IL DATABASE DI

ALGEBRANDO

Un progetto di:

Davide Valeriani

Matricola 190883

davide.valeriani@studenti.unipr.it

Corso di laurea in Ingegneria Informatica

Esame di Basi di Dati A

Prof. Stefano Cagnoni

Anno accademico 2008/2009

Sessione di esami: 08/06/2009 – 30/06/2009

SPECIFICHE DEL PROGETTO

Progettare e realizzare una base di dati che permetta di gestire i risultati di un'esercitazione sull'algebra dei numeri relativi svolta da alcuni alunni della scuola primaria attraverso il software grafico Algebrando.
Ciascun alunno può svolgere da 0 a 20 esercizi diversi e ad ogni esercizio è assegnato un punteggio pari a 5, 10 o 15 punti a seconda della difficoltà. Ogni volta che un alunno sbaglia, gli si dà la possibilità di ridare la risposta, decurtando però un punto dal totale. Una volta memorizzato il risultato di un alunno su un esercizio, nel caso in cui lo studente svolgesse nuovamente l'esercizio, il risultato verrebbe ignorato, tenendo per buono il precedente.
Il database deve prevedere la possibilità per gli insegnanti di visualizzare e cancellare i risultati di ogni alunno delle classi in cui hanno la cattedra, al fine di rivalutare gli studenti.
Inoltre, ogni insegnante può modificare composizione della classe aggiungendo o rimuovendo studenti (per far sì che uno studente possa o meno svolgere gli esercizi senza essere valutato). Alcuni insegnanti, inoltre, possono avere diritti amministrativi sul database che gli permettono di gestirlo in modo totale.
Oltre agli studenti, di cui sono memorizzati nome e cognome, devono essere gestite anche le classi con la composizione di studenti per ogni sezione e anno scolastico.
Si assuma che un docente possa insegnare in più classi e che non sia ancora in vigore il maestro unico!

Prima fase: analisi delle specifiche

Dal testo del problema, il primo concetto che appare è quello di **studente**, che andrà sicuramente previsto. Infatti, il testo dice esplicitamente che di ogni studente devono essere memorizzati nome, cognome, classe di appartenenza per ogni anno scolastico e risultati nei vari esercizi (da 0 a 20).
Un altro concetto importante è quello di **esercizio**, in quanto il testo prevede che ad ogni esercizio sia assegnato un punteggio in base alla difficoltà e si può pensare di voler memorizzare anche la soluzione dell'esercizio nel database, essendo un numero intero.
Dal testo si individua anche il concetto di **insegnante**, il quale può anche avere diritti di amministrazione della base di dati. Oltre ai dati anagrafici, quindi, sarà necessario memorizzare i diritti di accesso di cui ogni insegnante dispone (solo docente oppure docente amministratore). Supponiamo che la scuola sia piccola e che, pertanto, l'amministratore della base di dati sia anche un insegnante della scuola stessa.
Infine, il testo precisa un quarto concetto, quello di **classe**, a cui sarà associata una sezione e un anno scolastico in cui questa classe è attiva. Infatti, la classe 1^A sarà attiva nell'a.s. 2007/2008 ma anche nel 2008/2009 eccetera. Pertanto, occorre gestire più occorrenze della stessa classe per ogni anno. Per semplicità, supponiamo che in ogni anno scolastico vengano ricreate le classi ed aggiunti gli alunni che le compongono, anche perché un docente potrebbe insegnare in classi diverse in base agli anni e poi si suppone che gli esercizi in questione vengano proposti solo agli alunni della classe 5^.

Seconda fase: modello concettuale Entità/Relazioni

Le entità necessarie alla memorizzazione di tutti i dati sono pertanto 4: una che memorizzi le informazioni relative ai docenti (docente), una per memorizzare le informazioni relative agli studenti (studente), una per le informazioni sugli esercizi (esercizio) e una per le informazioni relative alle classi (classe).

