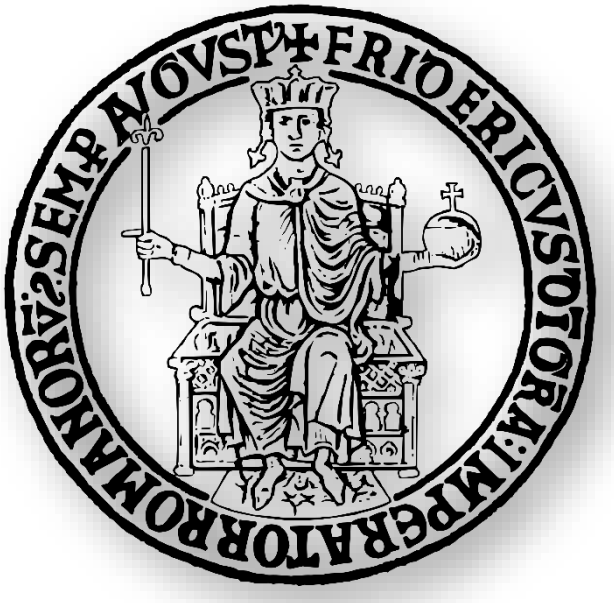
Università degli Studi di Napoli Federico II



Corso di Laurea in Ingegneria dell'informazione

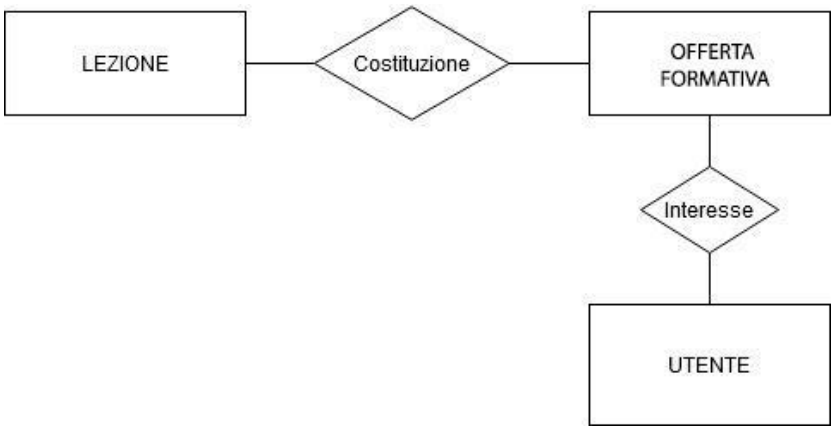
A.A 2019/2020

ESERCITAZIONE BASI DATI:
PIATTAFORMA ONLINE ADIBITA AL WEBLEARNING

STUDENTI:		*DOCENTE:*
Mazza Francesco	*N46004564*	*Angelo Chianese*
Paesano Alessio	*N46004473*	
Riccio Emanuele	*N46005201*	

1.0 INFORMAZIONI PRELIMINARI SULLA REALTÀ' DI INTERESSE

Si vuole modellare il database di un sito web adibito all'erogazione di corsi online. La piattaforma in questione è allestita sotto forma di sito web, al quale gli utenti possono liberamente connettersi. Gli utenti che si collegano al portale web possono essere studenti o docenti. La pagina web permette ad un utente (docente o studente) l'accesso alle videolezioni. Il docente può erogare il materiale didattico attraverso la creazione di un corso sulla piattaforma online. Un corso è formato da più lezioni suddivise in unità elementari di apprendimento. In ogni pagina di ciascuna lezione è possibile inserire lezioni in formato testuale oppure video. L'accesso al contenuto dei corsi è destinato ai soli studenti che, dopo essersi iscritto alla piattaforma, acquistano un corso messo a disposizione dai docenti e seguono le lezioni in modalità asincrona. Il successo di un corso online si valuta mediante i feedback rilasciati dagli studenti che lo acquistano. Lo studente può valutare ogni lezione e, dopo aver seguito la totalità delle stesse, immettere una valutazione complessiva del corso e nonché sostenere una prova finale per accertare la sua preparazione.



1.1 ANALISI REQUISITI

Per individuare con maggiore chiarezza le caratteristiche che il database deve supportare, riportiamo delle informazioni aggiuntive sugli elementi di interesse e sulle informazioni che sono collegate alla loro gestione. (requisiti per gestire le informazioni inerenti alla realtà di interesse).

DATI UTENTE

Al fine di poter usufruire della piattaforma, l'utente, in fase di registrazione, dovrà fornire nome, cognome, codice fiscale, e-mail e password per l'autenticazione. A seconda che l'utente sia uno Studente o un Docente si dovranno specificare informazioni aggiuntive.

