Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria "Enzo Ferrari" di Modena

Corso di laurea in ingegneria informatica

REALIZZAZIONE DI UN DATABASE PER L'INTEGRAZIONE DEI VOTI PROVENIENTI DALL'APPLICATIVO ESSE3

Relatore:

Dott. Ing. Laura Po

Candidato:

Romel Nelson Djoumessi

Anno Accademico 2011/2012

Indice generale

INTRODUZIONE

Esse3 è il sistema ideato dal KION-CINECA per la segreteria e i servizi agli studenti, è adottato da ben 64 atenei italiani (dato pubblicato da KION). Uno delle attività principale di Esse3 è che gli aspetti gestionali sono strettamente collegati alla presentazione e alla fruizione di pagine web in un unico ambiente coerente e facilmente accessibile. L'idea che i dati relativi ad un insegnamento inseriti da un operatore della segreteria didattica o dalla facoltà, siano gli stessi che automaticamente una funzione web visualizza sul sito pubblico di presentazione di un corso di laurea, gli stessi con cui la segreteria studenti opera riconoscimenti o le convalide di esami, gli stessi che lo studente visualizza sul "libretto virtuale" nella sua area web riservata, rappresenta per certi versi una vera e propria rivoluzione all'interno di una realtà dove spesso questi dati venivano – nel migliore dei casi - travasati più volte da un sistema all'altro, oppure circolavano su file excel, o addirittura solo su carta.
ESSE3 gestisce tutti i processi che ruotano attorno all'attività didattica nel suo complesso, da quelli di natura strettamente amministrativa (immatricolazione, iscrizione, trasferimenti, ecc..) a quelli più generali di supporto all'erogazione-fruizione della didattica stessa (piano degli studi on- line, pubblicazione di materiali dei corsi, forum di discussione fra studenti, prenotazione esami, ecc..).

Esse3 non permette di fare un'interrogazione trasversale sui dati degli appelli degli esami inseriti che è ciò che prefigge la tesi.

La tesi si prefigge di analizzare e realizzare una base di dati che integri le informazioni sugli esami degli studenti memorizzati su Esse3.

Attualmente, Esse3 permette di salvare i dati ed esportarli su file excel.

La tesi ha lo scopo di definire lo schema del database che manterrà i dati di tutti gli esami degli studenti ed elaborare una procedura software che permetta di caricare i dati provenienti dai file excel di Esse3 integrandoli nel database.

Sul database verranno poi create viste per fornire una visione d'insieme dell'andamento degli esami per un determinato insegnamento in un certo periodo temporale (per esempio per anno accademico).

SPECIFICA DEI REQUISITI

Si vuole realizzare un sistema informativo che gestisca gli esami e le valutazioni ottenute dagli studenti.

Gli studenti alla loro prima immatricolazione all'università registrano i propri dati anagrafi (nome, cognome, data di nascita, residenza, email e il codice fiscale) e si iscrivono ad un corso di laurea. La segreteria studenti assegna, indipendentemente dalla facoltà scelta e dal corso di laurea scelto, una matricola univoca all'interno dell'università.

Il database memorizza i corsi di laurea. Ogni corso di laurea è identificato da un codice univoco all'interno della facoltà di afferenza ed ha un nome che lo caratterizza.

Per un corso di laurea si hanno diversi insegnamenti. Ogni insegnamento ha un codice univoco all'interno di un determinato corso di laurea, il sistema memorizza il suo nome, il numero di crediti affidati e la sua descrizione. Ogni insegnamento è tenuto da un docente titolare . Ogni docente è identificato da un codice docente, ed ha un nome, un cognome ed il numero del suo ufficio.

Durante l'anno accademico si alternano diverse sessioni (estiva, invernale e straordinaria), ogni sessione è contraddistinta da una data di inizio e di fine.

Il regolarmento didattico dell'ateneo prevede che per ogni insegnamento in una sessione venga definito almeno un appello. In occasione di un appello lo studente può svolgere un esame.

Un esame è identificato da un codice univoco ed ha un nome e un voto. Gli esami possono essere totali o parziali.

Mentre l'esame totale valuta lo studente secondo il programma completo dell'insegnamento, un esame parziale valuta soltanto la conoscenza di una parte del programma. Il docente può definire il numero di parziali necessari per superare l'esame.

Il docente definisce quali esami saranno disponibili in occasione di un appello. Ogni appello è identificato da un codice univoco, si memorizza la data, l'ora e l'aula in cui si svolge, il tipo di prova (orale o scritto o pratica).

Uno studente di un corso di laurea può iscriversi agli appelli definiti per gli insegnamenti, si memorizza la data di iscrizione.

Solo a seguito di un appello, per gli studenti che hanno sostenuto l'esame totale e ottenuto un esito sufficiente è prevista la registrazione del voto finale (verbalizzazione). Anche per gli studenti che hanno sostenuto tutti gli esami parziali previsti dall'insegnamento e hanno ottenuto un esito sufficiente è prevista la verbalizzazione.

La votazione ottenuto dallo studente è sufficiente se il voto è compreso tra 18 e 33, insufficiente nel caso contrario.

Per gli studenti che verbalizzano il voto, c' è la composizione di un verbale che fa riferimento all'insegnamento. Ogni verbale è identificato da un numero , il database memorizza il voto finale , la data di verbalizzazione e la firma del docente.

Uno studente può verbalizzare una sola volta il voto per uno specifico insegnamento.

PROGETTAZIONE CONCETTUALE

L'obiettivo della *progettazione concettuale* è quello di rappresentare le specifiche con una descrizione formale e completa senza preoccuparsi né della modalità con cui queste informazioni verranno definite in un sistema reale.

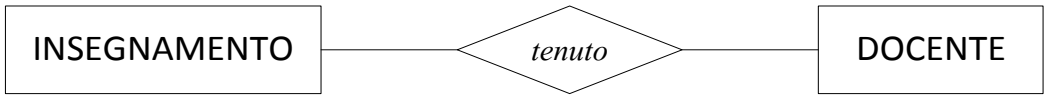
2.1 Progetto iniziale

Si deve costruire uno schema scheletro con i concetti più evidenti espressi nei requisiti. I concetti referenziati con maggiore frequenza nei requisiti sono buoni candidati per tale scelta: STUDENTE, CDL (*corso di laurea*), INSEGNAMENTO, DOCENTE , ESAME. A tali entità si impongono le associazioni più evidenti.

♦ Dai requisiti specifici , possiamo estrapolare l'entità STUDENTE che contiene i dati anagrafi (nome, cognome, data di nascita, residenza, email e il codice fiscale) e istituzionali (matricola) in relazione con l'entità corso di laurea (CDL), che raccoglie tutte le informazioni relative ad un corso di laurea di una facoltà dell'università in cui si è iscritto uno studente .

♦ Dalle specifiche, possiamo estrapolare l'entità INSEGNAMENTO che contiene tutte le informazioni sui corsi svolti all'interno di un determinato corso di laurea (CDL).

♦ Dalle specifiche possiamo estrapolare l'entità DOCENTE che contiene i dati anagrafici (nome, cognome) e amministrativi (codice docente, il numero del suo ufficio) sul docente che tiene un insegnamento .

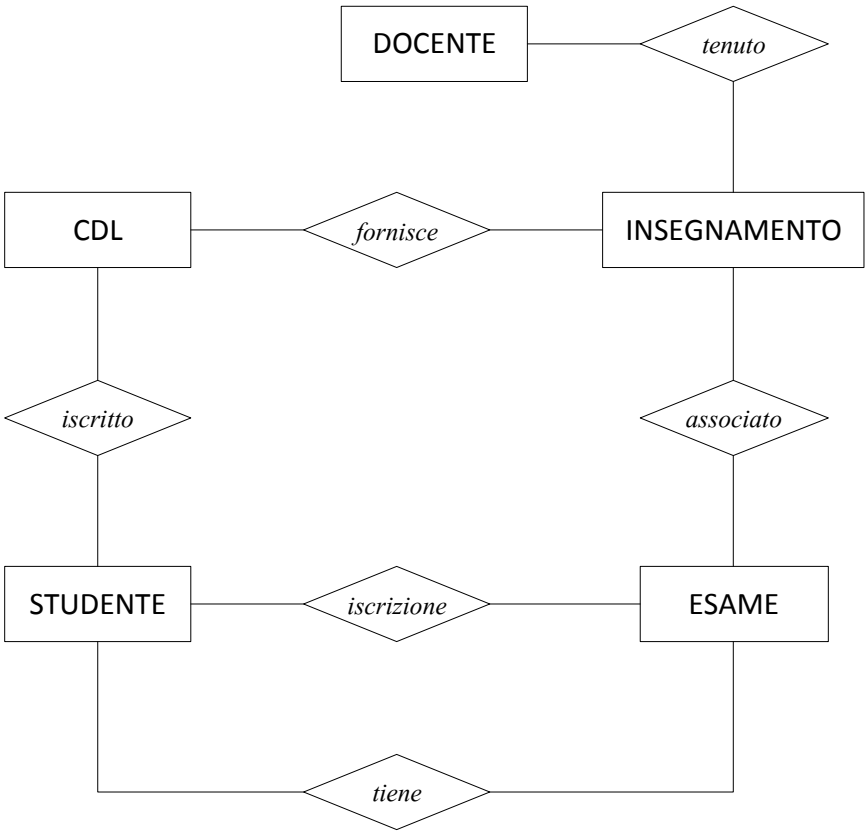


♦ Dai requisiti , possiamo ricavare l'entità ESAME che fornisce le informazioni sui esami che lo STUDENTE può iscriversi.

♦ Al fine di modellare il voto ottenuto dallo studente in un esame, scegliamo di inserire anche *tiene* tra esame e studente.

♦ Poiché ogni insegnamento ha un esame ad esso associato , quindi esiste una relazione tra l'entità INSEGNAMENTO e ESAME.

Adesso si costruisce uno schema scheletro con questi concetti più evidenti espressi nei requisiti.



Raffinamenti bottom-up

♦ Introduzione dell'entità SESSIONE che fornisce tutte le informazioni sull'intervallo temporale degli insegnamenti.

♦ Introduzione dell'entità VERBALE che fornisce tutte le informazioni sul processo di verbalizzazione.

♦ Introduzione dell'entità APPELLO che fornisce tutte le informazioni sugli appelli.

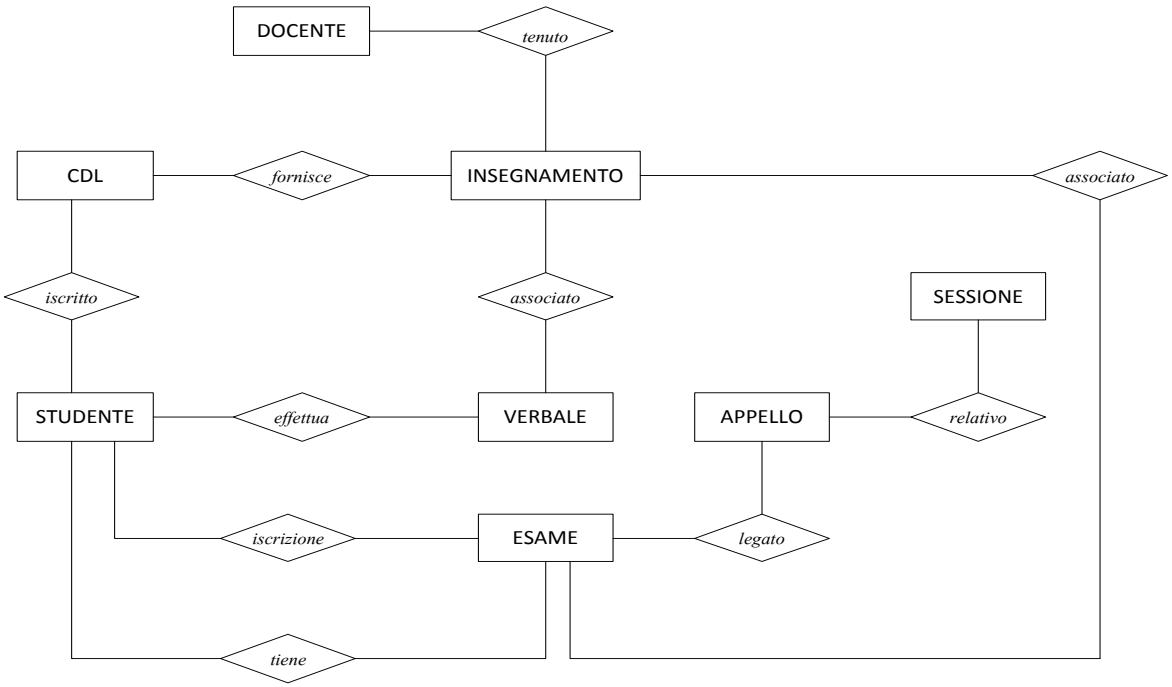
Raffinamenti inside-out

♦ Partendo da STUDENTE si rappresenta i verbali che effettuano gli studenti con l'introduzione dell'entità VERBALE e dell'associazione *effettua*.

♦ Partendo da ESAME si rappresenta tutti gli appelli relativo a questo esame con l'introduzione dell'entità APPELLO e dell'associazione *legato* .

♦ Partendo da APPELLO si rappresenta la sessione in cui si svolge l'appello con l'introduzione dell'entità SESSIONE e dell'associazione *relativo* .

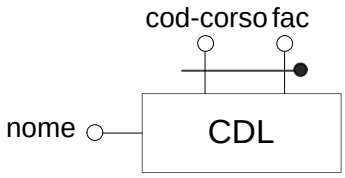
Andiamo ora a visualizzare la nuova versione del nostro scheletro rimodellato con questi informazioni aggiuntive.



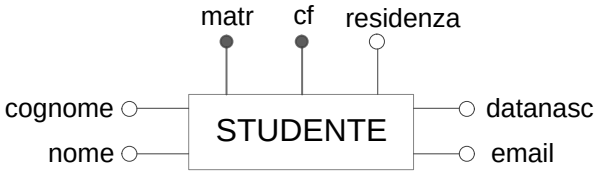
2.2 Raffinamento delle entità

2.2.1 Descrizione delle entità

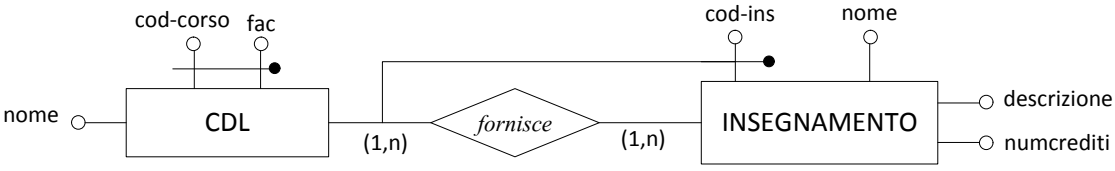
Informazioni relative ai corsi di laurea: Un corso di laurea è identificato da un codice univoco all'interno dell'università e dalla facoltà di afferenza e ha un nome che lo caratterizza.



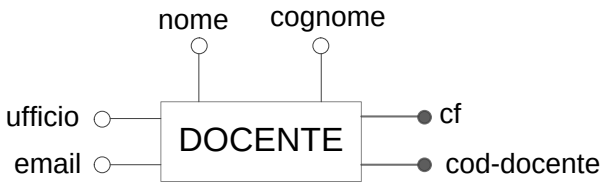
Informazioni relative agli studenti: Gli studenti alla loro prima immatricolazione all'università registrano i propri dati anagrafi (nome, cognome, data di nascita, residenza, email e il codice fiscale) e si iscrivono ad un corso di laurea. La segreteria studenti assegna, indipendentemente dalla facoltà scelta e dal corso di laurea scelto, una matricola univoca all'interno dell'università.



Informazioni relative agli insegnamenti: Ogni insegnamento ha un codice univoco all'interno di un determinato corso di laurea, il sistema memorizza il suo nome, il numero di crediti affidati ed la sua descrizione.



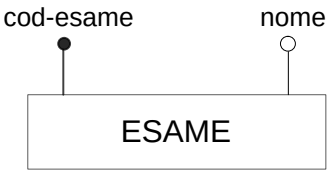
Informazioni relative ai docenti: Ogni docente è identificato da un codice docente, ed ha un nome, un cognome ed il numero del suo ufficio.



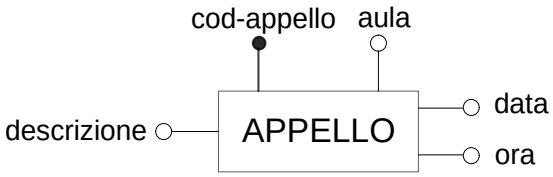
Informazioni relative alla sessione: Durante l'anno accademico si alternano diverse sessioni (estiva, invernale e straordinaria), ogni sessione è contraddistinta da una data di inizio e di fine.

Supponiamo che non esistano due sessioni con lo stesso nome in un determinato anno.

Informazioni relative all'esame: Un esame è identificato da un codice univoco e ha un nome che lo caratterizza.

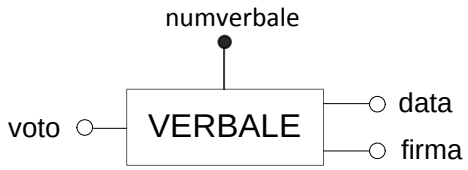


Informazioni relative all'appello: Ogni appello è identificato da un codice univoco, il database memorizza la data, l'ora , l'aula in cui si è svolto, il tipo dell'appello (orale o scritto o pratica) ed eventualmente la votazione ottenuto dallo studente.



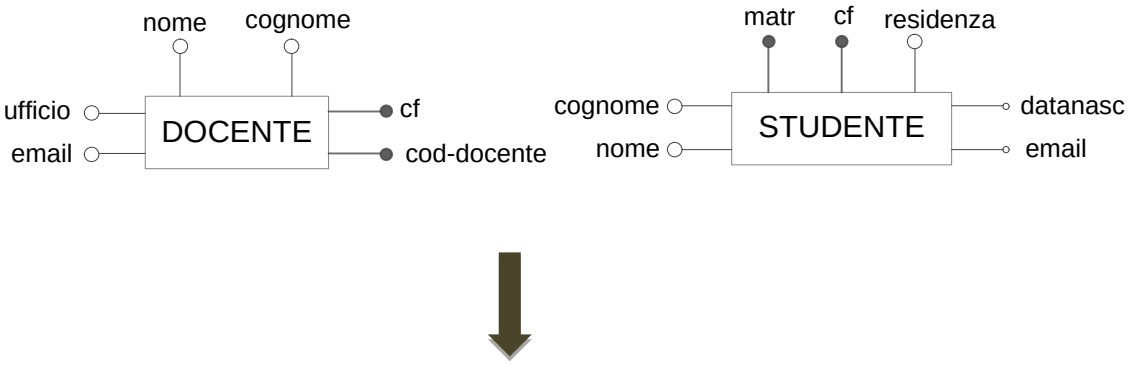
Dominio descrizione = {orale, scritto, pratica}

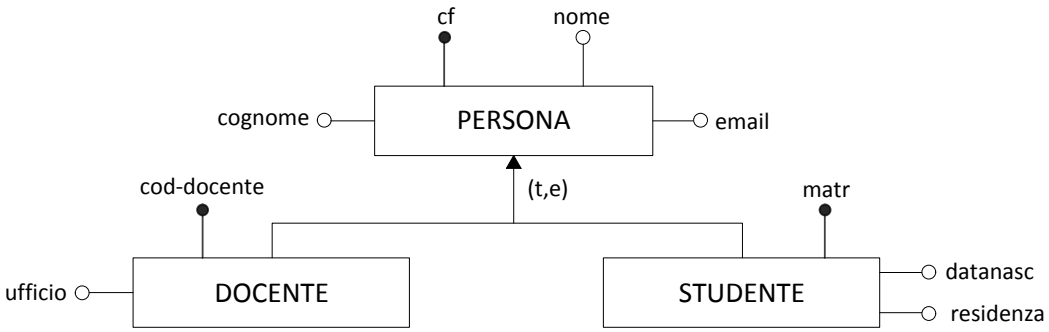
Informazioni relative alla verbalizzazione: Ogni verbale è identificato dalla matricola dello studente che effettua la verbalizzazione, il database memorizza il suo voto definitivo , la data di verbalizzazione e la firma del docente.