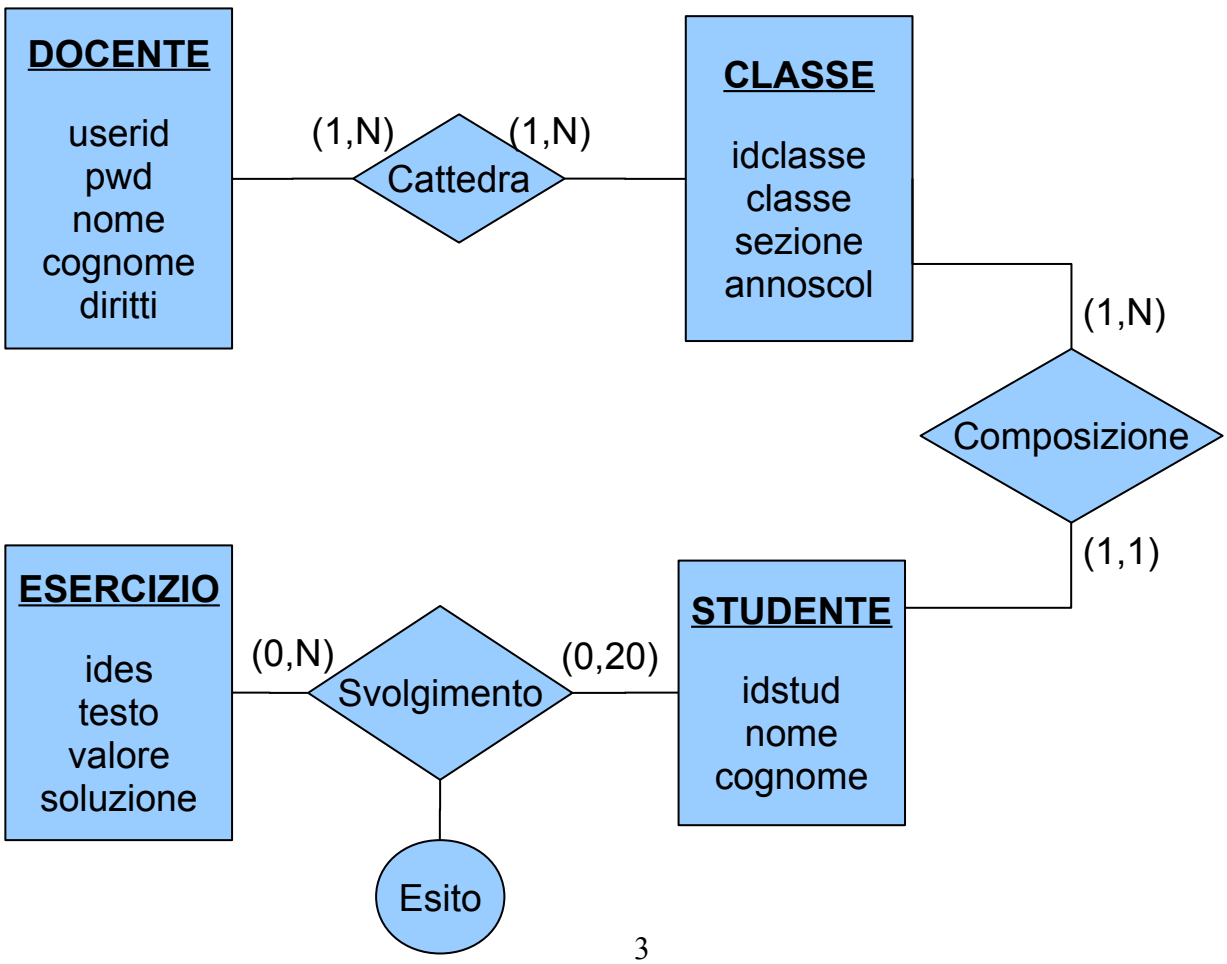
Le associazioni presenti tra queste entità sono 3:

- un'associazione **molti a molti** tra le entità docente e classe, in quanto un docente può insegnare a più classi e, allo stesso tempo, una classe può avere più docenti ad essa associati (non vige ancora il maestro unico);
- un'associazione **molti a molti** tra le entità studente ed esercizio, in quanto uno studente può svolgere più esercizi e, allo stesso tempo, un esercizio può essere svolto da più studenti;
- un'associazione **uno a molti** tra le entità classe e studente, dato che una classe può essere formata da più studenti, ma uno studente può, nello stesso anno scolastico, far parte di una sola classe;

In seguito, sono stati identificati gli attributi di ogni entità. In particolare:

- l'entità **docenti** è caratterizzata dall'userid e la password mediante i quali il docente si connetterà al database, dal nome e cognome del docente e dai diritti che ha sul database (docente normale o amministratore del database);
- l'entità **classi** è caratterizzata, oltre che da un codice identificativo della classe, dall'anno della classe (ad esempio 1°, 2°, ecc.), dalla sezione (ad esempio A, B, ecc.) e dall'anno scolastico a cui essa si riferisce;
- l'entità **studenti** è caratterizzata dalla matricola dello studente, dal nome e dal cognome dello studente stesso;
- l'entità **esercizi** è caratterizzata, oltre che da un identificativo dell'esercizio, dal testo dell'esercizio, dal punteggio massimo ottenibile dallo studente nello svolgere l'esercizio e dalla soluzione dell'esercizio.

Inoltre, nella relazione molti a molti tra studenti ed esercizi, sarà memorizzato il punteggio che lo studente ha ottenuto nello svolgere un esercizio.
Da queste informazioni è stato possibile ricavare il modello entità-relazioni (o modello concettuale).