DATI CORSO

Ogni corso deve essere creato da un docente, specificando nome, prezzo, e descrizione del contenuto. Il corso è formato da lezioni, dove ogni lezione è composta da più entità elementari di apprendimento, le quali possono essere sia in formato video che testuali.

DATI AGGIUNTIVI SUI CORSI

Lo studente, solo dopo aver acquistato un corso, ne può usufruire in modalità asincrona. Lo studente ha la facoltà di rilasciare un riscontro su ciascuna lezione sull'intero corso nonché l'opportunità di sostenere una prova finale.

1.2 Glossario dei termini

TERMINE	DESCRIZIONE	ATTINENZA
Utente	Colui che usufruisce del servizio.	*Studente, Docente, Offerta formativa.*
Studente	Utente che è interessato alla fruizione dei corsi.	*Offerta formativa, Lezione, Utente*
Docente	Utente che detiene la cattedra di un corso.	*Offerta formativa, Corso, Lezione, Utente*
Lezione	Materiale costitutivo del corso, formato da unità elementari di apprendimento.	*Corso*
Offerta formativa	Insieme dei corsi offerti agli studenti.	*Lezione, Corso*
Corso	Insieme di lezioni su un determinato argomento.	*Lezione, Docente*
Bundle	Insieme di più corsi acquistabili in una volta.	*Corsi*
Unità di apprendimento	Singola slide che contiene il materiale didattico riguardante la lezione corrispondente.	*Lezione*

1.3 Vincoli

CREAZIONE CORSO

1. Si suppone che un docente possa creare più corsi e che un corso possa avere un solo docente.
2. Un docente, per poter essere eliminato, deve necessariamente gestire i corsi ad esso associato.

ACQUISTO CORSO

1. Ogni studente può acquistare più corsi e lo stesso corso può essere acquistato da più studenti.
2. L'acquisto di uno specifico bundle da parte di uno studente può essere effettuato una sola volta.
3. L'acquisto può essere effettuato solamente se lo studente ha un saldo superiore o uguale al prezzo del corso.
4. Il saldo, a prescindere dall'acquisto di un corso, non puo' ammettere un valore negativo.

VOTAZIONE LEZIONE\CORSO

1. Uno studente può fornire una sola valutazione numerica (1-5) per ogni singola lezione, ed infine una sola valutazione testuale per ogni singolo corso.
2. La votazione da parte di uno studente sia per un corso che per le lezioni che vi appartengono può essere effettuata solo se lo studente ha acquistato il corso.

CONTENUTO LEZIONE

1. Una lezione può essere composta da elementi testuali oppure in formato video, non è possibile inserire entrambi i contenuti nella stessa pagina. Ciascuna pagina non può essere vuota.