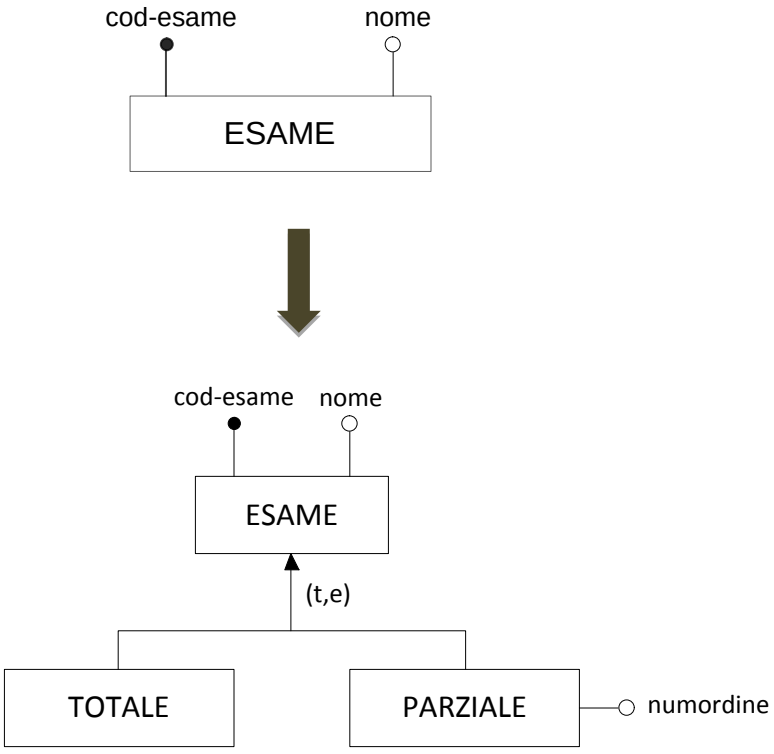
2.2.2 Ulteriori raffinamenti

♦ Osservando gli attributi delle entità STUDENTE e DOCENTE si scopre delle similarità, possono essere generalizzate in un sola entità PERSONA.



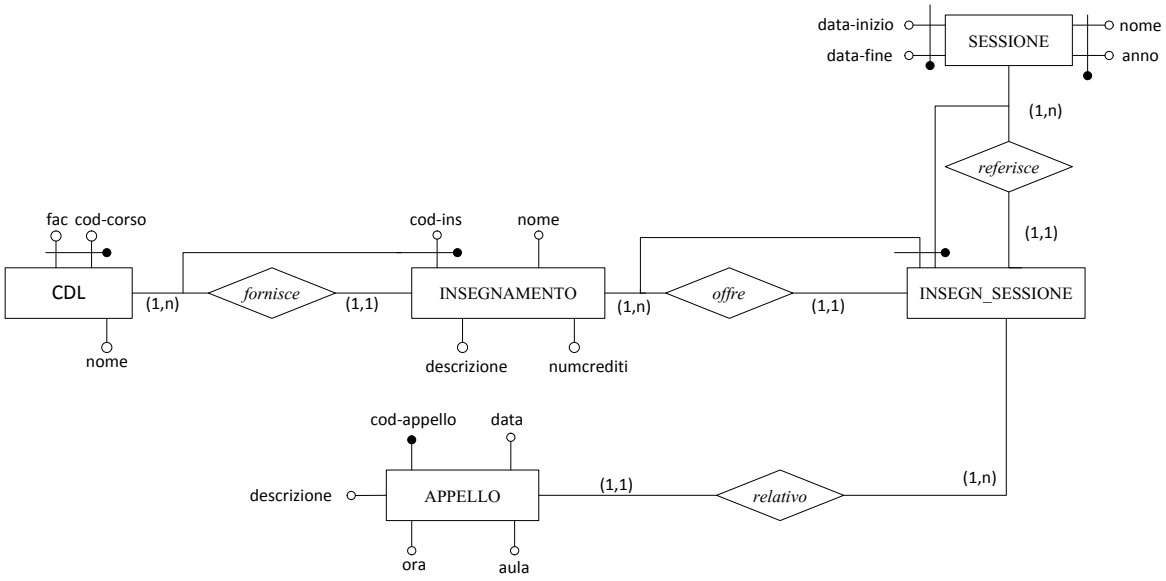


♦ Siccome un esame può essere totale (gli argomenti dell'esame riguardano tutto il programma dell'insegnamento) o parziale (gli argomenti sono relativi soltanto a una parte del programma) . Possiamo decidere di suddividere l'entità ESAME in due sottoentità TOTALE e PARZIALE.

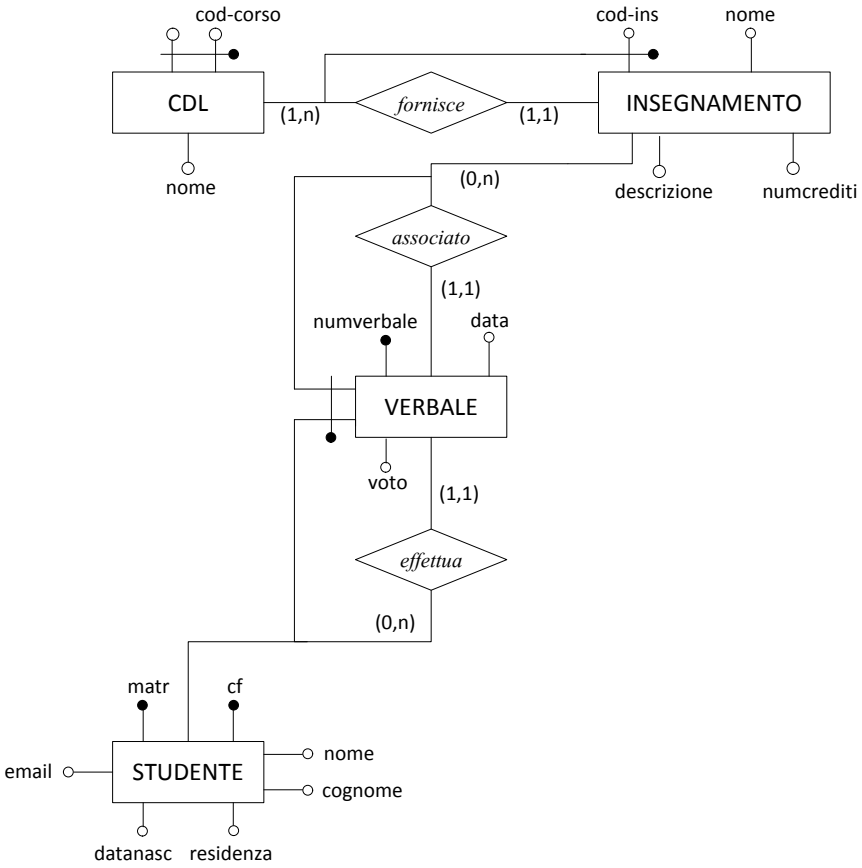


2.2.3 Vincoli del problema

♦ Il regolarmento didattico dell'ateneo prevede che per ogni insegnamento in una sessione venga definito almeno un appello.



♦ Uno studente può verbalizzare una sola volta il voto per uno specifico insegnamento.



A questo punto dobbiamo completare lo schema concettuale finale aggiungendo: attributi, identificatori, vincoli di integrità, cardinalità delle associazioni e coperture delle gerarchie.

2.3 Schema E/R finale

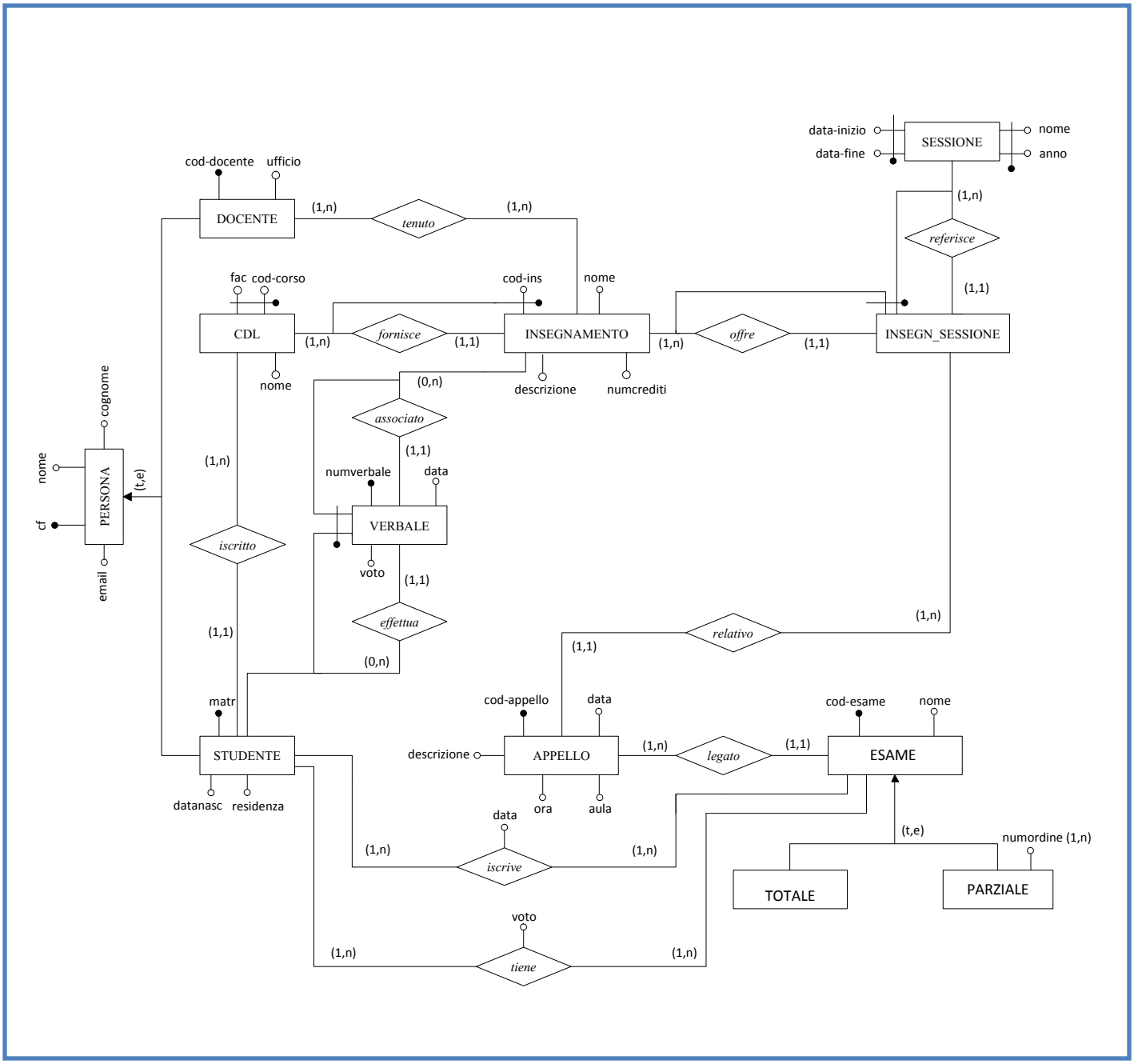


Figura 1: Schema E/R finale

2.3.1 Dizionario dei dati

ENTITA'	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORI
CDL (corso di laurea)	Corso di laurea in cui si immatricola lo studente	cod-corso, fac, nome	cod-corso fac
STUDENTE	Studente universitario	nome, cognome, cf, matr, datanasc, email, residenza	matr
INSEGNAMENTO	Corso svolto	cod-ins, nome, decrizione, numcrediti, cod-corso, fac	cod-ins fac cod-corso
DOCENTE	Docente universitario	nome, cognome, ufficio, cod-docente, cf, email	cod-docente
SESSIONE	Intervallo temporale	data-inizio, nome, anno, data-fine	data-inizio nome anno data-fine
TOTALE	Uno dei tipi d'esame	cod-esame, nome, voto	cod-esame
PARZIALE	Uno dei tipi d'esame	cod-esame, nome, voto, numordine (1,n)	cod-esame
APPELLO	Appello effettuato	tipo , cod-appello, data, ora, voto	cod-appello
VERBALE	Registrazione del voto	voto, data, firma, numverbale	numverbale

RELAZIONE	DESCRIZIONE	ENTITA' COINVOLTE	ATTRIBUTI
Iscritto	Associa lo studente al suo corso di laurea (CDL)	STUDENTE (1,1) CDL (1,N)	—
Fornisce	Associa ad un corso di laurea (CDL) gli insegnamenti che si svolgono.	CDL (1,N) INSEGNAMENTO (1,N)	—
Tenuto	Associa ad ogni insegnamento il docente che lo tiene.	DOCENTE (1,N) INSEGNAMENTO (1,N)	—
Iscrive	Associa lo studente all'esame in cui si è registrato.	STUDENTE (1,N) ESAME (1,N)	data
Tiene	Associa lo studente all' esame sostenuto.	STUDENTE (1,N) ESAME (1,N)	voto
Relativo	Associa l'appello alla sua sessione.	APPELLO (1,1) SESSIONE (1,N)	—
Legato	Associa un esame completo al suo appello	ESAME (1,1) APPELLO (1,N)	—
Effettua	Associa lo studente al verbale.	STUDENTE (1,N) VERBALE (1,1)	—

Associato	Associa l'insegnamento al verbale .	APPELLO (1,N) VERBALE (1,1)	__

2.3.2 Regole

REGOLE DI VINCOLO	
RV1	Non sono ammesse duplicazioni di tuple per ciascuna tabella
RV2	Gli identificatori di tutte le entità non possono assumere valori nulli
RV3	Uno studente non può essere iscritto a due corsi di laurea (CDL) contemporaneamente della stessa università.
RV4	Una matricola è univoca ad l'interno di un corso di laurea
RV5	Non sono ammesse più di tre appelli per un insegnamento in una sessione (espressa)
RV6	Un corso di laurea (CDL) ha almeno uno studente iscritto (espressa)

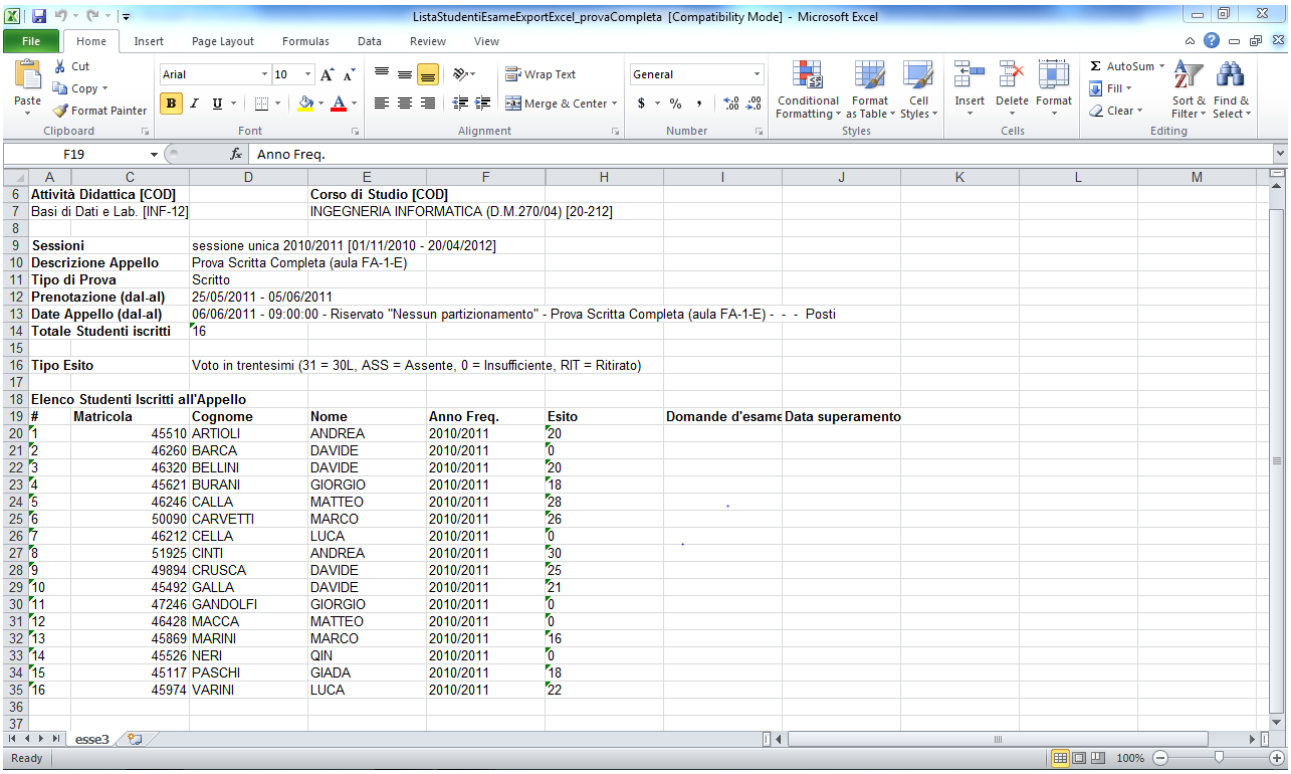
♦ Non tutte le regole di vincolo potranno trovare una implementazione nella base di dati.

MODIFICA DELLO SCHEMA E/R SULLA BASE DEI DATI ESPORTATI DALL'APPLICAZIONE ESSE3

3.1 Descrizione dell'applicazione Esse3

L'applicazione Esse3 permette di gestire l'intero processo relativo agli appelli d'esame: creazione, iscrizione studenti, pubblicazione voti e verbalizzazione. I dati gestiti dall'applicativo possono essere esportati in un file XLS.

Di seguito un esempio di file excel che esporta i dati e i voti degli studenti iscritti ad un appello.