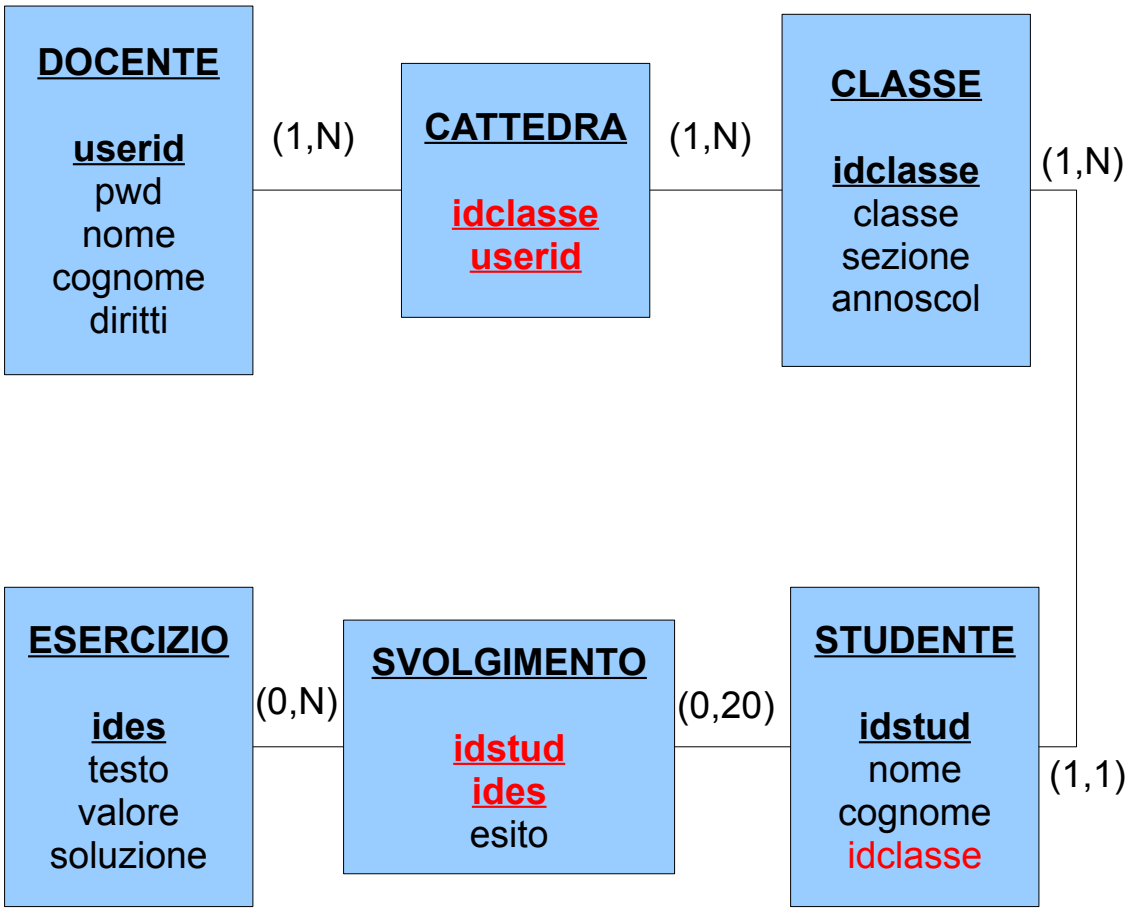


Terza fase: modello logico

Partendo dal modello entità/relazioni, per ottenere il modello logico occorre:

- trasformare le entità in tabelle;
- trasformare gli attributi delle entità in campi delle tabelle;
- identificare una chiave primaria per ogni tabella: userid per la tabella docente (ogni docente deve avere uno userid diverso per poter accedere al database), idclasse per la tabella classe, idstudente per la tabella studente e idesercizio per la tabella esercize;
- implementare la relazione molti a molti tra le tabelle docente e classe aggiungendo la tabella **Cattedra** che ha, come chiave primaria, le chiavi primarie delle tabelle classi e docenti;
- implementare la relazione molti a molti tra le tabelle esercizio e studente aggiungendo la tabella **Svolgimento** che ha, come chiave primaria, le chiavi primarie delle tabelle esercizi e studenti e, inoltre, il campo Esito che conterrà il punteggio ottenuto dallo studente nell'esercizio;
- implementare la relazione uno a molti (Composizione) tra le tabelle classe e studente aggiungendo, nella tabella studente, la chiave primaria della tabella classi come chiave esterna.

Dopo aver applicato le regole di derivazione, è stato possibile ricavare il modello logico (**attributo** = Chiave primaria; attributo = Chiave esterna).



Con il modello logico è così terminata la progettazione del database.

Le tabelle così create sono pertanto 6:
DOCENTE (**userid**, pwd, nome, cognome, diritti)
CLASSE (**idclasse**, classe, sezione, annoscol)
CATTEDRA (**idclasse**, **userid**)
STUDENTE (**idstud**, nome, cognome, idclasse)
ESERCIZIO (**ides**, testo, valore, soluzione)
SVOLGIMENTO (**idstud**, **ides**, esito)

Creazione delle tabelle

Il database viene creato su un server mysql. Per prima cosa, si creano le tabelle con l'istruzione SQL Create Table. Di seguito le istruzioni per creare le 6 tabelle.

Creazione della tabella "docente"

```
CREATE TABLE docente
(userid varchar(15) NOT NULL,
 pwd varchar(15) NOT NULL,
 nome varchar(15) NOT NULL,
 cognome varchar(20) NOT NULL,
 diritti varchar(15) NOT NULL default 'docente',
 PRIMARY KEY (userid));
```

Con questa istruzione viene creata una tabella chiamata "docenti" con cinque campi, di cui nessuno può contenere un valore nullo (questo il significato dell'istruzione NOT NULL). Inoltre, al campo "diritti" viene assegnato il valore "docente", ovvero, se non viene specificato, il docente inserito avrà diritti come semplice docente, non come amministratore. Infine, con l'ultima istruzione, si specifica quale degli attributi è la chiave primaria della tabella, ovvero userid.

Creazione della tabella "classe"

```
CREATE TABLE classe
(idclasse smallint(6) NOT NULL auto_increment,
 classe varchar(1) NOT NULL,
 sezione varchar(1) NOT NULL,
 annoscol varchar(9) NOT NULL,
 PRIMARY KEY (idclasse));
```

In questa tabella, la chiave primaria (idclasse) è di tipo contatore, che in MySQL si implementa aggiungendo l'istruzione auto_increment al tipo smallint.