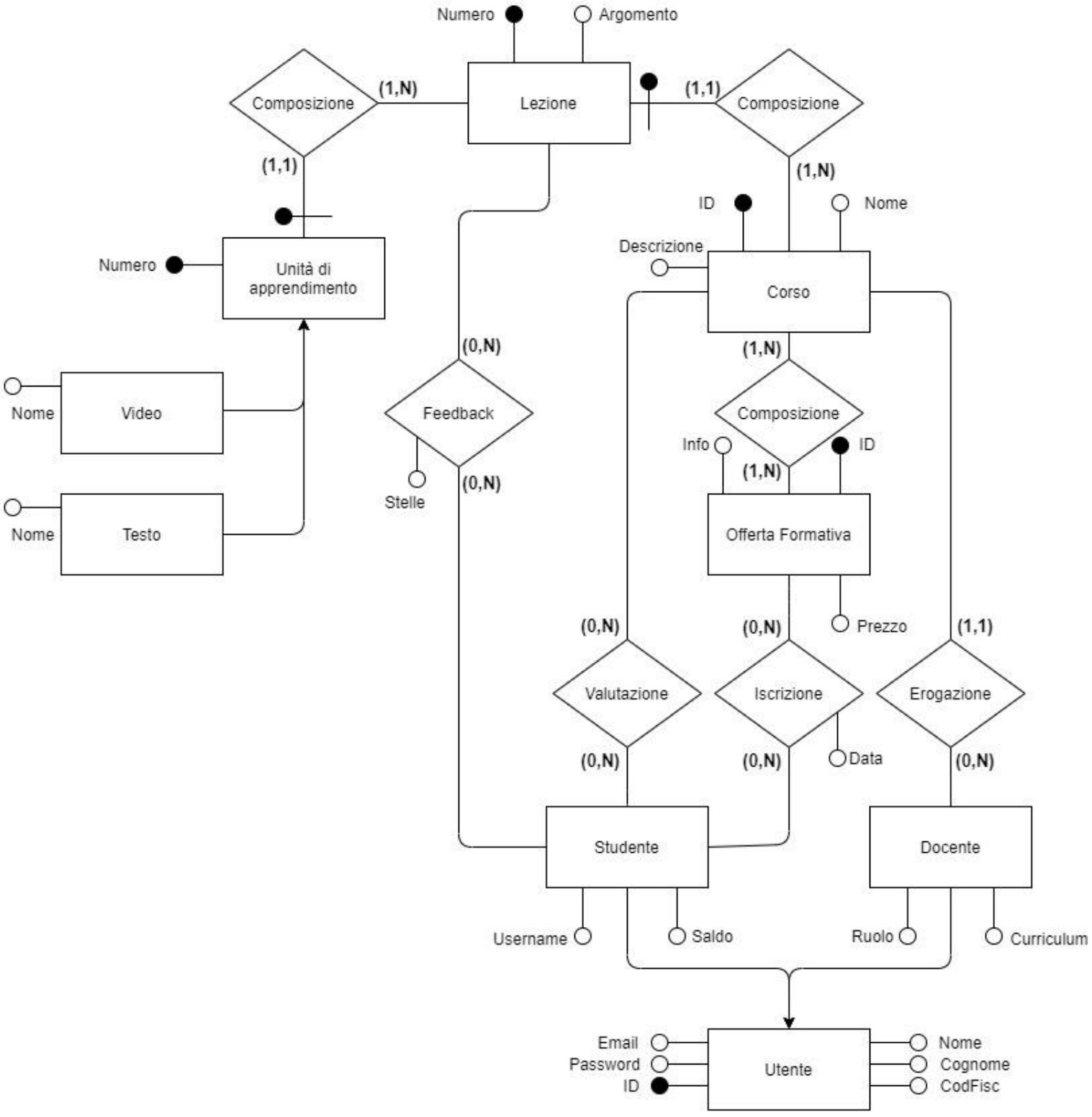
PASSWORD

1. Lo storage sul DB delle password deve essere eseguito garantendo la sicurezza delle stesse.

1.4 Dimensionamento

ELEMENTO	DIMENSIONE
UTENTI	*MAX 1.500*
STUDENTI	*MAX 1.000*
DOCENTI	*MAX 500*
CORSI	*MAX 500*
LEZIONI PER CORSO	*MAX 30*
UNITA' PER LEZIONE	*MAX 20*
ISCRIZIONI	*MAX 10.000*
VALUTAZIONI	*MAX 10.000*
FEEDBACK	*MAX 30.000*

1.5 Modello E-R



1.6 Dizionario dei dati

CLASSE	ATTRIBUTI	IDENTIFICATORE
Utente	ID, Nome, Cognome, Codice Fiscale, E-mail, Password	*ID*
Studente	Username, Saldo	*(ID)*
Docente	Curriculum, Ruolo	*(ID)*
Offerta formativa	ID, Prezzo, Info	*ID*
Corso	ID, Nome, Descrizione	*ID*
Lezione	Numero, Argomento	*Numero, (ID)*
Unità apprendimento	Numero, Titolo, ID	*Numero, (Numero),(ID)*

**gli identificatori fra parentesi sono da riferirsi come esterni, ereditati da una gerarchia o mediante un' associazione.*

1.7 Associazioni

ASSOCIAZIONE	DESCRIZIONE	CLASSI COINVOLTE
ISCRIZIONE	Mette in comunicazione uno studente con l'offerta formativa, mediante l'iscrizione ad essa.	*STUDENTE(0,N)* *OFFERTA FORMATIVA(0,N)*
VALUTAZIONE	Ogni studente può condividere un feedback su un corso mediante una valutazione testuale.	*STUDENTE(0,N)* *CORSO(0,N)*
EROGAZIONE	Ogni docente è in relazione con i corsi che allestisce.	*DOCENTE(0,N)* *CORSO(1,1)*
COMPOSIZIONE *(offerta-corso)*	Ogni offerta formativa è composta da un insieme di corsi.	*OFFERTA FORMATIVA(0,N)* *CORSO(1,N)*
COMPOSIZIONE *(corso-lezione)*	Un corso è composto da un insieme di lezioni.	*CORSO(1,N)* *LEZIONE(1,1)*
FEEDBACK	Ogni studente può relazionarsi a ciascuna lezione rilasciando un feedback.	*STUDENTE(0,N)* *LEZIONE(0,N)*
COMPOSIZIONE *(lezione-unità apprendimento)*	Le unità di apprendimento sono l'elemento essenziale che costituisce una lezione	*LEZIONE(1,N)* *UNITÁ APPRENDIMENTO(1,1)*