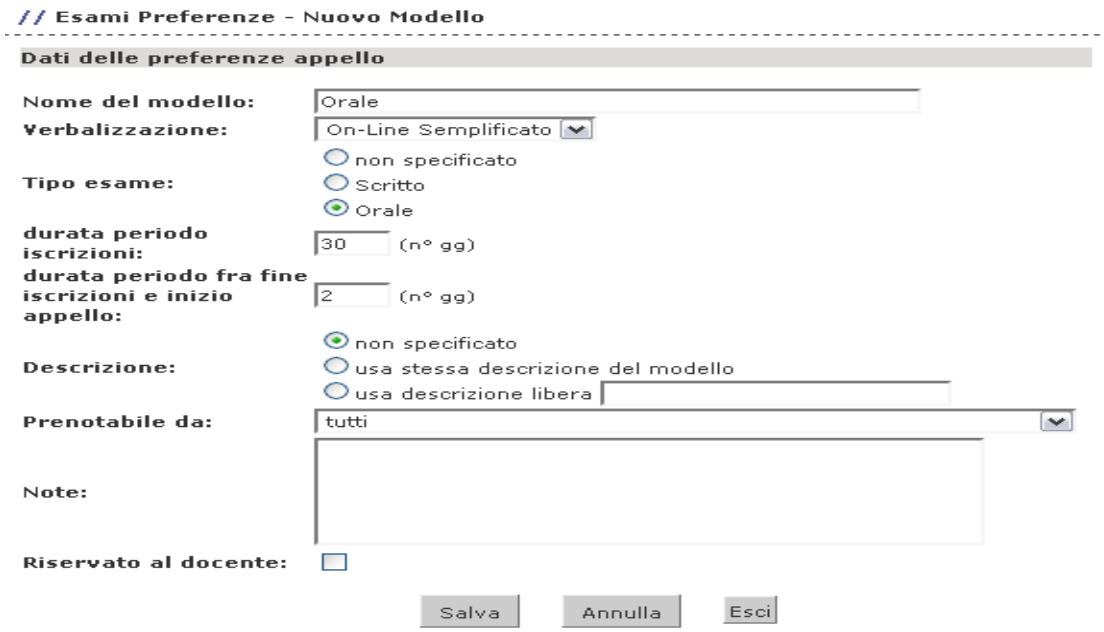


ListaStudentiEsameExportExcel_provaCompleta [Compatibility Mode] - Microsoft Excel

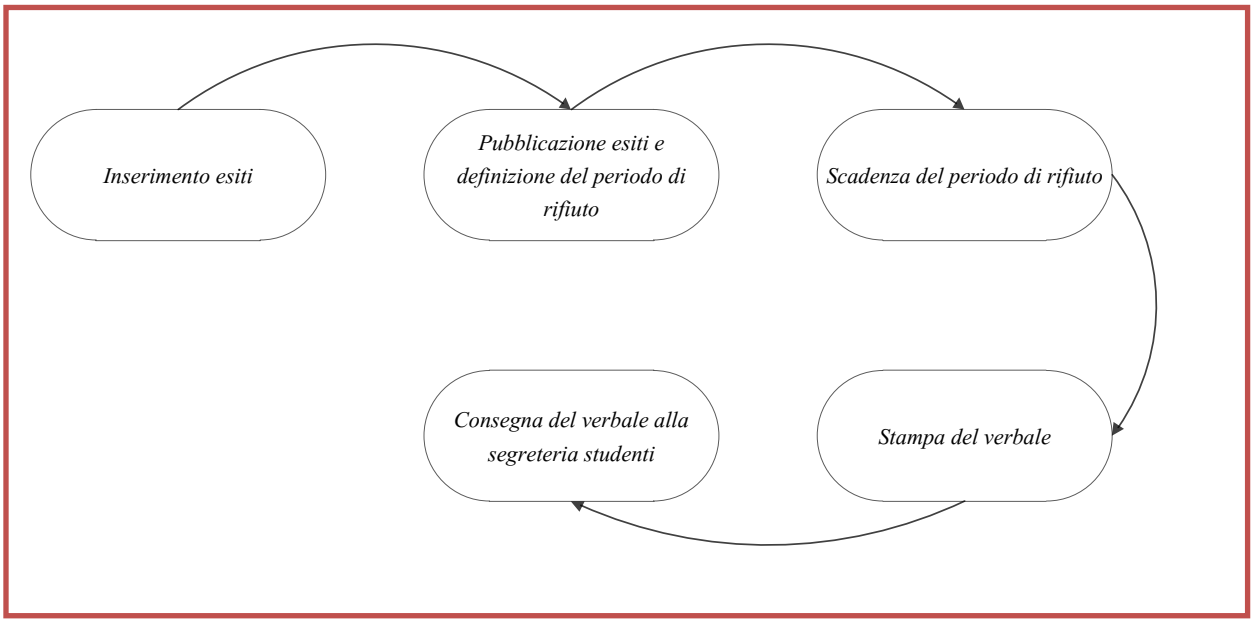
6	**Attività Didattica [COD]**		**Corso di Studio [COD]**								
7	Basi di Dati e Lab. [INF-12]		INGEGNERIA INFORMATICA (D.M.270/04) [20-212]								
8											
9	**Sessioni**	sessione unica 2010/2011 [01/11/2010 - 20/04/2012]									
10	**Descrizione Appello**	Prova Scritta Completa (aula FA-1-E)									
11	**Tipo di Prova**	Scritto									
12	**Prenotazione (dal-al)**	25/05/2011 - 05/06/2011									
13	**Date Appello (dal-al)**	06/06/2011 - 09:00:00 - Riservato "Nessun partizionamento" - Prova Scritta Completa (aula FA-1-E) - - - Posti									
14	**Totale Studenti iscritti**	16									
15											
16	**Tipo Esito**	Voto in trentesimi (31 = 30L, ASS = Assente, 0 = Insufficiente, RIT = Ritirato)									
17											
18	**Elenco Studenti Iscritti all'Appello**										
19	**#**	**Matricola**	**Cognome**	**Nome**	**Anno Freq.**	**Esito**	**Domande d'esame**	**Data superamento**			
20	1	45510	ARTIOLI	ANDREA	2010/2011	20					
21	2	46260	BARCA	DAVIDE	2010/2011	0					
22	3	46320	BELLINI	DAVIDE	2010/2011	20					
23	4	45621	BURANI	GIORGIO	2010/2011	18					
24	5	46246	CALLA	MATTEO	2010/2011	28					
25	6	50090	CARVETTI	MARCO	2010/2011	26					
26	7	46212	CELLA	LUCA	2010/2011	0					
27	8	51925	CINTI	ANDREA	2010/2011	30					
28	9	49894	CRUSCA	DAVIDE	2010/2011	25					
29	10	45492	GALLA	DAVIDE	2010/2011	21					
30	11	47246	GANDOLFI	GIORGIO	2010/2011	0					
31	12	46428	MACCA	MATTEO	2010/2011	0					
32	13	45869	MARINI	MARCO	2010/2011	16					
33	14	45526	NERI	QIN	2010/2011	0					
34	15	45117	PASCHI	GIADA	2010/2011	18					
35	16	45974	VARINI	LUCA	2010/2011	22					
36											
37											

3.1.1 Verbalizzazione Esse3

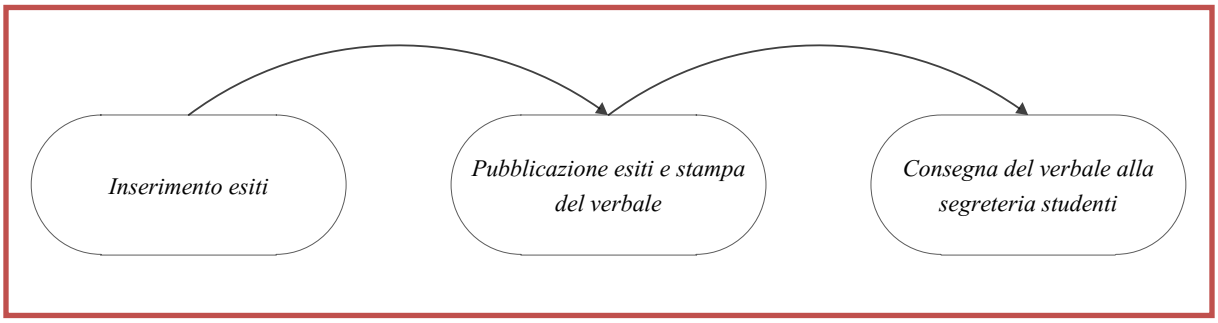
L'applicativo Esse3 prevede tre tipologie di verbalizzazione per un esame, per venire incontro alle diverse necessità espresse dai docenti:



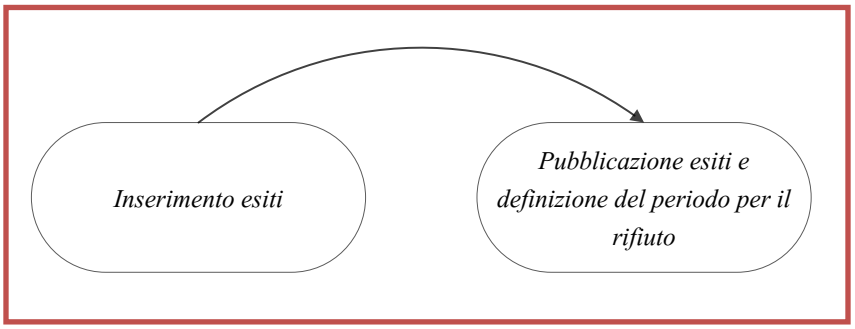
• **ON-LINE**: consigliato per esami scritti. La tipologia On-line consente di definire un periodo per il rifiuto, entro il quale lo studente può decidere se accettare o meno l'esito dell'esame. Soltanto quando sarà trascorso tale periodo (ovvero dal giorno successivo alla scadenza del periodo concesso per il rifiuto) sarà possibile procedere alla stampa del verbale.



• **ON-LINE SEMPLIFICATO**: consigliato per esami orali. Diversamente dalla tipologia **ON-LINE** non consente di definire un periodo per il rifiuto, pertanto l'operazione di stampa del verbale coincide con quella di pubblicazione degli esiti.



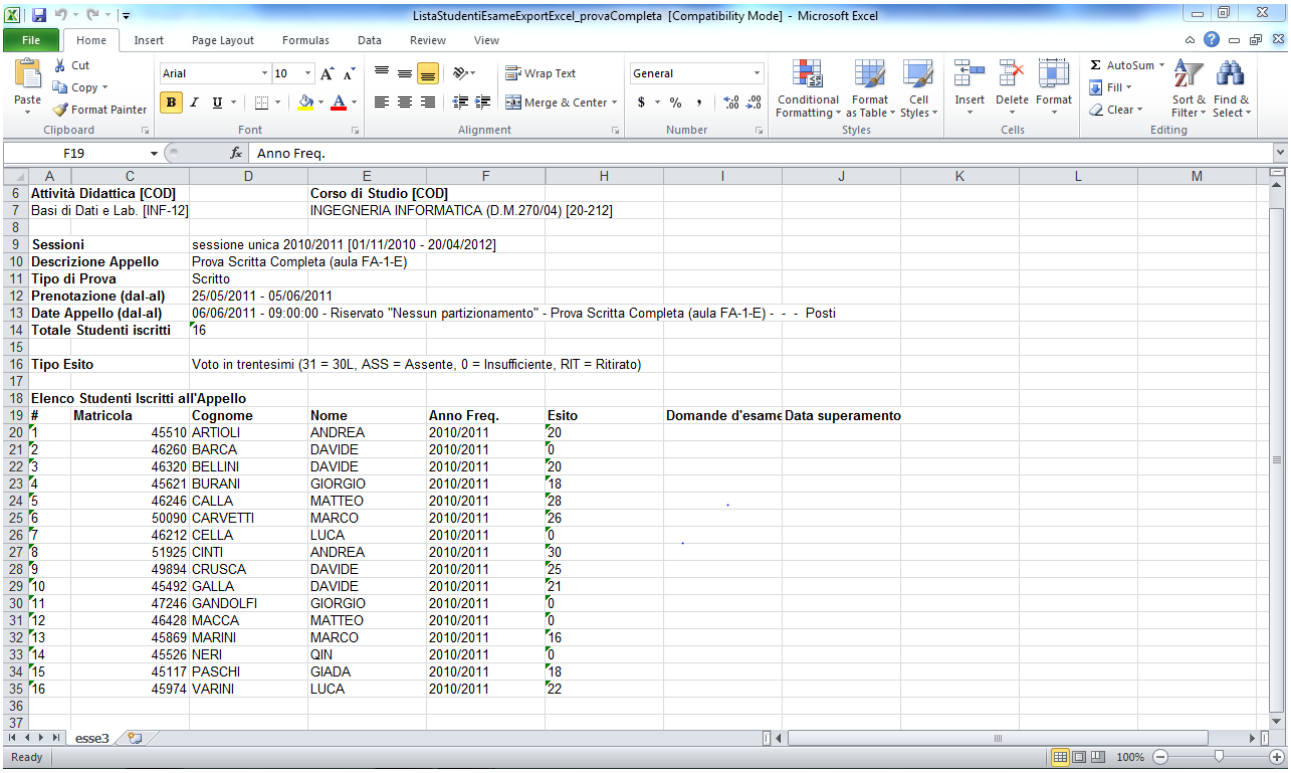
• **STANDARD**: consigliato per prove parziali. Questo tipo di verbalizzazione non produce un verbale. Consente agli studenti di iscriversi e successivamente permette la pubblicazione degli esiti ma non è finalizzato dalla stampa del verbale. L'esito inserito e pubblicato non sarà il voto finale dell'esame.



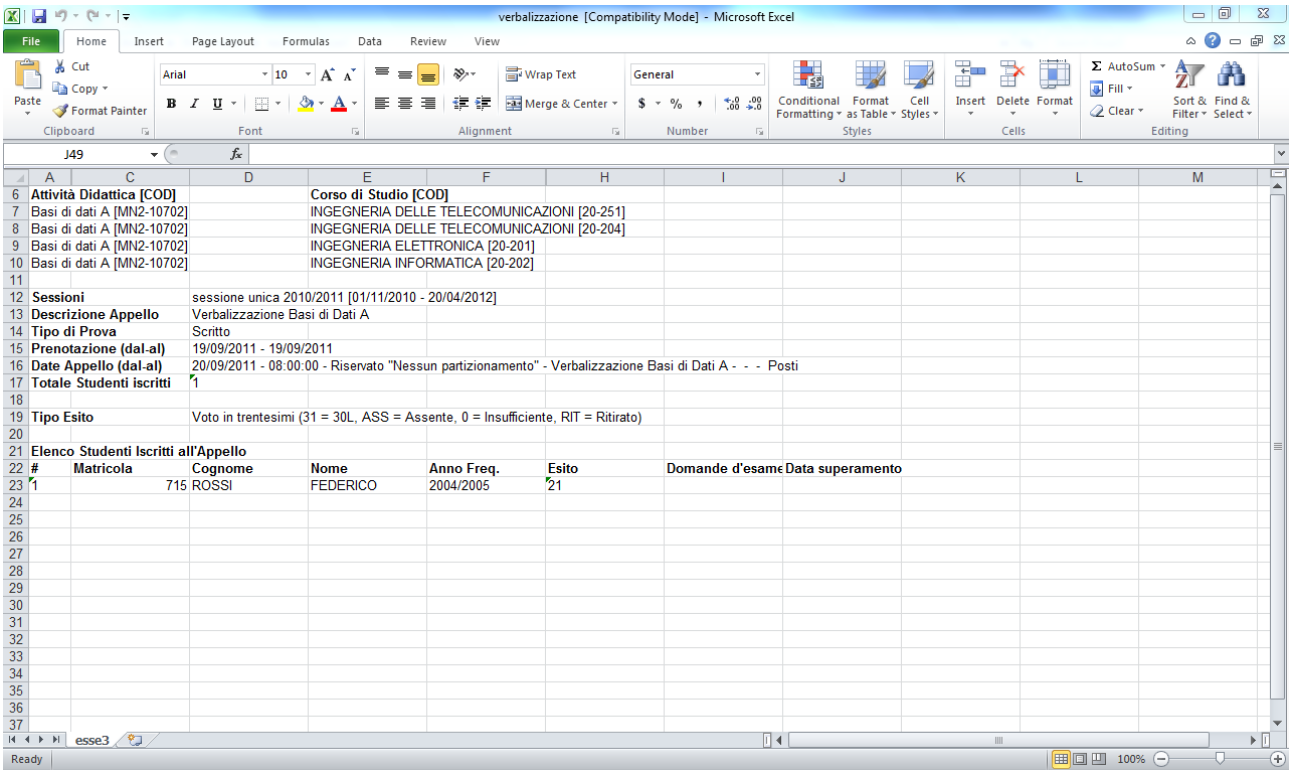
E' molto importante fare attenzione alla tipologia di verbalizzazione definita, poiché dal momento in cui si iscrive anche uno solo studente all'appello non sarà più possibile modificarla.

3.2 Modifiche apportate allo schema E/R

Siccome l'obiettivo della tesi è l'integrazione nel database esami di dati provenienti da tale applicativo, lo scopo di questa sessione sarà quello di analizzare i file esportati e verificare che le informazioni in essi contenute possono essere mappate all'interno dello schema E/R prodotto seguendo le specifiche.