Creazione della tabella "cattedra"

```
CREATE TABLE cattedra
(idclasse smallint(6) NOT NULL,
 userid varchar(15) NOT NULL,
 PRIMARY KEY (idclasse, userid),
 FOREIGN KEY (idclasse) REFERENCES classe(idclasse),
 FOREIGN KEY (userid) REFERENCES docente(userid));
```

La tabella cattedra è una tabella che ha la funzione di risolvere la relazione molti a molti che intercorre tra le tabelle classi e docenti. In questa tabella, la chiave primaria è la coppia delle chiavi esterne che si riferiscono alle due tabelle che essa collega. Perciò, dopo aver dichiarato la chiave primaria composta da una coppia di attributi, si definiscono le chiavi esterne (idclasse e userid), specificando a quale campo di quale tabella si riferiscono.

Creazione della tabella "studente"

```
CREATE TABLE studente
(idstud smallint(5) NOT NULL auto_increment,
 nome varchar(30) NOT NULL,
 cognome varchar(30) NOT NULL,
 idclasse smallint(6) NOT NULL,
 PRIMARY KEY (idstud),
 FOREIGN KEY (idclasse) REFERENCES classe(idclasse));
```

Creazione della tabella "esercizio"

```
CREATE TABLE esercizio
(ides smallint(5) NOT NULL,
 testo varchar(30) NOT NULL,
 valore smallint(2) NOT NULL,
 soluzione smallint NOT NULL,
 PRIMARY KEY (ides));
```

Nella tabella esercizi ho deciso di non assegnare alla chiave primaria il tipo contatore perché il codice dell'esercizio serve anche nel programma Algebrando, perciò lo sviluppatore deve avere la possibilità di assegnare un codice all'esercizio manualmente. Per semplicità, ho scelto di associare ad ogni esercizio un numero progressivo da 1 a 20, a seconda della posizione dell'esercizio nel programma.

Creazione della tabella "svolgimento"

```
CREATE TABLE svolgimento
(idstud smallint(5) NOT NULL,
 ides smallint(5) NOT NULL,
 esito smallint(3) default NULL,
 PRIMARY KEY (idstud, ides),
 FOREIGN KEY (idstud) REFERENCES studente(idstud),
 FOREIGN KEY (ides) REFERENCES esercizio(ides));
```

Come la tabella "cattedra", anche la tabella "svolgimento" è una tabella che permette di risolvere la relazione molti a molti tra le tabelle "studente" e "esercizio" passando dal modello concettuale a quello logico. Perciò, contiene come chiave primaria la coppia di chiavi primarie delle due tabelle, che sono anche chiavi esterne della stessa tabella "svolgimento". Inoltre, la tabella è composta anche dal campo "esito", che, dato che verrà riempito dal programma solo quando lo studente svolgerà quel dato esercizio, assume il valore NULL (cioè rimane vuoto) se si dovesse creare la tupla manualmente. Questo accorgimento è necessario in quanto il campo esito diverso da NULL significa che lo studente ha già svolto l'esercizio e, perciò, impedisce allo studente di memorizzare nuovamente il proprio risultato.

È obbligatorio creare le tabelle in quest'ordine, in quanto sono presenti chiavi esterne. In pratica, se venisse creata una tabella con una chiave esterna che si riferisce ad una tabella non ancora creata, il DBMS (DataBase Management System) darebbe errore.