2.0 Progettazione Logica

Il passaggio successivo consiste nell'avvicinare il modello ER ad una implementazione concreta. Si ricordi che il modello entity-relation ha l'intento di rappresentare la realtà di interesse ed evidenziare gli schemi di relazione tra le entità in gioco. L'ottenimento di un modello logico è, invece, essenziale per rendere evidente il collegamento tra i dati di interesse e la loro concreta rappresentazione. Affinché tale processo sia possibile è necessario ristrutturare lo schema E-R secondo i seguenti criteri fondamentali:

1. Eliminazione di attributi composti:
   Si noti che all'interno dello schema non sono previsti dati di questo tipo.

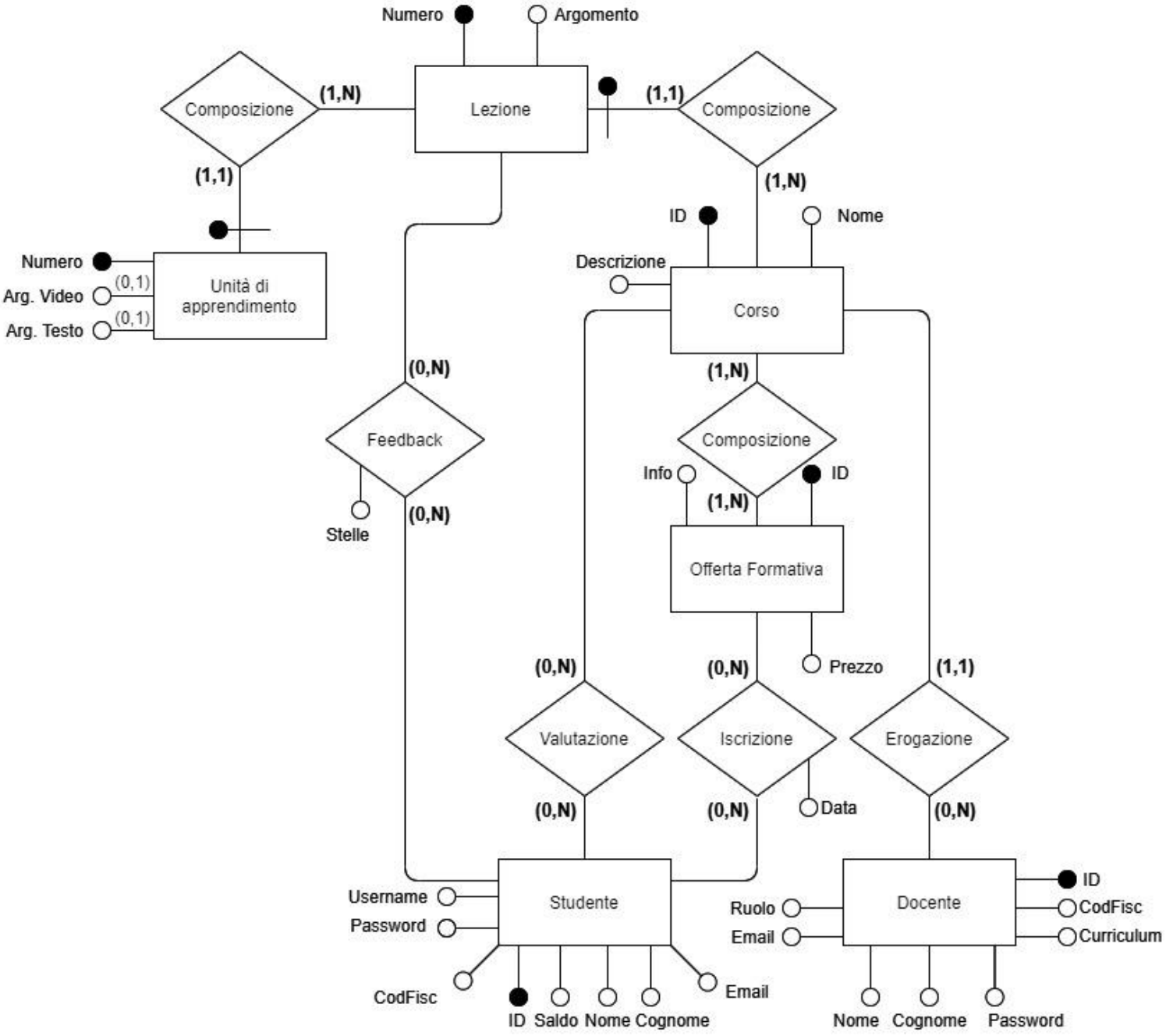
2. Introduzione di una chiave primaria per ciascuna entità:
   Le entità più importanti (utenti, offerte formative, corsi) sono dotate di un ID identificativo univoco per ciascuna delle loro istanze; le entità loro associate, invece, saranno dotate di una chiave numerica dipendente dall'entità di provenienza.