6	**Attività Didattica [COD]**			**Corso di Studio [COD]**							
7	Basi di Dati e Lab. [INF-12]			INGEGNERIA INFORMATICA (D.M.270/04) [20-212]							
8											
9	**Sessioni**		sessione unica 2010/2011 [01/11/2010 - 20/04/2012]								
10	**Descrizione Appello**		Prova Scritta Completa (aula FA-1-E)								
11	**Tipo di Prova**		Scritto								
12	**Prenotazione (dal-al)**		25/05/2011 - 05/06/2011								
13	**Date Appello (dal-al)**		06/06/2011 - 09:00:00 - Riservato "Nessun partizionamento" - Prova Scritta Completa (aula FA-1-E) - - - Posti								
14	**Totale Studenti iscritti**		16								
15											
16	**Tipo Esito**		Voto in trentesimi (31 = 30L, ASS = Assente, 0 = Insufficiente, RIT = Ritirato)								
17											
18	**Elenco Studenti Iscritti all'Appello**										
19	**#**	**Matricola**	**Cognome**	**Nome**	**Anno Freq.**	**Esito**	**Domande d'esame**	**Data superamento**			
20	1	45510	ARTIOLI	ANDREA	2010/2011	20					
21	2	46260	BARCA	DAVIDE	2010/2011	0					
22	3	46320	BELLINI	DAVIDE	2010/2011	20					
23	4	45621	BURANI	GIORGIO	2010/2011	18					
24	5	46246	CALLA	MATTEO	2010/2011	28					
25	6	50090	CARVETTI	MARCO	2010/2011	26					
26	7	46212	CELLA	LUCA	2010/2011	0					
27	8	51925	CINTI	ANDREA	2010/2011	30					
28	9	49894	CRUSCA	DAVIDE	2010/2011	25					
29	10	45492	GALLA	DAVIDE	2010/2011	21					
30	11	47246	GANDOLFI	GIORGIO	2010/2011	0					
31	12	46428	MACCA	MATTEO	2010/2011	0					
32	13	45869	MARINI	MARCO	2010/2011	16					
33	14	45526	NERI	QIN	2010/2011	0					
34	15	45117	PASCHI	GIADA	2010/2011	18					
35	16	45974	VARINI	LUCA	2010/2011	22					



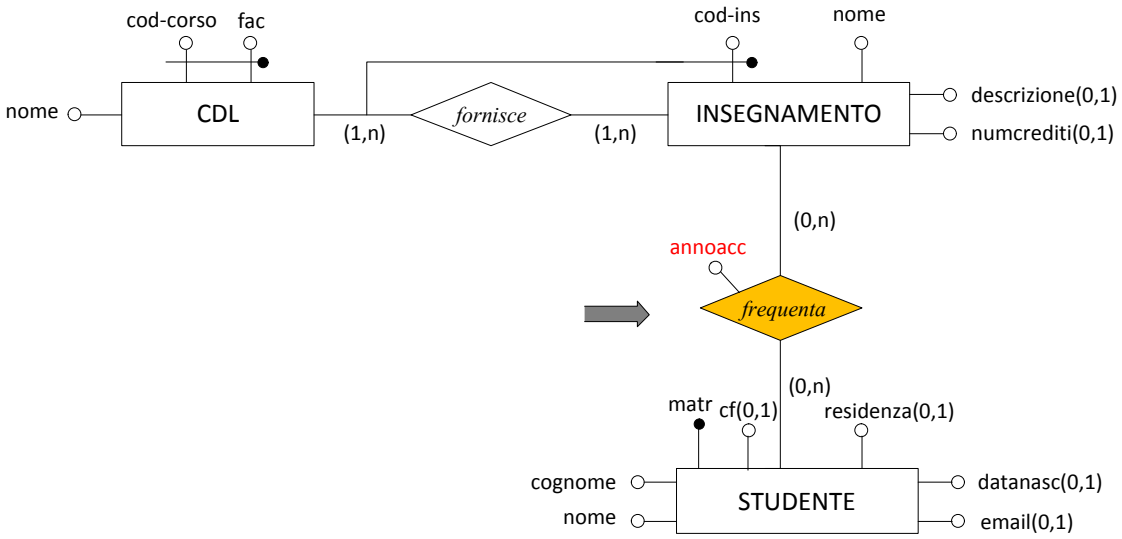
6	**Attività Didattica [COD]**			**Corso di Studio [COD]**							
7	Basi di dati A [MN2-10702]			INGEGNERIA DELLE TELECOMUNICAZIONI [20-251]							
8	Basi di dati A [MN2-10702]			INGEGNERIA DELLE TELECOMUNICAZIONI [20-204]							
9	Basi di dati A [MN2-10702]			INGEGNERIA ELETTRONICA [20-201]							
10	Basi di dati A [MN2-10702]			INGEGNERIA INFORMATICA [20-202]							
11											
12	**Sessioni**		sessione unica 2010/2011 [01/11/2010 - 20/04/2012]								
13	**Descrizione Appello**		Verbalizzazione Basi di Dati A								
14	**Tipo di Prova**		Scritto								
15	**Prenotazione (dal-al)**		19/09/2011 - 19/09/2011								
16	**Date Appello (dal-al)**		20/09/2011 - 08:00:00 - Riservato "Nessun partizionamento" - Verbalizzazione Basi di Dati A - - - Posti								
17	**Totale Studenti iscritti**		1								
18											
19	**Tipo Esito**		Voto in trentesimi (31 = 30L, ASS = Assente, 0 = Insufficiente, RIT = Ritirato)								
20											
21	**Elenco Studenti Iscritti all'Appello**										
22	**#**	**Matricola**	**Cognome**	**Nome**	**Anno Freq.**	**Esito**	**Domande d'esame**	**Data superamento**			
23	1	715	ROSSI	FEDERICO	2004/2005	21					

Analizzando in dettagli questi file, possiamo estrapolare alcuni concetti che andremo a modellare nel nostro schema E/R.

♦ Dall'analisi di questi file si evince che ogni studente che si iscrive(ed eventualmente sostiene) un esame, ha frequentato il relativo insegnamento in uno specifico anno accademico (*annoacc*). Per modellare tale informazione è necessario aggiungere un'associazione (**frequenta**) nello schema E/R tra le entità INSEGNAMENTO e STUDENTE, ed aggiungere l'attributo *annoacc* su quest'associazione.

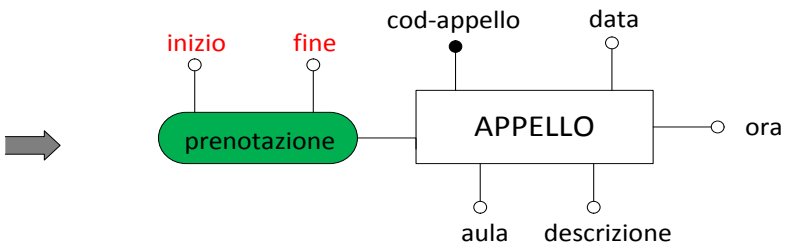
Siccome il codice fiscale (*CF*) non è un'informazione di studente contenuta nel file esportato da esse3, così come la data di nascita (*datanasc*), la residenza e l'email, tali attributi devono diventare attributi opzionali , cioè con card(0,1).

Poiché il numero di crediti (*numcrediti*) di un relativo insegnamento è un' informazione che non abbiamo, e nemmeno la descrizione di un insegnamento (*descrizione*), anche tali attributi dovranno diventare attributi opzionali, cioè con card(0,1).



♦ Un appello ha inoltre un intervallo di date che definisce la finestra temporale in cui ci si può iscrivere all'appello.

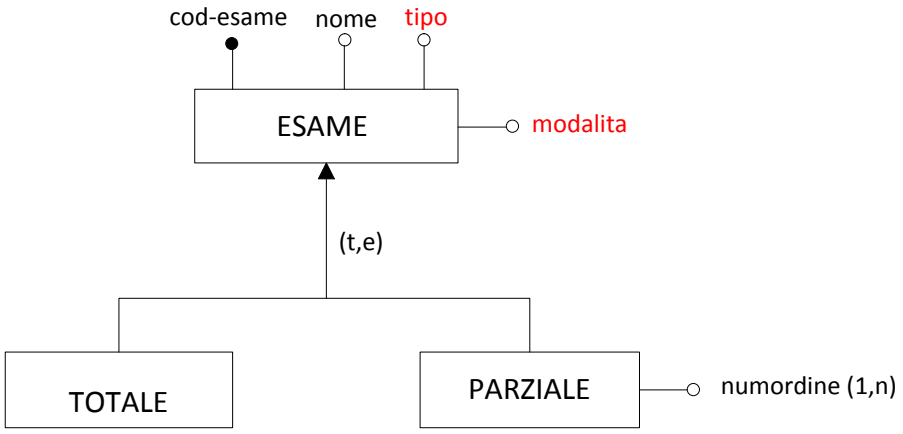
Abbiamo deciso di modellare tale informazione attraverso un attributo composto *prenotazione* dell'entità APPELLO che contiene le informazioni relative alla data d'inizio e di fine della prenotazione di un appello.



♦ Dall'analisi di questi file possiamo introdurre l'attributo *modalita* sull'entità ESAME che contiene l'informazione sulla scala di valutazione utilizzata l'applicazione ESSE3 gestisce infatti diversi metodi di valutazione:
Voto(espresso in trentesimi (31 = 30L,ASS = Assente, 0 = Insufficiente, RIT = Ritirato))
Giudizio1(Approvato/Non approvato)
Giudizio2(Altro/Assente/Respinto/Ritirato/Abbandono)
Giudizio3(Idoneo/Non idoneo)
Giudizio4(Ottimo/Distinto/Buono/Ritirato/Discreto/sufficiente/Insufficiente)

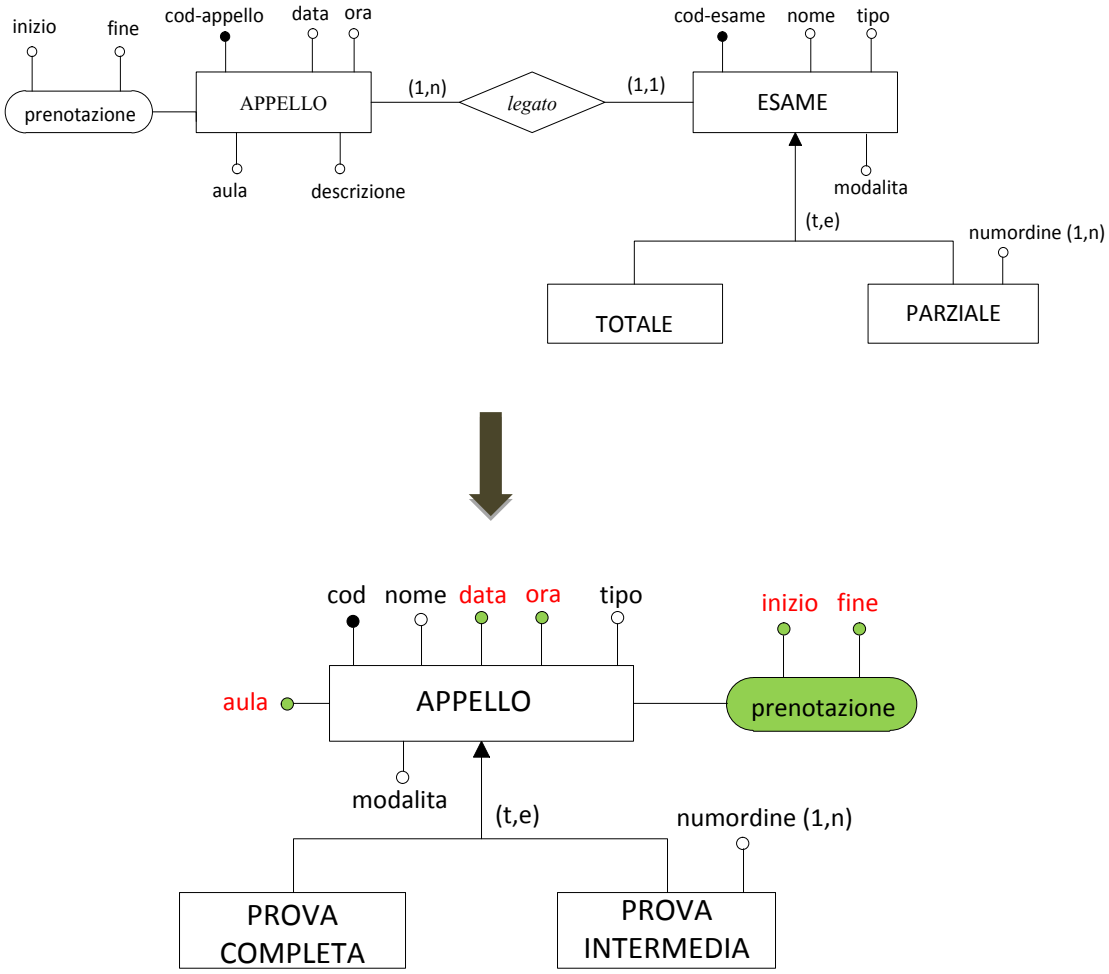
Possiamo introdurre l'attributo *tipo* sull'entità ESAME che contiene l'informazione sul tipo d'esame svolto.

Dominio di *tipo* = {scritto, orale}.

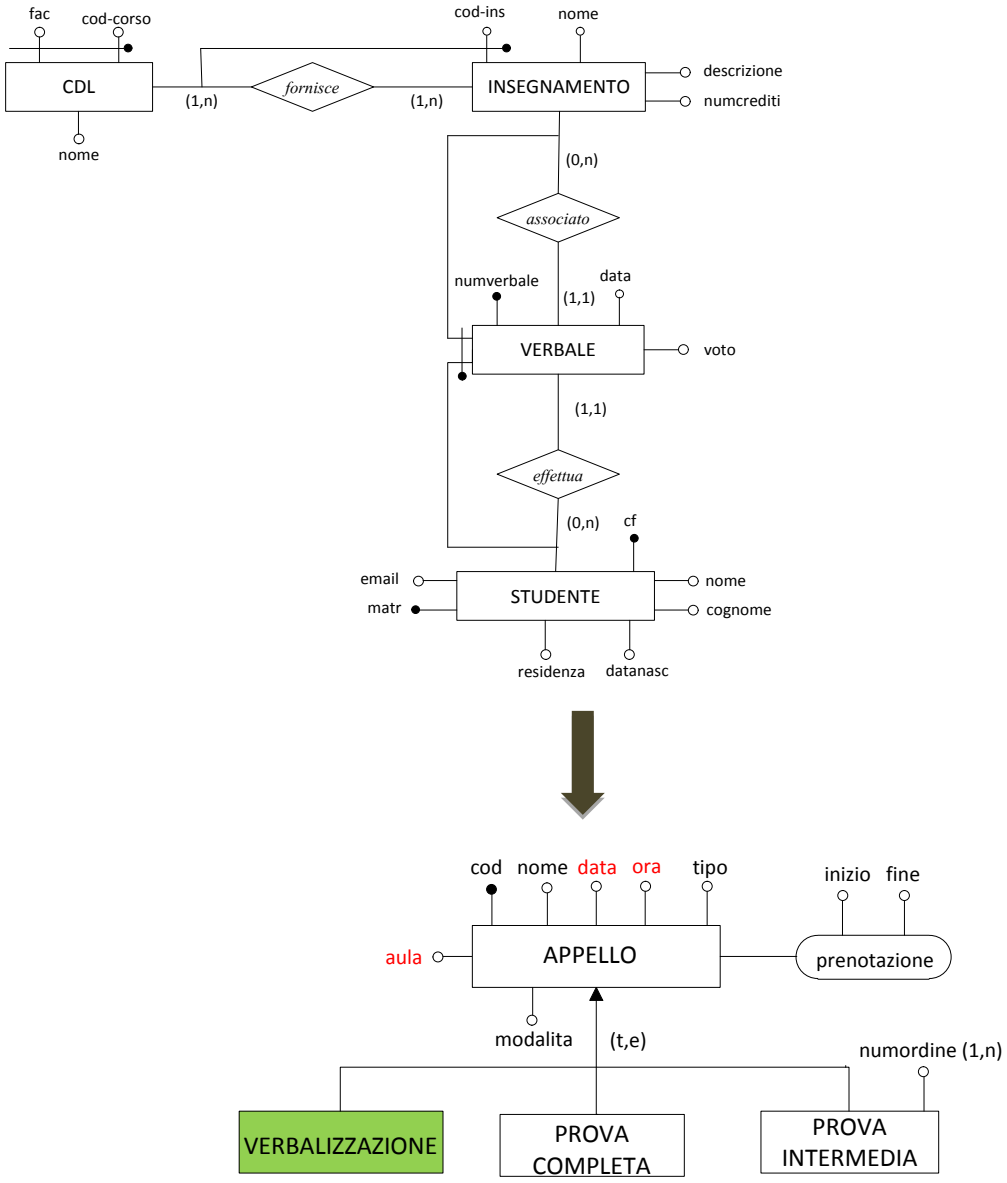


♦ Proseguendo con un'analisi dell'applicativo ESSE3 e dei file XLS che esporta , si scopre che le entità APPELLO , ESAME e VERBALE non vengono distinte dall'applicazione, ma esprimono sostanzialmente lo stesso concetto . Possiamo quindi decidere di modellare le due entità fondendole in una sola entità APPELLO. Per motivi di chiarezza andremo a modellare l'entità APPELLO con due

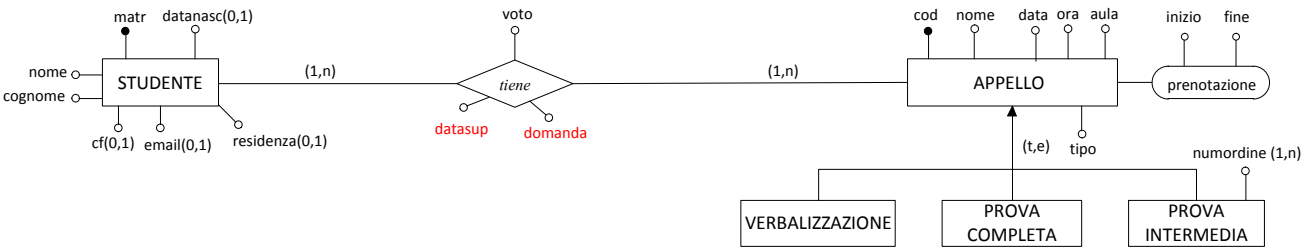
sottoentità PROVA COMPLETA e PROVA INTERMEDIA, per l'efficienza del nostro database manterremo l'attributo multiplo numordine (1,n) sulla sottoentità PROVA INTERMEDIA per fare variare le rispettive prove intermedie.



Proseguendo con un'analisi di questi file, si scopre che l'entità VERBALE esprime sostanzialmente lo stesso concetto con le entità PROVA COMPLETA e PROVA INTERMEDIA. Possiamo decidere di modellarlo come sottoentità dell'entità APPELLO.

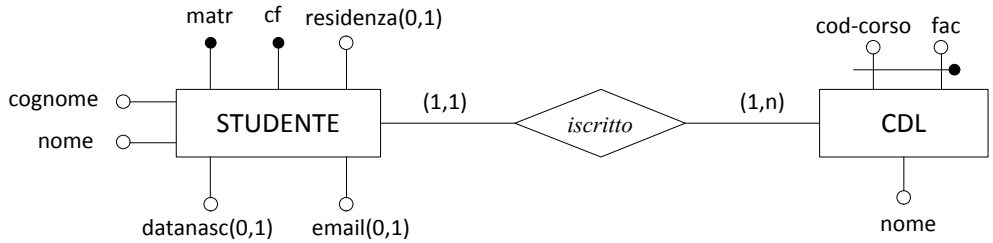


♦ Dall'analisi del file relativo alla verbalizzazione si evince che per ogni studente che tiene un appello ed eventualmente lo verbalizza si memorizza la data di superamento (datasup) e la domanda posta (domanda) durante la prova.

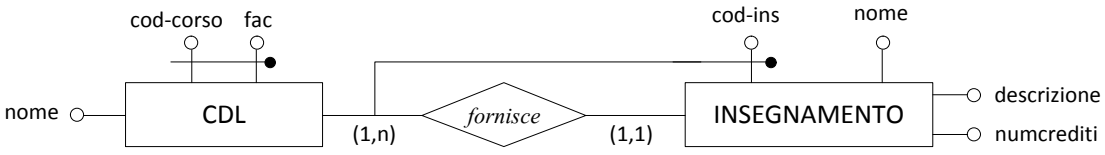


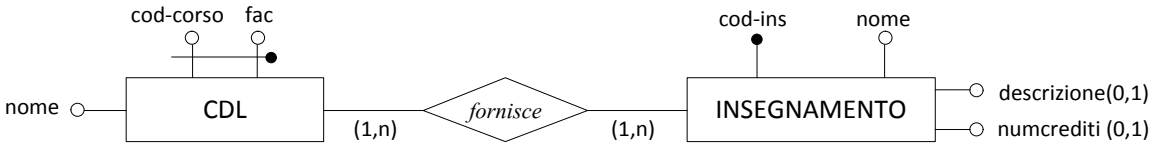
♦ Proseguendo con l'analisi del file XLS dell'applicazione relativa alla verbalizzazione si scopre che si possono rilassare alcuni vincoli come:

• Eliminazione dell'associazione tra l'entità CDL e STUDENTE visto che il concetto non è chiaramente espressa nel file.

