (String)request.getParameter("user");

else if (request.getParameter("medico") != null)
        medico = (String)request.getParameter("medico");
```

Parte VII
Strategie progettuali e considerazioni personali

Per rendere più realistico il database, sono stati aggiunti dei vincoli sugli attributi:

- la cartella clinica deve avere una data di ricovero maggiore o uguale della data di nascita del paziente;
- la cartella clinica deve avere una data di ricovero minore della data di dimissioni;

- le terapie devono essere fatte in un arco di tempo compreso tra la data di ricovero e dimissione della cartella clinica corrispondente;
- le terapie devono avere una data di inizio minore o uguale della data di fine;
- le diagnosi devono essere fatte in un arco di tempo compreso tra la data di ricovero e dimissione della cartella clinica corrispondente.

Considerazioni personali e strategie adottate durante lo sviluppo del progetto:

- È stato resa la data dimissione (attributo della cartella clinica) come attributo opzionale in quanto si è pensato che possono esserci delle cartelle cliniche di pazienti ancora ricoverati in reparto (nonostante le specifiche non esplicitassero questo fatto);
- Realizzazione del DB in modo tale da poter ottenere più relazioni possibili con la cartella clinica;
- Utilizzo del metodo Hibernate durante la realizzazione del progetto in modo tale da poter semplificare le query, tenendo presente che esse restituivano tanti valori nidificati a cui ci si poteva raggiungere tramite superchiavi;
- Durante la creazione della pagina relativa alle diagnosi (DiagnosiPage) il campo delle cartelle cliniche viene popolato tramite uno script ajax-json-jquery a seconda del paziente selezionato, in modo tale da evitare l'inserimento manuale di una cartella clinica potenzialmente errata;
- Per la realizzazione generale della pagina web che gestisce l'intero progetto ci siamo sentiti di renderla più gradevole graficalmente inserendo uno stile di impaginazione html in formato css;
- È stato utilizzato come ambiente di sviluppo Eclipse, in quanto garantiva il controllo degli errori una buona velocità di programmazione.

Parte VIII
Tecnologie aggiuntive utilizzate

10 Hibernate

Hibernate è un sistema o piattaforma middleware che offre un'interfaccia tra programmatore e database in modo da semplificare e gestire al meglio in maniera trasparente il database da interrogare o aggiornare. Questo sistema, tramite classi Java che rappresentano le entità del database, permette di avere un mapping tra variabili delle classi e attributi delle entità; in questo modo si può avere accesso agli attributi del database lavorando con i metodi get e set sulle variabili interessate.

Il mapping è reso possibile tramite file xml che chiarificano al sistema come mappare sia gli attributi delle entità sia le relazioni con la rispettiva cardinalità.

La progettazione si è incentrata sulla corretta scrittura dei file xml e delle relative classi; si è cercato inoltre di esplicitare quali fossero gli identificatori raggruppandoli in sottoclassi(qualora ce ne fossero più di uno), in modo da rendere ancora più chiara la programmazione: ogni classe avrà una dichiarazione sia di attributi sia di identificatori (quindi la creazione di una sottoclasse).

Oltre agli attributi e agli identificatori, si è deciso di associare la classe referenziata tramite attributo esterno alla classe interessata; in questo modo si può avere accesso a tutti i campi della classe referenziata applicando i metodi get e set alla sottoclasse, presente quindi nella classe interessata come variabile instanziata.

Come esempio pratico, la classe CartellaClinica ha i seguenti attributi:

```
private String id;
private Paziente paziente;
private Date dataRicovero;
private Date dataDimissione;
private String motivo;
private String prognosi;
private Set terapies = new HashSet(0);
private Set diagnosis = new HashSet(0);
private Set sintomis = new HashSet(0);
```

Come si può notare, la referenza all'entità Paziente (tramite attributo "codsan" nel modello relazionale) è resa tramite la dichiarazione della variabile paziente; quindi tramite una singola interrogazione di una specifica cartella clinica si possono sapere anche i campi del rispettivo paziente associato ad essa.

Come metodo di verifica della correttezza della progettazione per il sistema Hibernate, è stato utilizzato un plugin di Eclipse chiamato JBoss Tools, che contiene diverse features per Hibernate e permette di verificare la correttezza sia dei file xml sia delle classi Java a partire dalla descrizione del database.

Bisogna notare inoltre che le relazioni molti-a-molti hanno prodotto una referenza tradotta in Java con un HashSet, in modo da poter aver accesso ad ogni valore semplicemente di questo tipo di entità applicando i metodi set e get a questi HashSet. Questo si è rilevato di notevole importanza in quanto, per sapere la quantità di elementi di un certo tipo associati ad una entità, è bastato vedere la grandezza di questi HashSet utilizzando il metodo size, senza quindi fare un'ulteriore query. È stato necessario però forzare il sistema, in alcuni casi, ad avere un comportamento non "lazy", quindi impostando a false tale attributo nel file xml, in modo da caricare in memoria anche le istanze referenziate tramite le relazioni e quindi avere libero accesso ad esse.

11 Ajax JSON e JQuery

Poiché si è voluta rendere facile la scelta dei pazienti e delle relative cartelle cliniche nella DiagnosiPage (per rendere sicuro e affidabile un possibile insert da parte del medico nel database), è stato deciso di utilizzare un'insieme di tecnologie che potessero rendere dinamica la scelta da parte del medico. Sono state utilizzate tre tecnogie di sviluppo web chiamate Ajax, JSON e JQuery.

Ajax è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. Consiste nel creare una servlet Java che riceve parametri da parte di una pagina web (nel nostro caso una JSP); questa servlet elabora i dati e li invia alla pagina chiamante, in modo che possa ricevere dinamicamente dati e quindi modificare il suo aspetto o comportamento.

Si è voluta utilizzare questa tecnologia per rendere dinamica la selezione in un campo del form di tipo select; poiché però questi campi devono essere "popolati" con dati provenienti dal database, è stato necessario introdurre la tecnologia JQuery e JSON.

JQuery consiste in un insieme di librerie per semplificare la programmazione web, nel nostro caso è stato utilizzato internamente ad Ajax per eseguire una query sul database e mappare in un LinkedHashMap la risposta. Per inviare poi questa risposta alla pagina web, è stato utilizzato JSON, una teconogia che consente lo scambio di dati tra architetture client-server; in questo modo si passa il risultato del calcolo della servlet alla pagina in maniera trasparente senza usare metodi doGet e doPost.

Tutto questo è visibile nella servlet ActionServlet e nell'intestazione della pagina DiagnosiPage.