3. Eliminazione delle gerarchie:
   3.1. La gerarchia Utente-Studente-Docente viene risolta accorpando la classe padre nelle classi figlie, è stato scelto di procedere in questo senso per evitare attributi nulli specifici di una sola delle classi figlie (in quanto la gerarchia è di tipo totale-disgiunta).
   3.2. La gerarchia Unità di apprendimento-Video-Testo può essere semplificata accorpando le classi figlie nella classe padre, in particolare, la classe Unità di apprendimento presenterà gli attributi ArgomentoVideo e ArgomentoTesto, la presenza di attributi nulli identificherà immediatamente la tipologia dell'elemento di apprendimento.

4. Accorpamento/Divisione tra tabelle:
   È stato deciso, già in fase di progettazione del modello E-R, di suddividere le tabelle Offerta Formativa e Corso, allo scopo di evidenziare la relazione tra lo studente ed il piano di studi a cui è interessato, conservando però le specificità di ogni corso, ed il relativo docente che ne detiene l'insegnamento. Inoltre, le classi Lezione e Unità di Apprendimento sono state già divise, al fine di evidenziare le caratteristiche specifiche di ogni unità di apprendimento che sarebbero passate in secondo piano nella tabella Lezione.

5. Analisi delle ridondanze:
   È stato evitato l'inserimento di qualsiasi valore calcolabile. Le associazioni Feedback e Valutazione non presentano una reale ridondanza, in quanto, la prima è riferita alla singola lezione e l'altra al singolo corso. Inoltre, il feedback si misura in stelle (intero), mentre la valutazione è un commento di tipo testuale.

2.1 Modello E-R ristrutturato



2.2 Traduzione dello schema relazionale

Studente(ID, *Nome, Cognome, Username, E-mail, Password, CodFisc, Saldo*)

Docenti(ID, *Nome, Cognome, Ruolo , E-mail, Password, Codfisc, Curriculum*)

Lezioni(Numero, ID_Corso:Corsi, Argomento)

Corsi(ID, *ID_Docente:Docenti, Nome, Descrizione*)

OfferteFormative(ID, *Prezzo,Info*)

Unità di apprendimento(Numero, Numero_Lezione:Lezioni, ID_Corso:Lezioni, *Argomento video, Argomento testo*)

Feedback(ID_Studente:Studenti, Numero_Lezione:Lezioni, ID_Corso:Lezioni, *Stelle*)

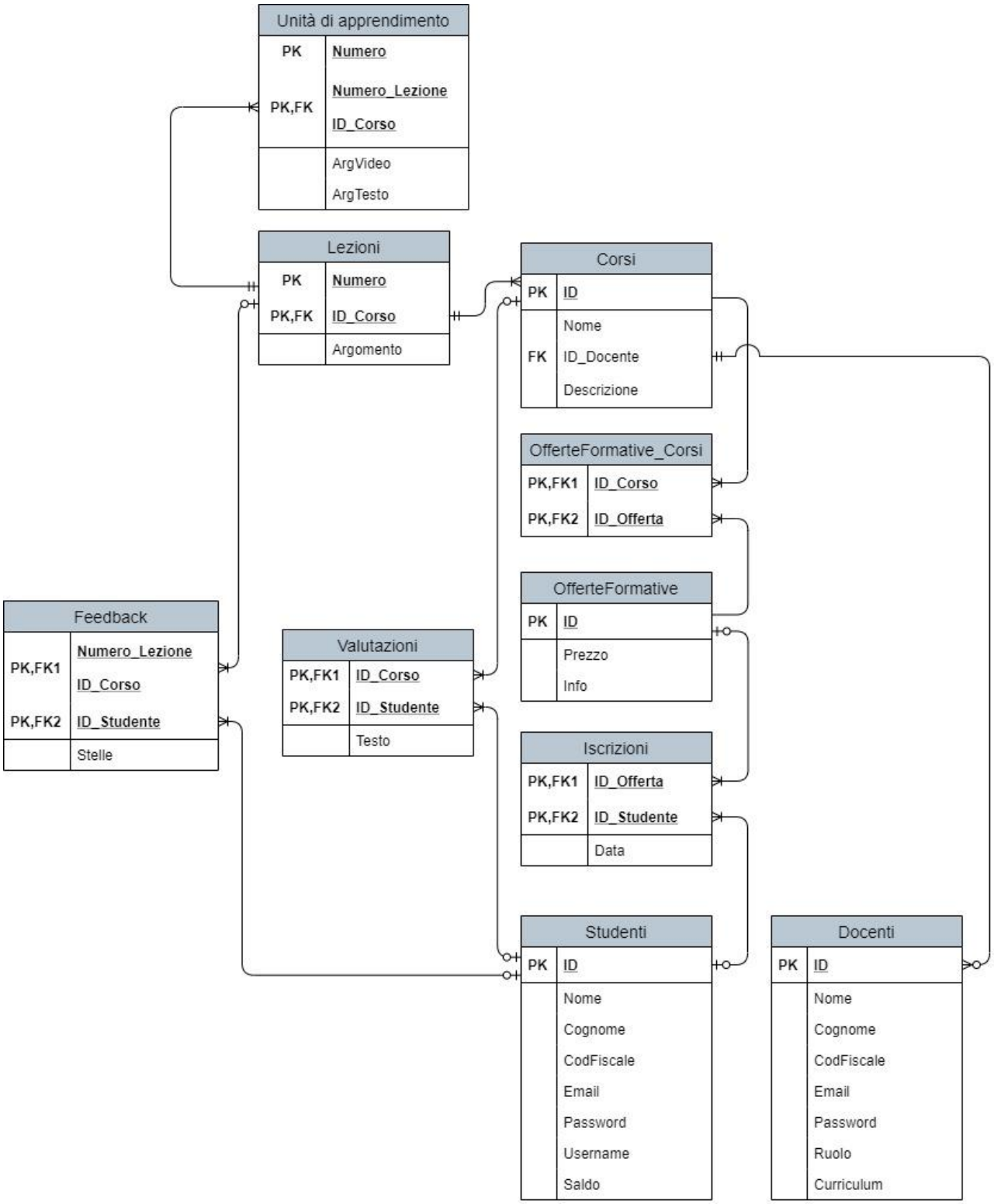
OfferteFormative-Corsi(ID_Offerta:OfferteFormative, *ID_Corso:Corsi*)