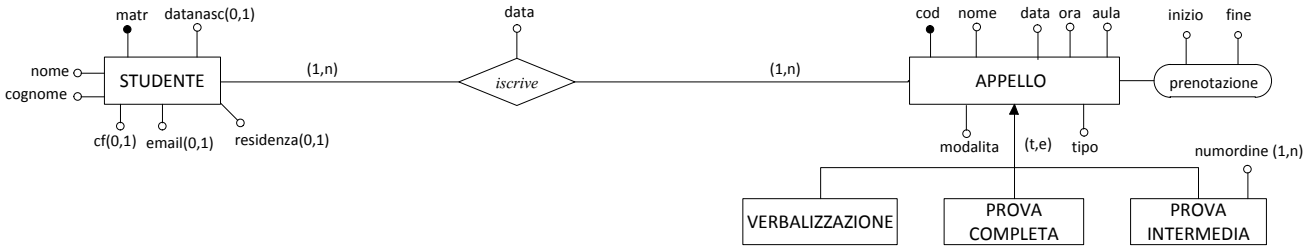


• Siccome in ESSE3 un insegnamento può fare riferimento a più CDL (corso di laurea) e questo entra in conflitto con l'identificatore e la card di partecipazione card (Insegnamento, Fornisce) = (1,1), andiamo a modellare questo concetto.

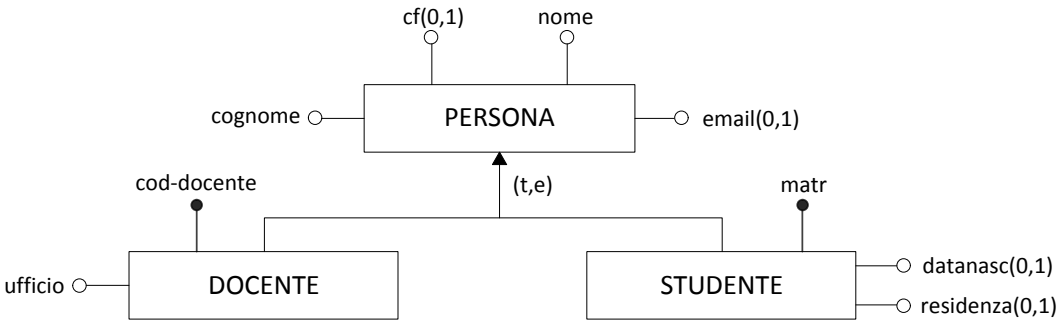


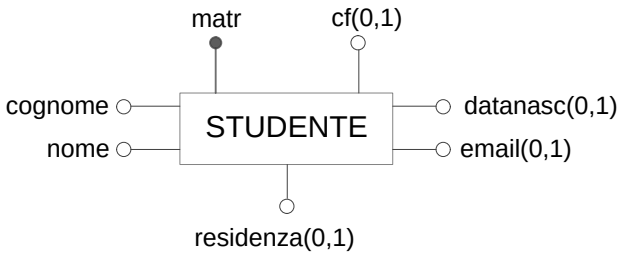


♦ Possiamo decidere di eliminare l'associazione (*Iscrive*) tra l'entità STUDENTE e APPELLO visto che il concetto non è espresso nel file.



♦ Dall'analisi del file XLS dell'applicazione si scopre che non abbiamo nessuna informazione sul docente di un insegnamento, per motivi di semplicità possiamo decidere di rimuovere l'entità DOCENTE dal nostro schema. Di conseguenza si perde la generalizzazione





3.3 Schema E/R finale

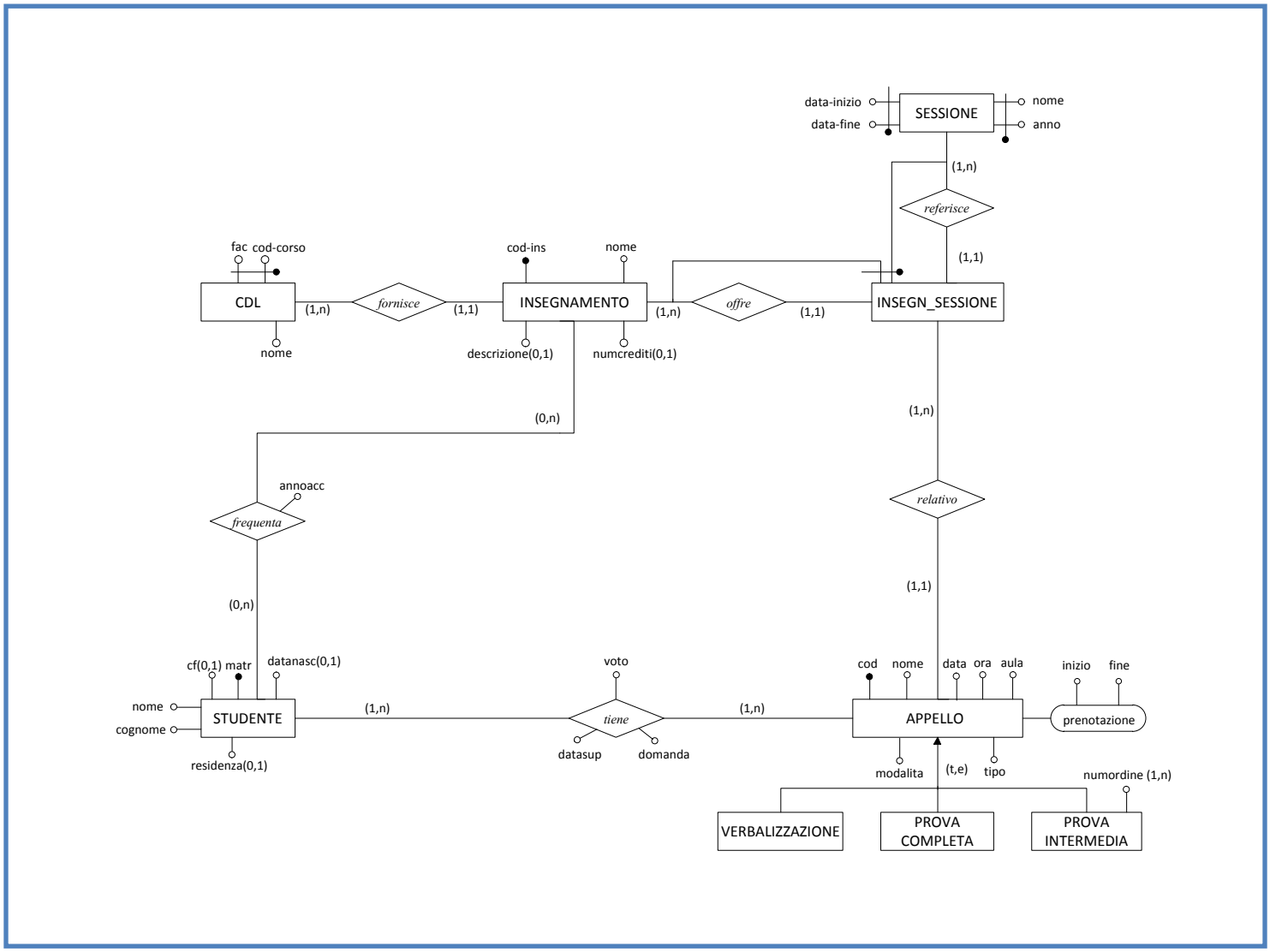


Figura 2: Schema E/R finale con modifiche portate da Esse3

PROGETTAZIONE LOGICA RELAZIONALE

Con il termine *progettazione logica* si intende la traduzione di uno schema definito tramite un modello concettuale in uno schema definito da modello logico. La fase di progettazione logica, nel nostro caso, opera su uno schema concettuale realizzato tramite il modello E/R. Tale fase è indispensabile in quanto non esistono DBMS in grado di operare direttamente sugli oggetti di uno schema E/R. La trasformazione vedrà la produzione di uno schema logico relazionale.

♦ La prima fase consiste in una semplificazione dello schema E/R (eliminazione di gerarchie e identificazioni esterne, normalizzazione di attributi composti o multipli, scelta di chiavi primarie) basata su criteri di ottimizzazione dello schema. Il risultato è ancora uno schema E/R , quindi , questa fase risulta indipendente dal modello logico scelto per l'implementazione della base di dati.

Completata la ristrutturazione dello schema concettuale, è ora possibile effettuarne la traduzione verso lo schema relazionale.

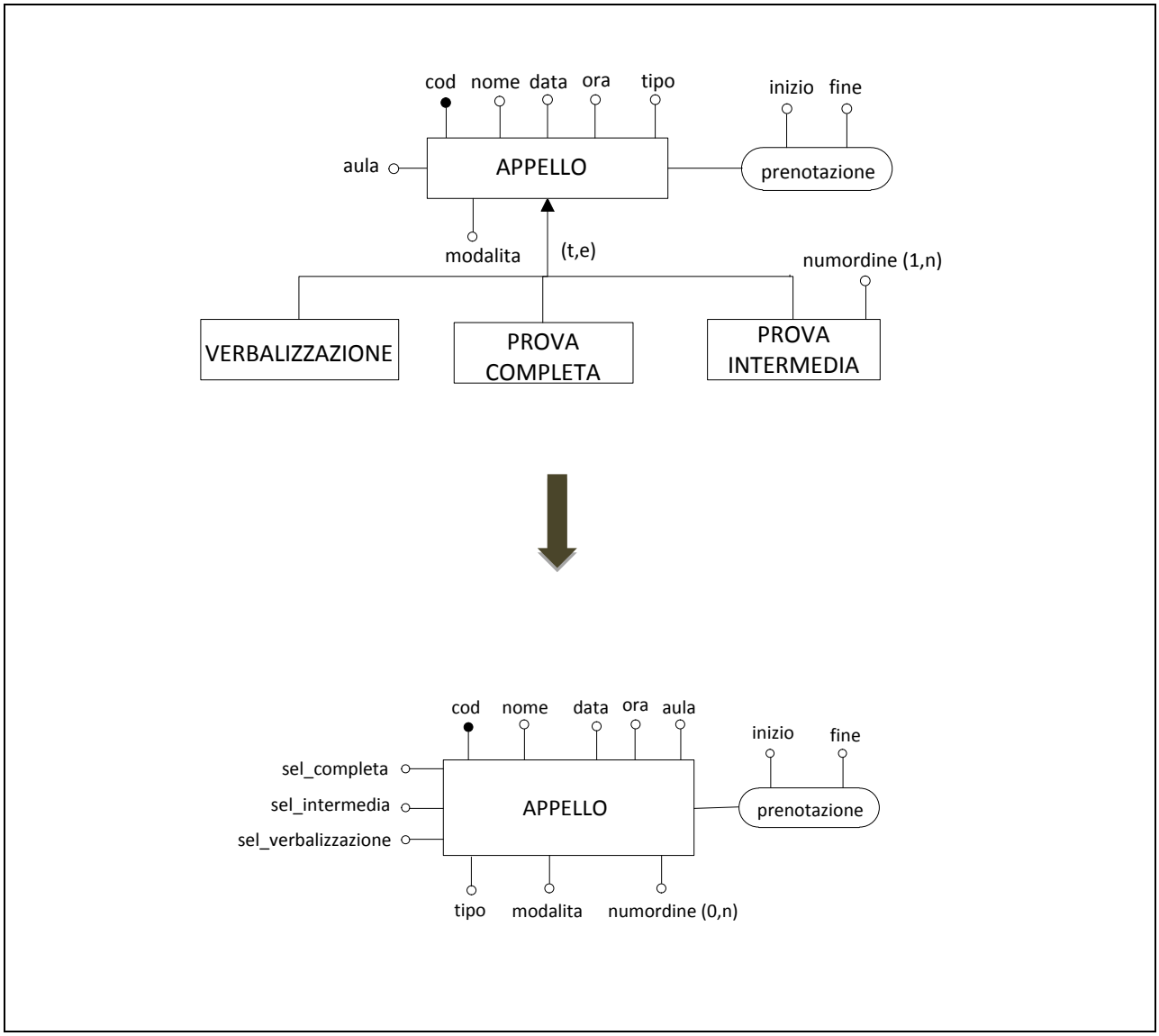
♦ La seconda fase è riferita ad un particolare modello logico , il modello relazionale , e consiste nella vera propria trasformazione dello schema E/R semplificato in uno schema relazionale.

4.1 Eliminazione delle gerarchie

In questa attività si ottiene uno schema E/R semplificato in cui ogni gerarchia è sostituita da appropriate entità ed associazioni.

4.1.1 Collasso verso l'alto

Il collasso verso l'alto riunisce tutte le entità figlie nell'entità padre. Con questa metologia possiamo modellare l'entità APPELLO.



4.2 Eliminazione degli attributi composti

Per eliminare un attributo composto A da un'entità E si può procedere in 2 modi:

♦ Eliminazione dei sotto-attributi di A

• L'attributo composto diventa un attributo semplice.

• Il compito dell'applicazione è di garantire che il nuovo attributo contenga valori coerenti con la semantica dell'attributo composto ristrutturato.

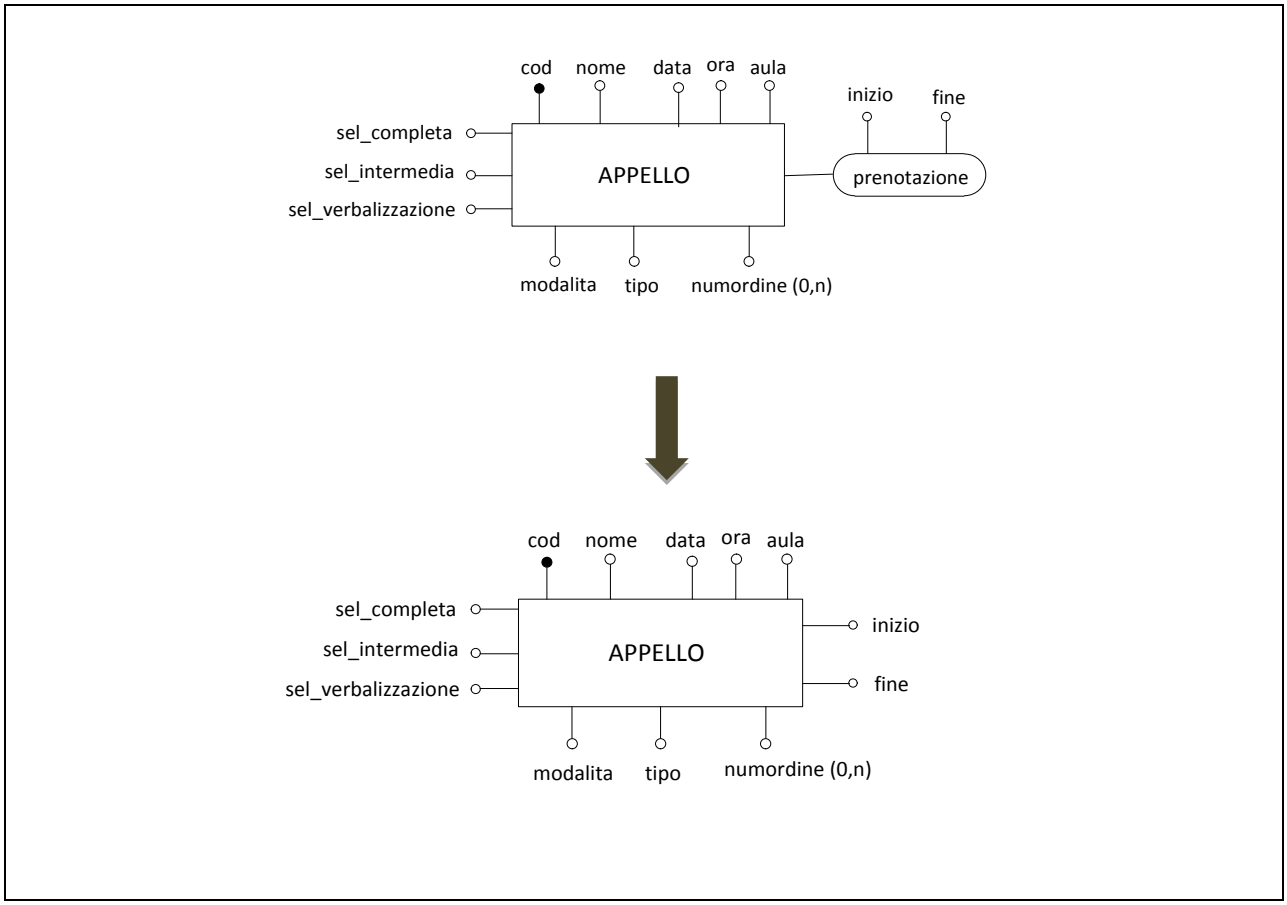
♦ Considerare tutti i sotto-attributi di A come attributi di E

• Ridefinizione del dominio dell'attributo.

• Si perde la relazione tra i sotto-attributi.

• Eventuali vincoli di cardinalità esistenti per l'attributo composto vengono associati a ciascuno dei nuovi attributi generati tramite la ristrutturazione.

• Se le componenti dell'attributo composto sono a loro volta attributi composti, si ri-applica la procedura.



4.3 Schema E/R senza gerarchia e l'attributi composto

Dopo tutto questi passaggi avendo lo scopo di semplificare nostro schema E/R, in modo da rendere più facile la traduzione nello schema relazionale , abbiamo il seguente schema E/R senza gerarchie con l'accorpamento di alcune entità.