Interrogazioni (Query)

Per soddisfare tutte le richieste del problema, sono necessarie le seguenti query:

INSERIMENTO DI UN RISULTATO

Controllo che lo studente nome cognome esista nel database e ne recupero l'id

select idstud from studente where (nome='$nome') and (cognome='$cognome')

Controllo che lo studente non abbia già svolto l'esercizio

select count(esito) from svolgimento where (ides='$ides') and (idstud='$idstud')

Inserisco il risultato dell'esercizio

insert into svolgimento(esito, ides, idstud) values ('$esito', '$ides', '$idstud')

LOGIN

Controllo la correttezza di userid e password, verificando la presenza di una tupla con quella coppia di valori

*select * from docenti where userid = '$user' and pwd = '$pwd'*

DOCENTE

Elenco delle classi in cui il docente ha la cattedra

select i.idclasse, c.classe, c.sezione, c.annoscol from cattedra as i, classe as c where (i.userid='$user') and (i.idclasse=c.idclasse)

Elenco degli studenti di una classe

*select * from studente as s, classe as c where (s.idclasse=c.idclasse) and (s.idclasse='$classe') order by s.cognome, s.nome;*

Numero di studenti di una classe

select count(*) AS 'tot' from studente where idclasse='$classe'

Controllo che lo studente che il docente vuole inserire non sia già presente

*select * from studente where (nome='$nome') and (cognome='$cognome') and (idclasse='$idclasse')*

Inserimento di uno studente

insert into studente(idstud, nome, cognome, idclasse) values('$idstud', '$nome', '$cognome', '$idclasse')

Numero di esercizi svolti da uno studente

select COUNT(r.ides) AS 'numero' from studente s, svolgimento r, esercizio e where (s.cognome='$cognome') and (s.nome='$nome') and (s.idstud=r.idstud) and (r.ides=e.ides)

Elenco degli esercizi svolti da uno studente con relativo punteggio

select r.idstud, r.ides, e.testo, r.esito, e.valore from svolgimento r, esercizio e where (r.idstud='$idstud') and (r.ides=e.ides) order by r.ides

Somma dei punteggi ottenuti negli esercizi svolti da uno studente

select sum(r.esito) AS 'tot', count(r.ides) AS 'num' from studente s, svolgimento r where (s.cognome='$cognome') and (s.nome='$nome') and (s.idstud=r.idstud)

Somma dei valori degli esercizi svolti da uno studente

select sum(e.valore) from svolgimento r, esercizio e where (r.ides=e.ides) and (r.idstud='$idstud')

Modifica dei dati di uno studente

replace into studente(idstud, nome, cognome, idclasse) values('$idstud', '$nome', '$cognome', '$idclasse')

Cancellazione di uno studente

delete from studente where (idstud='$id')

Cancellazione del risultato di un esercizio

delete from svolgimento where (ides='$ides') and (idstud='$idstud')

AMMINISTRATORE

Controllo che il docente abbia diritti di amministratore

*select * from docente where userid = '$user' and pwd = '$pwd' and diritti='amministratore'*

Inserimento o Modifica di una classe

replace into classe(classe, sezione, annoscol) values('$classe', '$sezione', '$annoscol')

Elenco delle classi

*select * from classe order by classe, sezione, annoscol*

Inserimento o Modifica di un docente

replace into docente(userid, pwd, nome, cognome, diritti)values('$userid', '$pwd', '$nome', '$cognome', '$dirdoc')

Cancellazione di una classe

delete from classe where (idclasse='$idclasse')

Cancellazione di un docente

delete from docente where (userid='$userid')

Elenco dei docenti

*select * from docente order by cognome, nome*

Cancellazione di tutte le cattedre di un docente

delete from cattedra where userid='$userid'

Inserimento di una cattedra di un docente

insert into cattedra(idclasse, userid) values('$value', '$userid')

Visualizzazione delle informazioni sugli esercizi

*select * from esercizio*

Numero di esercizi contenuti nel database

select count(*) AS 'tot' from esercizio

Test della base di dati

Per testare la base di dati, si è implementato un sito mediante il linguaggio PHP che permetta di gestire facilmente la base di dati. Il sito è hostato sul dominio http://www.daddy88.com/algebrando
Il programma software Algebrando può essere scaricato dal sito http://www.daddy88.com/algebrando/setup.exe
Una volta installato, è possibile, inserendo un nome di un alunno presente nella base di dati, misurare la propria preparazione sull'algebra dei numeri relativi.

Davide Valeriani