Iscrizioni(ID_Offerta:OfferteFormative, ID_Studente:Studenti, *data*)

Valutazioni(ID_Studente:Studenti, ID_Corso:Corsi)

*NOTAZIONE: **":"** indica la tabella di provenienza*

2.3 Schema E-R tradotto



3.0 Implementazione SQL

3.1 CREATE TABLE

```
CREATE TABLE Studenti(
ID int,
Nome varchar2(50) not null,
Cognome Varchar2(50) not null,
CodFiscale varchar2(16) not null,
Email varchar2(100) not null,
Password varchar2(32) not null,
Username varchar2(20) not null,
Saldo number not null,

CONSTRAINT PK_STUDENTI primary key(ID),
CONSTRAINT VALIDITA_MAIL_STUDENTI check (Email like '__%@__%.__%' and
Email not like '%@%@%'),
CONSTRAINT HASHED_PWD_STU check (LENGTH(Password) = 32),
CONSTRAINT LEN_CF_STU check (LENGTH(CodFiscale) = 16)
);

CREATE TABLE Docenti (
ID int,
Nome varchar2(50) not null,
Cognome Varchar2(50) not null,
CodFiscale varchar2(16) not null,
Email varchar2(100) not null,
Password varchar2(32) not null,
Ruolo varchar2(20) not null,
Curriculum blob not null,

CONSTRAINT PK_DOCENTI primary key(ID),
CONSTRAINT VALIDITA_MAIL_DOCENTI check (Email like '__%@__%.__%' and
Email not like '%@%@%'),
CONSTRAINT HASHED_PWD_DOC check (LENGTH(Password) = 32),
CONSTRAINT LEN_CF_DOC check (LENGTH(CodFiscale) = 16)
);

CREATE TABLE Corsi (
ID int,
Nome varchar2(50) not null,
ID_Docente int not null,
Descrizione varchar2(500)  not null,

CONSTRAINT PK_CORSI primary key (ID)
);

CREATE TABLE Lezioni (
Numero smallint,
ID_Corso int,
Argomento Varchar2(100) not null,

CONSTRAINT PK_LEZIONI primary key (Numero, ID_Corso)
);
```

```
CREATE TABLE OfferteFormative(
ID int,
Prezzo float not null,
Info varchar2(500) not null,

CONSTRAINT PK_OFFERTEFORMATIVE primary key (ID)
);


CREATE TABLE UnitaDiApprendimento(
Numero int,
Numero_Lezione smallint,
ID_Corso int,
ArgVideo varchar2(100),
ArgTesto varchar2(100),

CONSTRAINT PK_UNITADIAPPRENDIMENTO primary key (Numero, Numero_Lezione,
ID_Corso),
CONSTRAINT VALIDITA_PAGINA check ((ArgVideo is null and ArgTesto is not
null) or (ArgTesto is null and ArgVideo is not null))
);


CREATE TABLE Valutazioni (
ID_Corso int,
ID_Studente int,
Testo Varchar2(500) not null,

CONSTRAINT PK_VALUTAZIONI primary key (ID_Studente, ID_Corso)
);

CREATE TABLE Feedback (
Numero_Lezione int,
ID_Corso int,
ID_Studente int,
Stelle int  not null,

CONSTRAINT PK_FEEDBACK primary key (Numero_Lezione, ID_Corso,
ID_Studente),
CONSTRAINT NUM_STELLE check (Stelle > 0 and  Stelle < 6)
);

CREATE TABLE Iscrizioni(
ID_Offerta int,
ID_Studente int,
data date not null,

CONSTRAINT PK_ISCRIZIONI primary key (ID_Offerta,  ID_Studente)
);

CREATE TABLE OfferteFormative_Corsi(
ID_Corso int,
ID_Offerta int,

CONSTRAINT PK_OFFORMATIVE_CORSI primary key (ID_Offerta,  ID_Corso)
);
```