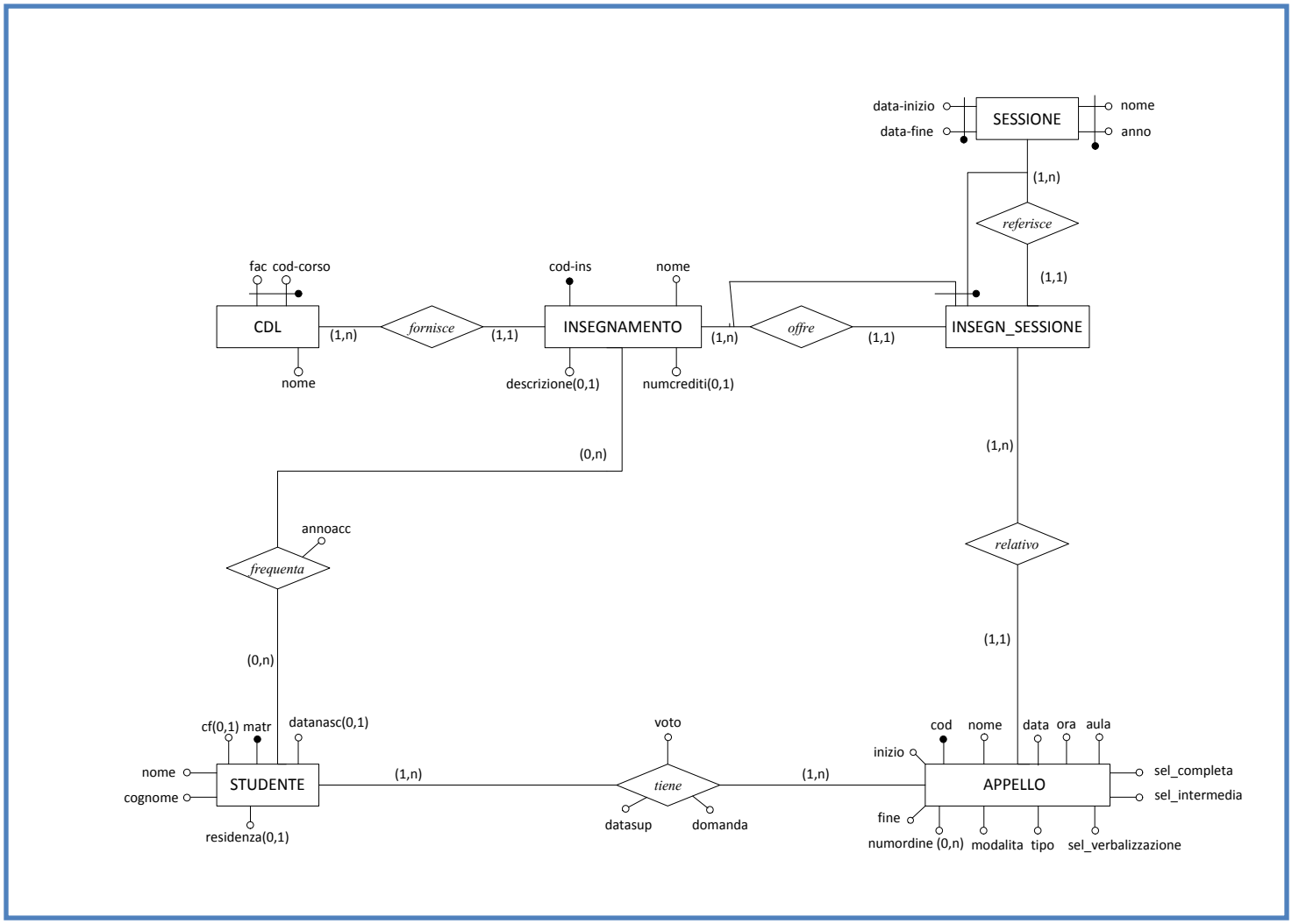


Figura 3: Schema E/R senza gerarchia e attributi composti

4.4 Schema relazionale

4.4.1 Traduzione di entità e associazioni

Nello schema ristrutturato sono presenti solo associazioni binarie dei seguenti tipi:

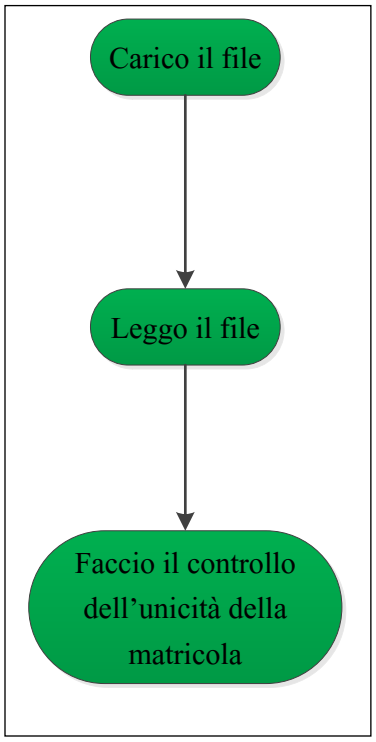
♦ (-,N) : (-,N) ⇒ si applica la traduzione standard

♦ (1,1) : (-,N) ⇒ traduzione compatta a due relazioni

4.4.2 Traduzione dallo schema E/R allo schema relazionale

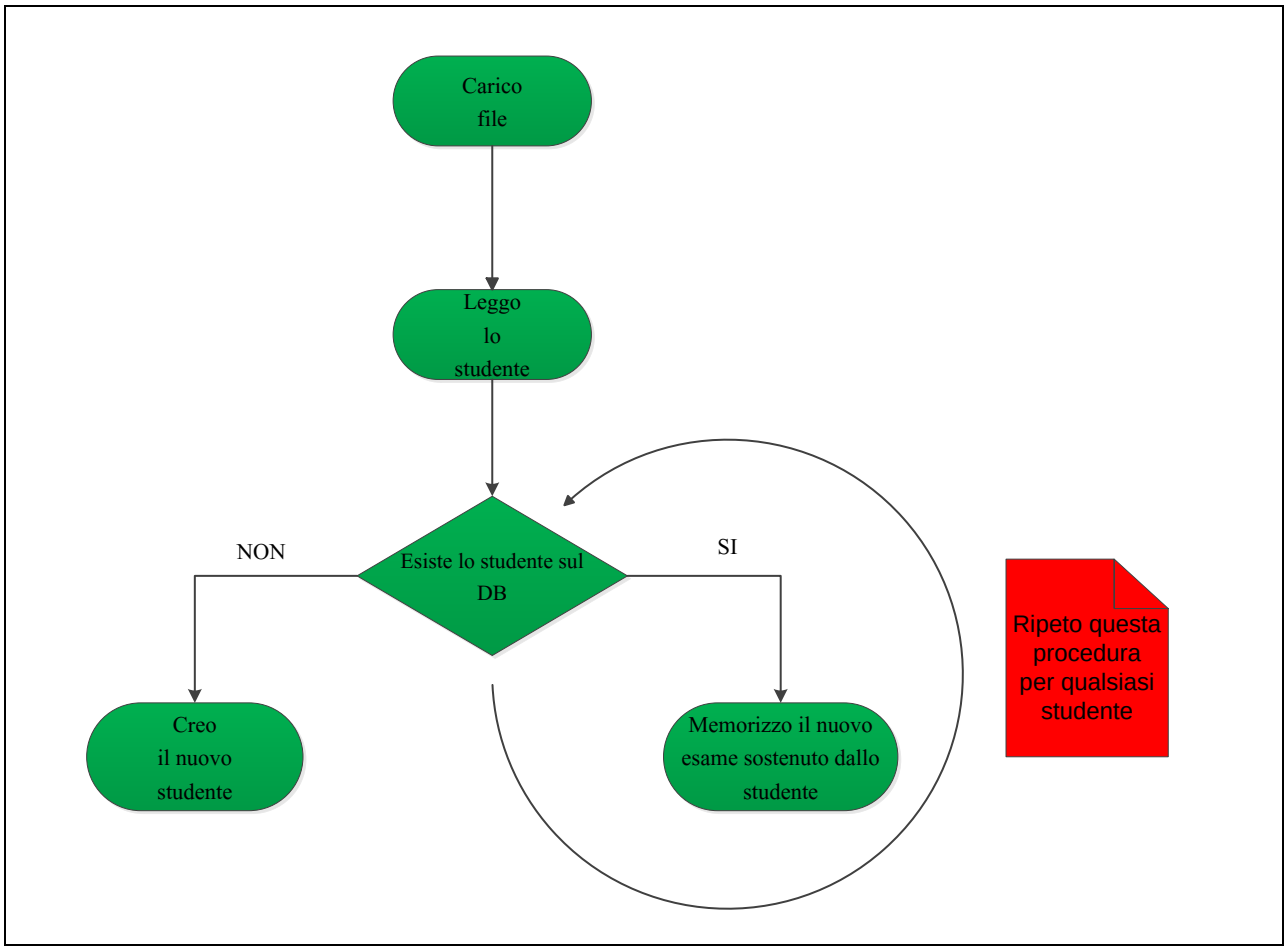
CDL (FAC, CODCORSO, NOME)
INSEGNAMENTO (CODINS, NOME, DESCRIZIONE, NUMCREDITI, FAC, CODCORSO)
FK: FAC, CODCORSO **REFERENCES** CDL NOT NULL
SESSIONE (DATAINIZIO, DATAFINE, NOME, ANNO)
AK: NOME, ANNO
INSEGN_SESSIONE (CODINS, DATAINIZIO, DATAFINE)
FK: CODINS **REFERENCES** INSEGNAMENTO
FK: DATAINIZIO, DATAFINE **REFERENCES** SESSIONE
STUDENTE (MATR, CF, NOME, COGNOME, DATANASC, RESIDENZA, EMAIL)
FREQUENTA (CODINS, MATR, ANNOACC)
FK: CODINS **REFERENCES** INSEGNAMENTO
FK: MATR **REFERENCES** STUDENTE
APPELLO (COD, DATAINIZIO, DATAFINE, CODINS, NOME, DATA, ORA, AULA, TIPO,
MODALITA, COMPLETA, INTERMEDIA,VERBALIZZAZIONE,INIZIO,FINE)
FK: CODINS, DATAINIZIO, DATAFINE **REFERENCES** INSEGN_SESSIONE NOT NULL
NUMORDINE (COD, ORDINE)
FK: COD **REFERENCES** APPELLO
TIENE (MATR, COD, VOTO, DOMANDA, DATASUP)
FK: MATR **REFERENCES** STUDENTE
FK: COD **REFERENCES** APPELLO

4.4.3 Specifiche dell'applicazione software e script di generazione della base di dati



Dopo il caricamento di un file excel (xls), l'applicazione software inizia il processo di lettura del file andando a controllare l'univocità della matricola dello studente poiché è l'identificativo dello studente ad l'interno di una facoltà, questo è possibile mediante i vincoli imposti durante la generazione dello script della base di dati.

Alla carica del secondo file, se lo studente esiste già nella base di dati, l'applicazione mantiene i suoi dati, è si procede con la memorizzazione delle informazione relative al nuovo esame sostenuto.



Un database di SQL Server è costituito da un set di tabelle in cui è archiviato un set specifico di dati strutturati. Una tabella contiene una raccolta di righe, definite anche record o tuple, e colonne, definite anche attributi.

• Ogni colonna nella tabella è progettata per contenere un tipo di informazioni specifico, ad esempio date, nomi, importi in valuta e numeri.

• Alle tabelle sono associati numerosi tipi di controlli, ad esempio vincoli, trigger, valori predefiniti e tipi di dati utente personalizzati, che garantiscono la validità dei dati.

• È possibile aggiungere vincoli di integrità referenziale dichiarativa (DRI, *Declarative Referential Integrity*) alle tabelle per garantire che i dati correlati delle diverse tabelle rimangano consistenti.

Dallo schema relazionale si realizza lo script di creazione della base di dati, mediante la definizione di opportune opzioni, e la creazione di eventuale tabelle.

```
-- INIZIO PRELIMINARI
SET NOCOUNT ON
SET DATEFORMAT dmy
USE master
GO

IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = 'db')
DROP DATABASE [db]
GO

CREATE DATABASE [db]
GO

USE [db]
GO
```

```
-- Creazione della tabella Corso di laurea(CDL)

CREATE TABLE [dbo].[CDL] (
    [fac] [nvarchar](100) NOT NULL,
    [codcorso][nvarchar](100) NOT NULL,
    [nome] [nvarchar](100) NOT NULL,
    PRIMARY KEY ([fac],[codcorso])  )
GO
```

```
-- Creazione della tabella Insegnamento

CREATE TABLE [dbo].[INSEGNAMENTO] (
    [codins][nvarchar](100) NOT NULL PRIMARY KEY,
    [nome] [nvarchar](100) NOT NULL,
    [descrizione][nvarchar](100),
    [numcrediti][int] NULL,
    [fac] [nvarchar](100) NOT NULL,
    [codcorso][nvarchar](100) NOT NULL,
    FOREIGN KEY ([fac], [codcorso]) REFERENCES [CDL] )
GO
```

```
-- Creazione della tabella Sessione

CREATE TABLE [dbo].[SESSIONE] (
    [datainizio][date] NOT NULL,
    [datafine][date] NOT NULL,
    PRIMARY KEY ([datainizio],[datafine]),
    [nome][nvarchar](100) NOT NULL,
    [anno][nvarchar](100) NOT NULL    )
GO
```

```
-- Creazione della tabella Insegn_sessione
```

```
CREATE TABLE [dbo].[INSEGN_SESSIONE] (
    [codins][nvarchar](100) NOT NULL FOREIGN KEY REFERENCES [INSEGNAMENTO],
    [datainizio][date] NOT NULL,
    [datafine][date] NOT NULL,
    PRIMARY KEY ([codins],[datainizio],[datafine]),
    FOREIGN KEY ([datainizio],[datafine]) REFERENCES [SESSIONE] )
GO
```

```
-- Creazione della tabella Studente
```

```
CREATE TABLE [dbo].[STUDENTE] (
    [matricola] [int] NOT NULL PRIMARY KEY,
    [cf][nvarchar](100) NULL,
    [nome][nvarchar](100) NOT NULL,
    [cognome] [nvarchar](100) NOT NULL,
    [datanasc][date] NULL,
    [residenza][nvarchar](100) NULL,
    [email][nvarchar](100) NULL )
GO
```

```
-- Creazione della tabella Frequentare
```

```
CREATE TABLE [dbo].[FREQUENTA] (
    [codins][nvarchar](100) NOT NULL FOREIGN KEY REFERENCES[INSEGNAMENTO],
    [matricola][int] NOT NULL FOREIGN KEY REFERENCES [STUDENTE],
    [annoacc][nvarchar](100) NOT NULL,
    PRIMARY KEY ([codins],[matricola])  )
 GO
```

```
-- Creazione della tabella Appello
```

```
CREATE TABLE [dbo].[APPELLO] (
    [cod][int] PRIMARY KEY IDENTITY,
    [codins][nvarchar](100) NOT NULL,
    [nome][nvarchar](100) NOT NULL,
    [data][date] NOT NULL,
    [ora][nvarchar](100) NOT NULL,
    [aula][nvarchar](100) NOT NULL,
    [inizio][date] NOT NULL,
    [fine][date] NOT NULL,
    [modalita][nvarchar](100) NOT NULL ,
    [tipo][nvarchar](100) NOT NULL,
    [completa][nvarchar](100),
    [intermedia][nvarchar](100),
    [verbalizzazione][nvarchar](100),
    [datainizio][date] NOT NULL,
   [datafine][date] NOT NULL,
FOREIGN KEY ([codins],[datainizio],[datafine]) REFERENCES [INSEGN_SESSIONE] )
GO
```

```
-- Creazione della tabella Numordine
```

```
CREATE TABLE [dbo].[NUMORDINE] (
    [cod][int] FOREIGN KEY REFERENCES [APPELLO] PRIMARY KEY IDENTITY,
    [ordine][int]  )
GO
```

```
--Creazione della tabella Tiene
```

```
CREATE TABLE [dbo].[TIENE] (
    [matricola][int] NOT NULL FOREIGN KEY REFERENCES [STUDENTE],
    [nome][nvarchar](100),
    [data][date] NOT NULL ,
    [voto][int],
    [ordine][int],
    [domanda][nvarchar](100) NULL,
    [datasup][date] NULL,
    PRIMARY KEY ([matricola],[nome],[data]) )
GO
```

• Struttura della base di dati

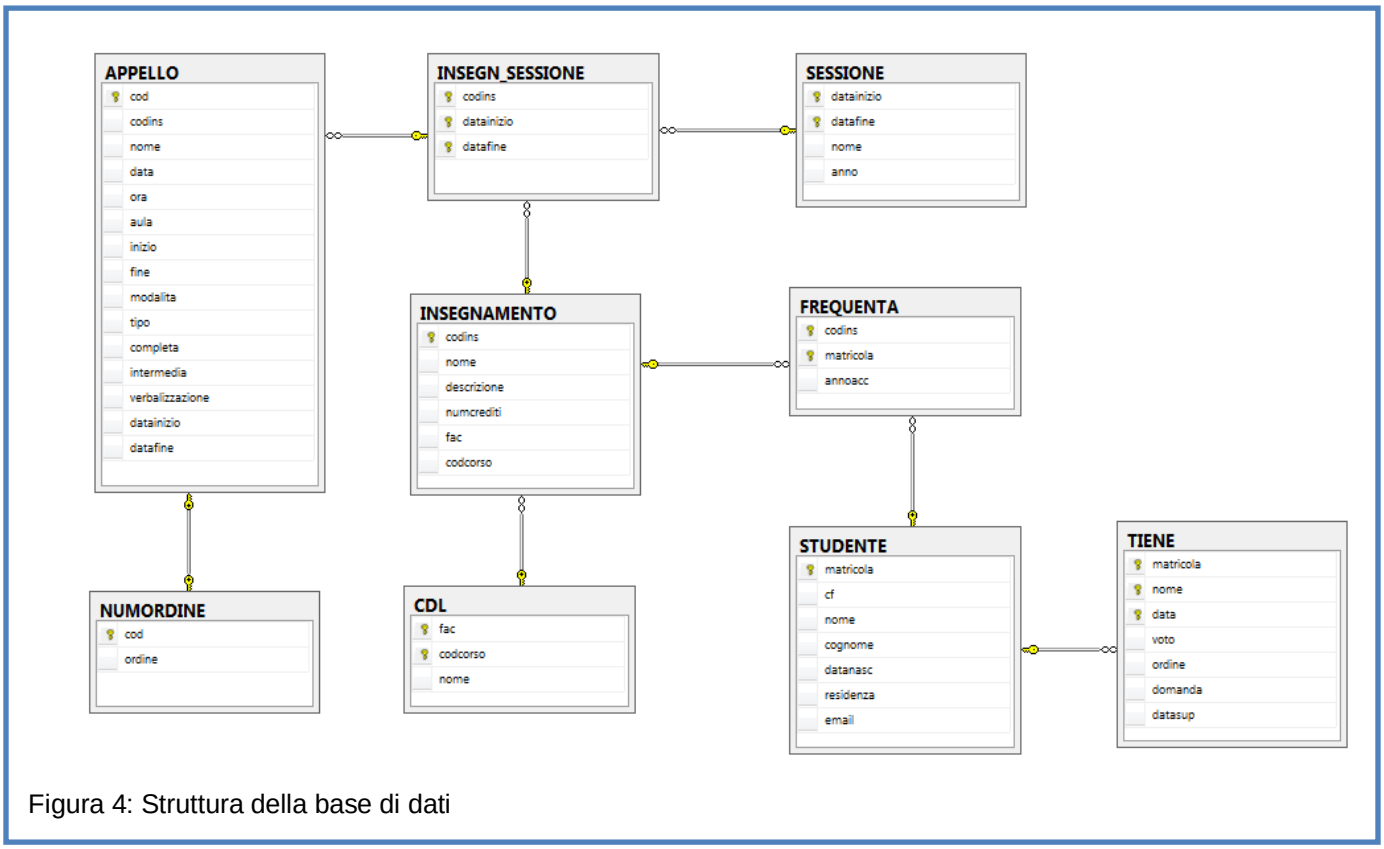
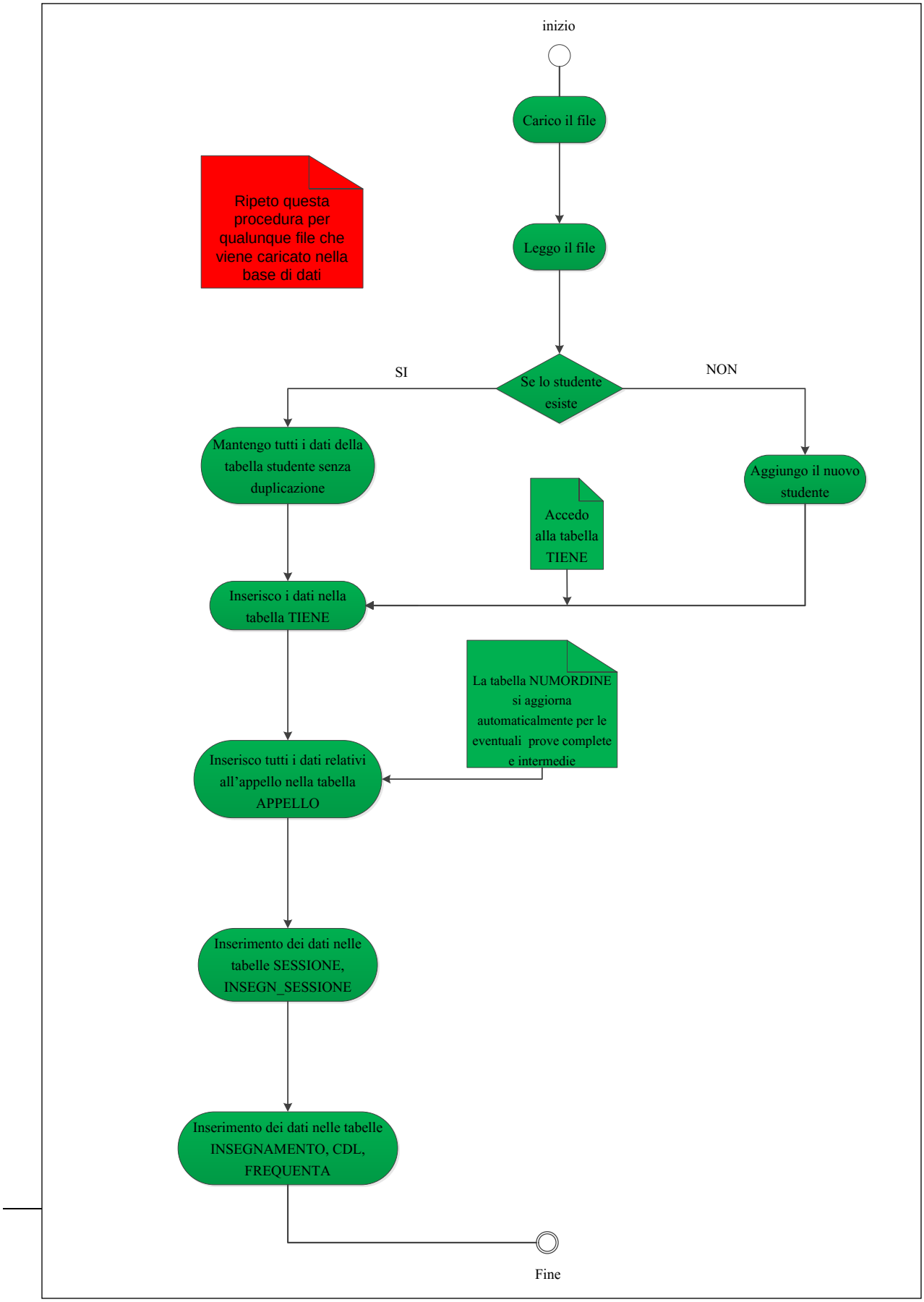