3.2 ALTER TABLE

```
ALTER TABLE Lezioni
ADD CONSTRAINT FK_LEZIONI_CORSI FOREIGN KEY (ID_Corso)
REFERENCES Corsi(ID)
ON DELETE CASCADE;

ALTER TABLE Corsi
ADD CONSTRAINT FK_CORSI_DOCENTI FOREIGN KEY (ID_Docente)
REFERENCES Docenti(ID);
--Per poter eliminare un docente e' necessario gestire i corsi creati
dallo stesso (Non c'e' "ON DELETE CASCADE")

ALTER TABLE UnitaDiApprendimento
ADD CONSTRAINT FK_UNITAPPR_LEZIONI FOREIGN KEY (Numero_Lezione,
ID_Corso)
REFERENCES Lezioni(Numero, ID_Corso)
ON DELETE CASCADE;

ALTER TABLE Feedback
ADD CONSTRAINT FK_FEEDBACK_LEZIONI FOREIGN KEY (Numero_Lezione,
ID_Corso)
REFERENCES Lezioni(Numero, ID_Corso)
ON DELETE CASCADE;

ALTER TABLE Feedback
ADD CONSTRAINT FK_FEEDBACK_STUDENTI FOREIGN KEY (ID_Studente)
REFERENCES Studenti(ID)
ON DELETE CASCADE;

ALTER TABLE OfferteFormative_Corsi
ADD CONSTRAINT FK_OFFORM_CORSI FOREIGN KEY (ID_Corso)
REFERENCES Corsi(ID)
ON DELETE CASCADE;

ALTER TABLE OfferteFormative_Corsi
ADD CONSTRAINT FK_OFFORM_OFF FOREIGN KEY (ID_Offerta)
REFERENCES OfferteFormative(ID)
ON DELETE CASCADE;

ALTER TABLE Iscrizioni
ADD CONSTRAINT FK_ISCRIZIONI_OFF FOREIGN KEY (ID_Offerta)
REFERENCES OfferteFormative(ID)
ON DELETE CASCADE;

ALTER TABLE Iscrizioni
ADD CONSTRAINT FK_ISCRIZIONI_STUDENTI FOREIGN KEY (ID_Studente)
REFERENCES Studenti(ID)
ON DELETE CASCADE;

ALTER TABLE Valutazioni
ADD CONSTRAINT FK_VALUTAZIONI_STUDENTI FOREIGN KEY (ID_Studente)
REFERENCES Studenti(ID)
ON DELETE CASCADE;
```

3.3 QUERY

ELENCO DEI CORSI ASSOCIATI PER OGNI DOCENTE

```
SELECT d.Nome, d.Cognome, c.Nome AS NOME_CORSO
FROM docenti d JOIN corsi c
ON c.ID_DOCENTE = d.ID;
```

NUMERO DI ELEMENTI VIDEO E TESTUALI PER OGNI LEZIONE DI UN CERTO CORSO

```
SELECT l.Numero AS NUMERO_LEZIONE, l.Argomento, count(u.argvideo) AS
NUMERO_VIDEO, count(u.argtesto) AS NUMERO_TESTO
FROM lezioni l JOIN unitadiapprendimento u ON u.NUMERO_LEZIONE =
l.Numero
WHERE l.ID_CORSO = 1 --ID Del corso di interesse
GROUP BY l.Numero,l.argomento
```

CORSI LA CUI VALUTAZIONE MEDIA DELLE LEZIONI È MAGGIORE O UGUALE DI 4 STELLE

```
SELECT  c.ID, c.nome, avg(f.stelle) AS MEDIA
FROM
(corsi c JOIN lezioni l ON l.ID_CORSO = c.id)
JOIN feedback f ON f.ID_CORSO = l.ID_CORSO AND f.NUMERO_LEZIONE =
l.NUMERO
GROUP BY c.ID, c.nome
HAVING avg(f.stelle) >= 4;
```