Figura 4: Struttura della base di dati


inizio
Carico il file
Ripeto questa procedura per qualunque file che viene caricato nella base di dati
Leggo il file
Se lo studente esiste
SI
NON
Mantengo tutti i dati della tabella studente senza duplicazione
Aggiungo il nuovo studente
Accedo alla tabella TIENE
Inserisco i dati nella tabella TIENE
La tabella NUMORDINE si aggiorna automaticalmente per le eventuali prove complete e intermedie
Inserisco tutti i dati relativi all'appello nella tabella APPELLO
Inserimento dei dati nelle tabelle SESSIONE, INSEGN_SESSIONE
Inserimento dei dati nelle tabelle INSEGNAMENTO, CDL, FREQUENTA
Fine

INSERIMENTO DEI DATI NELLA BASE DI DATI

Per integrazione s'intende l'interazione funzionale di uomini, processi e tecnologie con l'obiettivo di creare un insieme unitario. Come presupposti per l'integrazione abbiamo:

Acquisizione dei dati: Devono essere garantite modalità automatiche, rapide ed efficienti per raccogliere i dati dalle molteplici fonti .

Trasmissione dei dati : E' necessaria un'infrastruttura che garantisca compatibilità tra i diversi sistemi connessi e continuità ed affidabilità nella trasmissione delle informazioni.

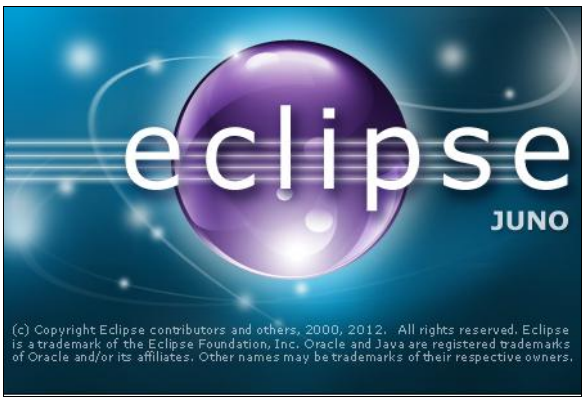
Memorizzazione: Non può essere realizzata a livello di singola postazione di lavoro o nell'ambito di una singola area funzionale, è indipendente dall'ubicazione fisica dei relativi archivi e dall'architettura utilizzata.

L'applicativo Esse3 (Sistema di pubblicazione degli esiti degli esami) permette di gestire l'intero processo relativo agli appelli d'esame : creazione, iscrizione, pubblicazione dei voti e verbalizzazione. I dati gestiti dall'applicativo possono essere esportati in un file excel (xls).

Siccome l'obiettivo della tesi è l'integrazione nel database esami di dati provenienti da tale applicativo, lo scopo di questa sessione sarà di trovare una procedura automatica che permetterà di caricare questi dati mediante l'uso di **Apache POI** (Libreria per la manipolazione dei documenti microsoft in java) e di una connessione **JDBC** per il trasferimento dei dati nella base di dati progettata nel ambiente **SQL SERVER** (nel nostro studio).

5.1 Strumenti necessari

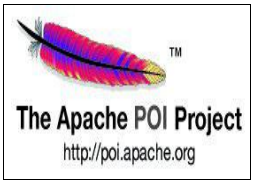
5.1.1 L'ambiente Eclipse

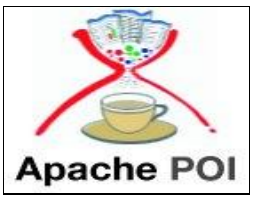


Eclipse è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma. Ideato da un consorzio di grandi società quali Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP e Serena Software, chiamato Eclipse Foundation sullo stile dell'open source.

- Può essere adattato alle esigenze più varie semplicemente aggiungendo un plug-in.
- Eclipse è scaricabile all'indirizzo *www.eclipse.org*
- Eclipse non necessita di installazione, per avviare eclipse è sufficiente eseguire *eclipse.exe*
- E' disponibile per tutte le architetture più diffuse (Windows, Linux, Mac ...)

5.1.2 Il progetto java **Apache POI**





Il progetto **Apache POI** ha come obiettivo quello di creare delle API Java per poter manipolare i diversi formati su cui sono basati i documenti Office di Microsoft, in particolare l'API è in grado di manipolare file di tipo OOXML (Office Open XML) e di tipo OLE 2 (OLE 2 Compound Document). Lo standard OLE 2 comprende i file di Office con estensione XLS, DOC e PPT mentre lo standard OOXML comprende quelli con estensione XLSX, DOCX e PPTX.

Possiamo scaricare le API come pacchetto zip o tar all'indirizzo: *http://poi.apache.org/download.html* tale archivio comprende diversi jar file che non sono tutti necessari ma dipendono dal formato e dal tipo di file che vogliamo utilizzare secondo questo schema

Component	Application type	Maven artifactId
POIFS	OLE2 Filesystem	poi
HPSF	OLE2 Property Sets	poi
HSSF	Excel XLS	poi
HSLF	PowerPoint PPT	poi-scratchpad
HWPF	Word DOC	poi-scratchpad
HDGF	Visio VSD	poi-scratchpad
HPBF	Publisher PUB	poi-scratchpad
HSMF	Outlook MSG	poi-scratchpad
XSSF	Excel XLSX	poi-ooxml
XSLF	PowerPoint PPTX	poi-ooxml
XWPF	Word DOCX	poi-ooxml
OpenXML4J	OOXML	poi-ooxml-schemas, ooxml-schemas

Inoltre questi jar hanno a loro volta dei jar come prerequisiti:

File .jar	Prerequisiti
poi-version-yyyymmdd.jar	commons-logging, log4j
poi-scratchpad-version-yyyymmdd.jar	poi-version-yyyymmdd.jar
poi-ooxml-version-yyyymmdd.jar	poi-version-yyyymmdd.jar, poi-version-yyyymmdd.jar
poi-ooxml-schemas-version-yyyymmdd.jar	xmlbeans, geronimo-stax-api_1.0_spec
ooxml-schemas-version-yyyymmdd.jar	xmlbeans, geronimo-stax-api_1.0_spec

5.1.3 IL driver **JDBC**

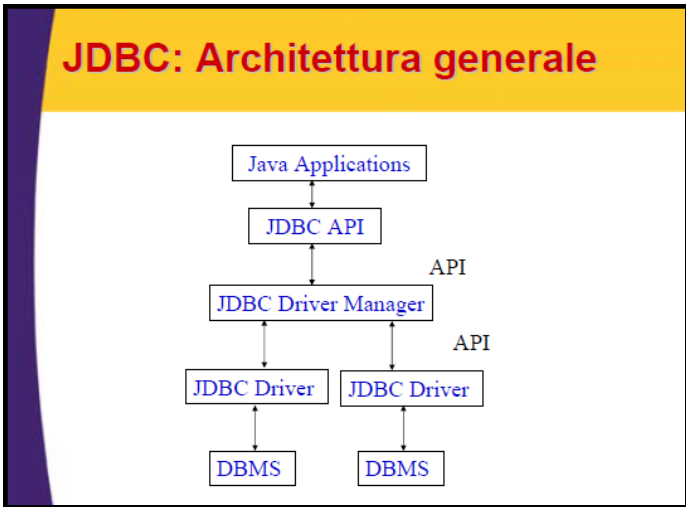
JDBC è una libreria di Java, quindi un insieme di classi, che fornisce i metodi per l'accesso e la manipolazione di basi di dati di tipo relazionale.

Mantenendo la filosofia di Java, le API JDBC sono state progettate per ottenere l'accesso ai dati indipendentemente dalla piattaforma e dal particolare tipo di database utilizzato; **JDBC** offre infatti delle interfacce standard e quindi uniche per qualunque DB; per ciascuno di questi deve poi esistere un driver che si occupi di tradurre le chiamate **JDBC** effettuate dall'applicazione in opportune

istruzioni per accedere e manipolare i dati di uno specifico database.

Quando l'applicazione richiede l'accesso ad un DB un particolare componente detto Driver Manager controlla il tipo di DB per cui si è richiesta la connessione e carica il driver appropriato.

In questo modo è possibile cambiare sia la piattaforma che il DB semplicemente cambiando driver, sempre che ne esista uno.



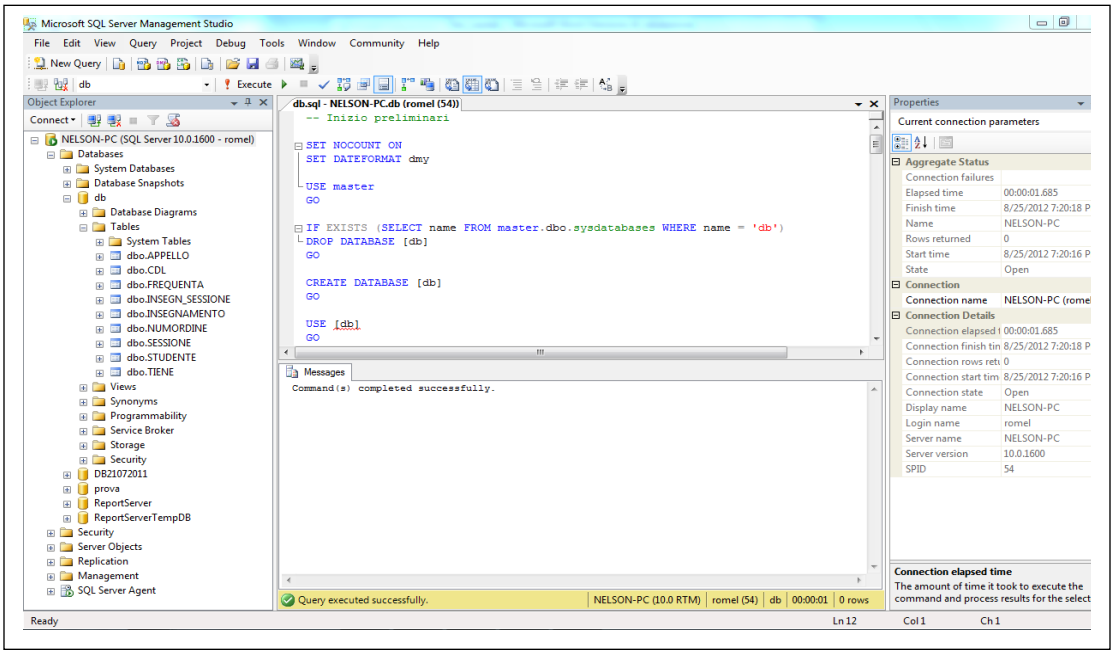
Un driver per essere considerato pienamente compatibile con **JDBC** deve implementare tutte le interfacce previste dalle specifiche di **JDBC**. Se questa condizione è soddisfatta il driver si dice JDBC-Compliant. E' scaricabile dal sito: http://java.sun.com/products/jdbc

5.2 Procedura per l'integrazione dei dati

La procedura di integrazione dei dati nella basi di dati è suddivisa in 3 fasi distinte:

5.2.1 Esecuzione dello script sul SQL SERVER

In questa fase l'utente esegue lo script di creazione della base di dati su *SQL SERVER*



5.2.2 Scrittura di un programma di appoggio in java

In questa fase si crea un'applicazione software in java dotata di un'interfaccia grafica che permette all'utente di configurare i parametri della connessione e di gestire il caricamento dei file

Excel(xls). Concettualmente questa applicazione ha il compito di stabilire un collegamento tra i dati estratti dai file Excel(xls) anziché i loro inserimento nella base di dati mediante una connessione JDBC mediante l'uso di un apposito driver JDBC (relativo alla base di dati es: JDBC per *Oracle*, JDBC per *Mysql*, JDBC per *SQL SERVER*).

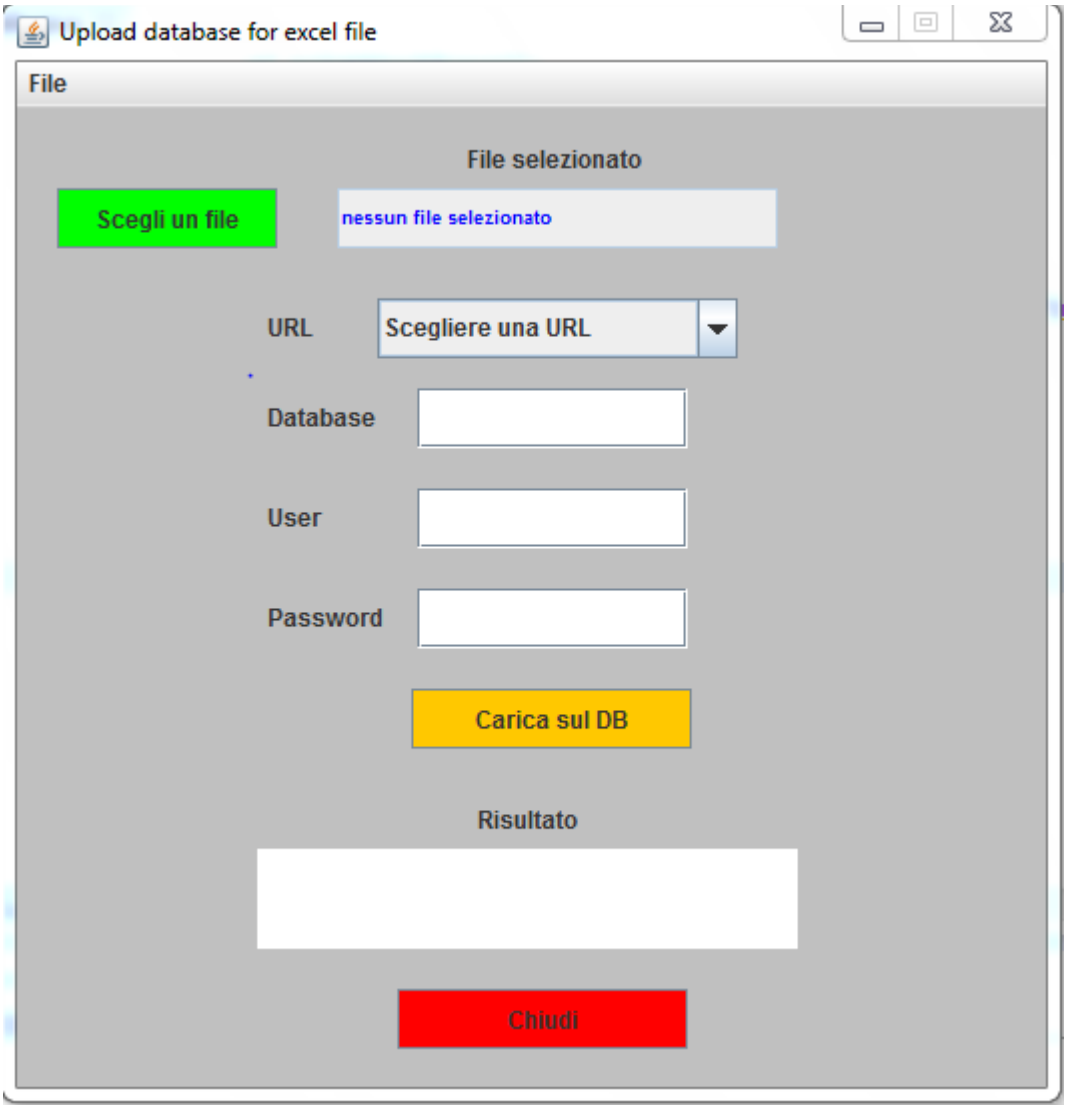


Figura 5: Applicazione software

5.2.3 specifiche dell'applicazione software e uso di una connessione JDBC

L'applicazione in java è costituito da 9 classi principali, che hanno il compito di assicurare la corretta lettura del file excel e l'inserimento dei suoi dati nella base di dati in SQL SERVER mediante l'uso di una connessione **JDBC**. (*Per tutta la documentazione sulle classi consultare la javadoc)*

La classe Pannello

E' la classe che gestisce l'interfaccia grafica del software con tutte le specifiche per semplificare l'uso del software da parte dell'utente, semplificando l'interfaccia grafica.

La classe ReadAppello

E' la classe che si occupa della corretta lettura delle informazioni sull'appello svolto cioè su:

- Il codice dell'insegnamento (*codins*)
- Il nome dell'insegnamento *(nome)*
- La data (data) e l'ora dell'appello *(ora)*
- La scala di valutazione usata *(modalita)*
- La data di inizio *(inizio)* e di fine*(fine)* della prenotazione dell'appello
- Il tipo di prova *(tipo)* anziché il tipo d'appello *(completa, intermedia)*

La classe ReadFile

E' la classe che si occupa della corretta lettura delle informazioni sull'insegnamento svolto, la sessione in corso e il corso di laurea coinvolto cioè su:

- Il nome dell' insegnamento *(nome)* e il suo codice *(codins)*
- Il nome della sessione *(nome),* la data di inizio *(datainizio)* e di fine *(datafine)* della sessione
- Il nome del corso di laurea *(nome),* il suo codice *(codcorso)* e la facoltà di afferenza *(fac)*

La classe ReadFrequenta

E' la classe che si occupa della corretta lettura delle informazioni sugli insegnamenti frequentati dallo studente, cioè su:

- Il codice dell'insegnamento *(codins)*
- La matricola dello studente *(matricola)*

La classe ReadNumOrdine

E' la classe che si occupa di capire una prova totale oppure intermedia mediante l'uso di un attributo*(ordine).*

La classe ReadStudent

E' la classe che si occupa della lettura delle informazioni sullo studente cioè su:

- La sua matricola *(matricola),* il suo nome *(nome),* e il suo cognome *(cognome).*

La classe ReadTiene

E' la classe che si occupa della gestione delle informazioni, quando lo studente dà un esame cioè su:

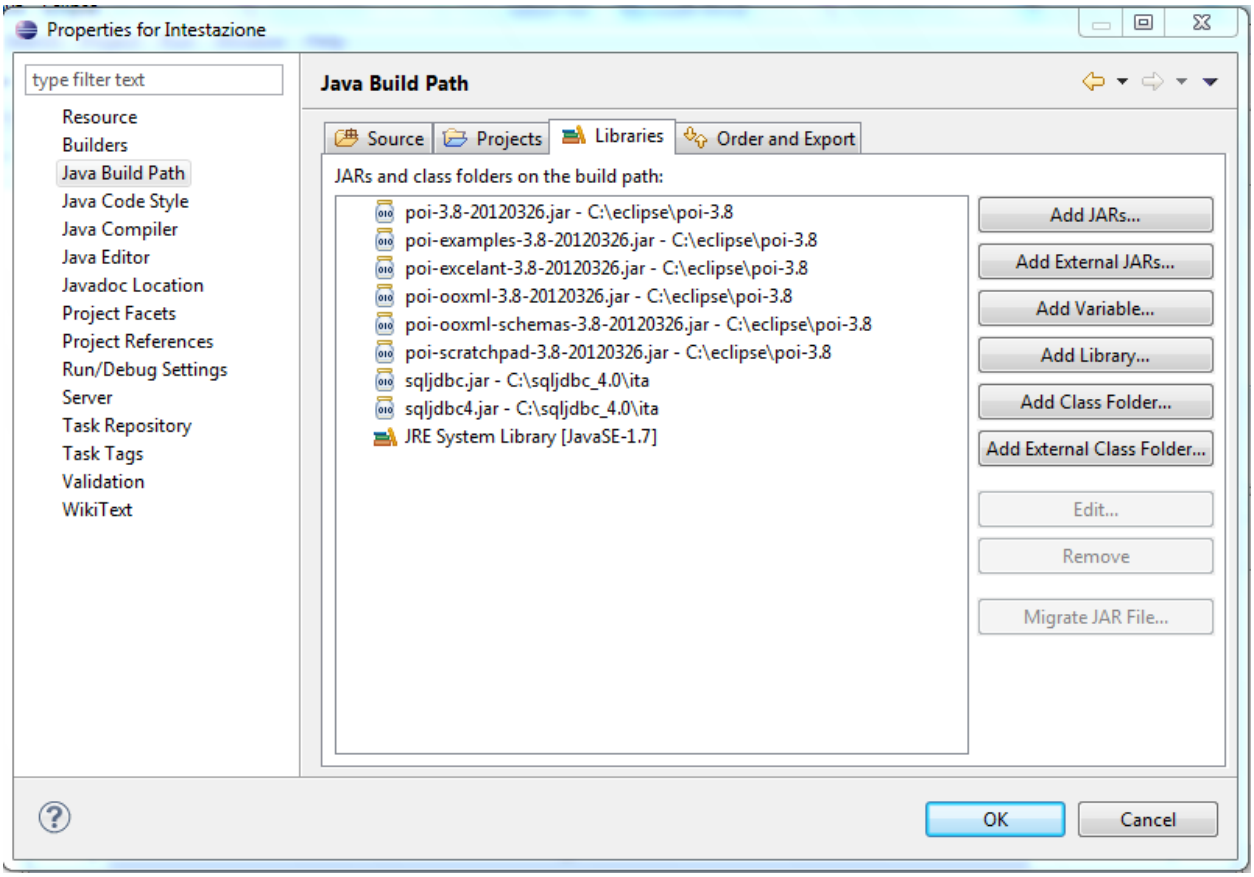
- Il nome dell'esame *(nome)*
- La data in cui lo studente ha svolto l'esame (data) e il suo rispettivo voto *(voto).*

La classe TestRead

E' la classe main del programma, che cordona il processo di lettura nelle diverse classi del programma

La classe UploadDB

Dopo aver scritto il programma in java dell'applicazione e si integra le librerie per la gestione dei file excel in java mediante la scompattazione del pacchetto Zip di **Apache POI,** poi si prosegue con l'uso delle differenti funzione di gestione di una connessione **JDBC**.



Le classi e le interfacce JDBC si trovano nel package java.sql, contenuto nella distribuzione standard del JDK (Java Development Kit).

Le classi e le interfacce più importanti del package java.sql sono:

Il package *java.sql*

- DriverManager
- Connection
- Statement
- PreparedStatement
- CallableStatement
- ResultSet
- DatabaseMetaData
- ResultSetMetaData
- Types

• La classe *DriverManager* gestisce i driver **JDBC** e seleziona il driver apposito per il database in uso.
• L'interfaccia *Connection* rappresenta una sessione di lavoro sul database e permette di ottenere informazioni sulla base di dati. Le operazioni sul database avvengono all'interno di una sessione.
• Le interfacce *Statement*, *PreparedStatement* e *CallableStatement* permettono di eseguire interrogazioni e aggiornamenti sulla base di dati.
•L'interfaccia *ResultSet* fornisce l'accesso alle tabelle.
• *DatabaseMetaData* e `ResultSetMetaData` consentono di ottenere informazioni sullo schema del database.
• La classe *Types* definisce tutta una serie di costanti che identificano i tipi di dati **SQL**.

Con queste classi e interfacce è possibile avere un controllo presso che totale della base di dati.

• Connessione ad un Database

La prima cosa da fare per iniziare a lavorare sul database è caricare il driver **JDBC** adatto. Per farlo è necessario forzare il caricamento della classe che rappresenta il driver utilizzando il metodo *forName* della classe `Class`.

Supponendo di voler utilizzare il driver **JDBC,** il caricamento si farà tramite l'istruzione:

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

A questo punto si stabilisce la connessione fra l'applicazione Java ed il database chiamando il metodo getConnection della classe *DriverManager* nel modo seguente:

```
Connection con = DriverManager.getConnection("jdbc:sqlserver://localhost; database = db_name; user = username; password = user_password");
```

• Estrazione e modifica dei dati del Database

Per eseguire delle interrogazioni o modifiche sulla base di dati si crea per prima cosa un oggetto

di tipo *Statement* utilizzando il metodo *createStatement* dell'oggetto di tipo *Connection* ottenuto in precedenza.

```
Statement st = con.CreateStatement();
```

L'interfaccia Statement fornisce una serie di metodi come *executeQuery* ed *executeUpdate* per eseguire rispettivamente delle operazioni di lettura o modifica sulla base di dati tramite istruzioni **SQL**. Il metodo *executeQuery* verrà utilizzato per eseguire delle istruzioni **SELECT** mentre il metodo executeUpdate per l'esecuzione di istruzioni **INSERT**, **UPDATE** e **DELETE**.

Il parametro richiesto dal metodo *executeQuery* è una stringa contenente il codice **SQL** da eseguire. Questo metodo non fa altro che passare le istruzioni **SQL** al driver **JDBC** e successivamente ricevere da esso il risultato della query salvandolo in un oggetto di tipo *ResultSet*.
Se invece si vogliono inserire dei dati in una tabella si utilizzerà un codice simile al seguente:

```
st.executeUpdate("INSERT into Tabella1 (campo1, campo2) values (1,2)");
```

Il metodo *executeUpdate* non ritorna un oggetto ResultSet, le istruzioni **INSERT**, **UPDATE** e **DELETE** non danno infatti come risultato una tabella. Il valore ritornato è invece un intero che indica il numero di record modificati.

Oltre all'interfaccia *Statement* esistono anche le interfacce *PreparedStatement* e *CallableStatement*.

L'utilizzo di *PreparedStatement* è conveniente nel caso di query eseguite ripetutamente, in quanto consente di specificare parzialmente delle query e richiamarle successivamente modificando solamente i nomi dei parametri. In questo modo l'esecuzione è più rapida, in quanto il *PreparedStatement* conosce già la forma dell'interrogazione e deve solo cambiarne i parametri. Inizialmente bisogna quindi creare l'interrogazione indicando con il carattere '?' i parametri il cui valore

```
PreparedStatement ps = con.prepareStatement("INSERT into Tabella (parametro1,parametro2)
values (?,?)");
```

sarà inserito.

Per inserire i valori mancanti si utilizzano i metodi *setXXX* (dove XXX è un tipo di dato) a cui vanno passati due parametri: il primo è l'indice del parametro della query che si vuole specificare, il secondo è il valore che gli si vuole dare.

• Elaborare i dati

Oltre ad eseguire delle interrogazioni sulla base di dati, molto probabilmente si vorranno anche visualizzare i risultati ottenuti. A differenza di **SQL** però, un linguaggio procedurale come Java non è pensato per operare su tabelle.
A questo scopo sono quindi presenti le interfacce *ResultSet, ResultSetMetaData e DatabaseMetaData.*

ResultSet e cursori

Un oggetto di tipo *ResultSet* fornisce l'accesso ad una tabella.
Poiché una tabella può essere composta da due o più colonne (campi) e righe (record), per accedere ai dati si utilizza un cursore.
Un cursore è simile ad un puntatore che indica un determinato record di una tabella; è possibile quindi leggere i valori dei campi del record selezionato. In ogni momento è possibile spostare il cursore sul record successivo o precedente a quello corrente oppure è possibile posizionarlo su un qualunque altro record specificandone l'indice.

Quando l'oggetto *ResultSet* viene creato il cursore punta al record "precedente al primo", quindi in realtà non punta a niente. In questo modo però è possibile scrivere un codice più elegante per scorrere tutta la tabella utilizzando un ciclo while.

• Il metodo *next* sposta il cursore sul record successivo a quello corrente e ritorna vero se il nuovo record selezionato è valido, falso se non ci sono altri record nella tabella.

• Il metodo *getString* richiede come parametro il nome di un campo della tabella e ritorna il valore di quel campo nel record corrente. Questo metodo legge il valore del campo come se fosse un campo di testo ma esiste un metodo per ogni tipo di dato (*getInt, getFloat, getDate, getObject...)*

LE VISTE

Nel nostro studio abbiamo deciso di creare delle viste, per rendere più agevole visualizzare i dati degli esami, per evitare che l'utente dovesse specificare spesso le stesse query sulla base di dati.

• Una vista è una struttura virtuale basata sulla definizione di una *SELECT* che agisce come filtro rispetto ad un insieme di tabelle di base.

• Nessun dato viene archiviato nella vista (Nel database viene archiviata in realtà solo un'istruzione *SELECT.)*

• I dati aggregati dalla vista possono essere modificati tramiteopportune condizioni

• Sintassi:

```
CREATE VIEW <nome_vista>
AS  <clausola_SELECT>
    [WITH CHECK OPTION]
```

Dove "*WITH CHECK OPTION*" implica che operazioni di modifica tramite la vista saranno impedite se comportano la scomparsa di informazioni dalla vista stessa.

• È possibile considerare una vista come una tabella virtuale o una query archiviata.

• È possibile utilizzare questa tabella virtuale facendo riferimento al nome della vista nelle istruzioni Transact-SQL, con la stessa modalità utilizzata per fare riferimento a una tabella.

• In modo analogo a una tabella, una vista è costituita da un set di colonne e righe di dati denominate.

• Le righe e le colonne di dati provengono da tabelle a cui fa riferimento la query che definisce la vista e sono prodotte dinamicamente quando si fa riferimento alla vista.

6.1 Creazione delle Viste

VA1) Vista che riporta i voto definitivo degli studenti che hanno sostenuto tutti i parziali

```
CREATE VIEW STUDESAMI
AS
SELECT S.nome, S.cognome, A1.nome as insegnamento, SUM (T1.voto+T2.voto)/2 as voto_definitivo
FROM STUDENTE S, TIENE T1,TIENE T2, APPELLO A1, APPELLO A2, NUMORDINE N1,
NUMORDINE N2
WHERE S.matricola = T1.matricola AND S.matricola = T2.matricola
AND T1.ordine = 1 AND T2.ordine = 2
AND T1.nome = A1.nome AND T2.nome = A2.nome AND T1.nome = T2.nome
AND A1.cod = N1.cod AND A2.cod = N2.cod
AND N1.ordine = 1 AND N2.ordine = 2
GROUP BY S.nome, S.cognome, A1.nome
```

VA2) Vista che riporta gli studenti che hanno superato un detto esame per corso di laurea durante una sessione.

```
CREATE VIEW STUDSUPERATO
 AS
SELECT S.matricola, S.nome, S.cognome, C.nome as corso_di_laurea, I.nome as nome_ins, T.voto,
T.data, SE.nome
FROM CDL C JOIN INSEGNAMENTO I ON (C.codcorso = I.codcorso)
JOIN FREQUENTA F ON (F.codins = I.codins)
JOIN STUDENTE S ON (S.matricola = F.matricola)
JOIN TIENE T ON (S.matricola = T.matricola)
JOIN APPELLO A ON (A.data = T.data AND A.nome = T.nome)
JOIN INSEGN_SESSIONE INS ON (INS.datainizio= A.datainizio AND INS.datafine = A.datafine)
JOIN SESSIONE SE ON (INS.datainizio= SE.datainizio AND INS.datafine = SE.datafine)
WHERE T.voto >= 18
ORDER BY C.codcorso
```

VA3) Vista che riporta per ogni corso e per ogni studente il tipo di prova svolto, la relativa data e il voto ottenuto durante tale prova

```
CREATE VIEW ANDAMENTO
 AS
SELECT A.nome,S.matricola,S.nome,S.cognome,
      case N.ordine when 0 then 'Prova completa' when 1  then 'Primo parziale' when 2 then 'Secondo
parziale' else 'Verbalizzazione' end as tipo_prova,T.data,T.voto
FROM STUDENTE S JOIN TIENE T ON (S.matricola = T.matricola)
    JOIN APPELLO A ON  (A.data  =  T.data  AND  A.nome = T.nome)
    JOIN NUMORDINE N ON (N.cod = A.cod)
```

Attualmente , Esse3 permette di salvare i dati ed esportarli su file excel. La tesi aveva lo scopo di definire lo schema del database che mantiene i dati di tutti gli esami degli studenti ed elaborare una procedura software che permetta di caricare i dati provenienti dai file excel di Esse3 integrandoli nel database.

Sul database vengono poi create viste che forniscono una visione d'insieme dell'andamento degli esami per un determinato insegnamento in un certo periodo temporale (per esempio per anno accademico).

Durante il nostro studio siamo partiti dei requisiti che una base di dati che gestisce gli esami deve soddisfare andando a produrre lo schema E/R del database. Siamo andati a modificare questo schema E/R sulla base dei dati esportati dall'applicativo Esse3. Con lo schema E/R così raffinato abbiamo prodotto lo schema relazionale con le regole della progettazione logica relazionale, generato lo script di creazione della nostra base di dati.

Abbiamo realizzato un software che permetta di caricare i dati presenti in file excel nel database locale ; a tal fine abbiamo usato il progetto *java apache POI* per leggere le informazioni memorizzate sui file excel esportati da esse3, poi attraverso una connessione *JDBC* al database abbiamo definito una procedura di importazione dati nel database locale.

Per rendere più agevole una visione d' insieme sui dati presenti nel database. Abbiamo definito alcune viste che forniscono un andamento degli esami per insegnamento per un certo periodo temporale.

La tesi ha realizzato tutti gli obiettivi che ci eravamo posti.

Sarebbe possibile migliorare il software , in modo che l'applicativo fornisca all'utente un output descrittivo dell'operazione di caricamento dati effettuata (ad es. numero di studenti inseriti nel database , il numero dei voti inseriti, piuttosto che l'appello già presente ecc…) , per una corretta integrazione dei dati nella base di dati.

BIBLIOGRAFIA E SITOGRAFIA

1. *Progetto di Basi di Dati Relazionali lezioni ed esercizi*
   Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra , Maurizio Vincini

2. *http://poi.apache.org/download.html*
3. *http://www.lukeonweb.net/leggi/286/jdbc_introduzione_a_java_database_connectivity.asp*
4. *http://www.appuntisoftware.it/apache-poi-manipolare-documenti-microsoft-in-java/*
5. *http://www.microsoft.com/it-it/download/details.aspx?id=21599*
6. *http://www.eclipse.org/downloads/*
7. *http://it.wikipedia.org/wiki/Eclipse_(informatica)*
8. *http://poi.apache.org/spreadsheet/index.html*
9. *http://forums.devshed.com/ms-sql-development-95/jdbc-and-sql-server-2008t-579217.html*
10. *http://www.hwupgrade.it/forum/archive/index.php/t-1114878.html*
11. *http://stackoverflow.com/questions/746259/java-string-to-datetime-conversion-issue*
12. *http://stackoverflow.com/questions/759036/how-to-convert-string-into-time-format-and-add-two-hours*
13. *http://java.ittoolbox.com/groups/technical-functional/java-l/reading-data-from-excel-file-using-jexcel-api-and-storing-the-excel-data-in-to-database-table-1137545*
14. *http://www.developpez.net/forums/d1028502/java/general-java/jdbc/inserer-base-donnees-jbdc/*
15. *http://www.jroller.com/coreteam/entry/import_data_from_excel_to*
16. *http://java.ittoolbox.com/groups/technical-functional/java-l/read-data-from-excel-using-java-and-insert-it-in-to-mssql-database-1310853*
17. *http://nitinaggarwal.wordpress.com/2009/02/21/reading-contents-of-excel-using-java-code/*
18. *http://code.geekinterview.com/java/read-excel-file-data-java-code.html*
19. *Programmazione a oggetti in Java : dai fondamenti a internet / Giacomo Cabri, Franco Zambonelli. - Bologna : Pitagora, °2003*
20. *http://www.cineca.it/sites/default/files/esse3_articolo.pdf*

Desidero ringraziare sentitamente il prof. LAURA PO che mi ha accompagnato e sostenuto durante l'intero periodo di stesura della tesi, sempre disponibile e presente, nonostante i suoi innumerevoli impegni.

Non posso non ringraziare i miei genitori in Camerun, senza i quali non avrei ricevuto l'educazione civica ed accademica che oggi ho e che mi hanno condotto a questa importante tappa. Mi mancano le parole per ringraziare il mio fratello LUC, ROSTAND, CYRILLE , LINDA e WILLIAM per i loro sostegni e preziosi consigli.

Gli ultimi ma non meno importanti ringraziamenti vanno a tutti i miei amici, che mi hanno sostenuto ed incoraggiato, senza i quali non avrei mai potuto farcela.