STABILIRE QUALE DOCENTE HA PUBBLICATO PIÚ CORSI DI TUTTI

```
SELECT d.nome,d.cognome, count(c.ID) AS NUMERO_CORSI
FROM docenti d JOIN corsi c ON c.ID_Docente = d.ID
GROUP BY (d.id, d.nome, d.cognome)
HAVING count(c.ID) =
(
SELECT MAX(count(c1.ID))
FROM docenti d1 JOIN corsi c1 ON c1.ID_Docente = d1.ID
GROUP BY (d1.id)
)
```

RAPPORTO, PER OGNI STUDENTE, DEI CORSI RECENSITI DIVISO QUELLI ACQUISTATI

```
SELECT s.Nome,s.Cognome, count(v.ID_CORSO) / count(c.ID) * 100 AS
PERCENTUALE
FROM ((((studenti s JOIN iscrizioni i ON s.ID = i.ID_Studente)
JOIN offerteformative o ON (o.ID = i.ID_Offerta))
JOIN offerteformative_corsi oc ON o.ID = (oc.ID_Offerta))
JOIN corsi c ON (c.ID = oc.ID_Corso))
LEFT JOIN valutazioni v ON (v.ID_Corso = c.ID AND v.ID_Studente =
s.ID)
GROUP BY (s.ID,s.nome,s.cognome)
```

3.4 TRIGGER

CONTROLLO SULL'INSERIMENTO DI UNA VALUTAZIONE:
IL TRIGGER ASSICURA CHE OGNI VALUTAZIONE SIA RILASCIATA DA STUDENTI CHE ABBIANO ACQUISTATO UNA QUALSIASI OFFERTA FORMATIVA IN CUI SIA PRESENTE IL CORSO VALUTATO

```
CREATE OR REPLACE TRIGGER valutazione_corso_studente
BEFORE INSERT ON Valutazioni
FOR EACH ROW               --trigger di tupla

DECLARE
non_iscritto EXCEPTION;    --eccezione
NUMISCR INTEGER;           --contatore


BEGIN
SELECT COUNT(*) INTO NUMISCR
FROM (((Studenti d join iscrizioni i ON (d.ID = i.ID_Studente))
JOIN offerteformative o ON (o.ID = i.ID_offerta))
JOIN offerteformative_corsi oc ON (oc.ID_Offerta = o.ID))
JOIN corsi c ON (c.ID = oc.ID_Corso) --eliminabile
WHERE d.ID = :new.ID_Studente AND c.ID = :new.ID_Corso;

IF NUMISCR = 0 THEN
    RAISE non_iscritto;
END IF;

EXCEPTION
WHEN non_iscritto THEN RAISE_application_error(-20069, 'Studente non
iscritto');
END;
```

CONTROLLO SULL'INSERIMENTO DI UN FEEDBACK:
IL TRIGGER ASSICURA CHE OGNI FEEDBACK SIA RILASCIATO DA STUDENTI CHE ABBIANO ACQUISTATO IL CORSO ASSOCIATO ALLA LEZIONE VALUTATA

```
CREATE OR REPLACE TRIGGER feedback_lezione_studente
BEFORE INSERT ON feedback
FOR EACH ROW

DECLARE
non_iscritto EXCEPTION; --eccezione
NUMISCR INTEGER;        --contatore

BEGIN
--conto in quante offertete formative a cui lo studente e'  iscritto
--e'presente il corso che contiene la lezione a cui si vuole inserire
il feedback

SELECT COUNT(*) INTO NUMISCR
FROM (offerteformative o JOIN offerteformative_corsi oc ON
(oc.ID_Offerta = o.ID))
JOIN iscrizioni i ON i.ID_Offerta = oc.id_offerta
WHERE (oc.ID_Corso = :new.ID_Corso AND i.ID_Studente =
:new.ID_Studente);

IF NUMISCR = 0 THEN    --controllo contatore
    RAISE non_iscritto;
END IF;

EXCEPTION
WHEN non_iscritto THEN RAISE_application_error(-20004, 'Studente non
iscritto');
END;
```

CONTROLLO SULL'INSERIMENTO DI UNA NUOVA LEZIONE:
IL TRIGGER ASSICURA CHE L'INSERIMENTO DI UNA NUOVA LEZIONE SIA POSSIBILE SOLO SE SI TRATTA DELLA PRIMA O È PRESENTE QUELLA PRECEDENTE:

```
CREATE OR REPLACE TRIGGER numero_lezioni_crescente
BEFORE INSERT ON Lezioni
FOR EACH ROW

DECLARE
non_crescente EXCEPTION;
NUM_LEZIONE constant Lezioni.Numero%TYPE := :new.Numero;
CORSO constant Lezioni.ID_Corso%TYPE := :new.ID_Corso;
LEZIONE_PRECEDENTE integer;
var INTEGER;

BEGIN
IF( NUM_LEZIONE <> 1 ) THEN
    SELECT count(l.Numero) INTO LEZIONE_PRECEDENTE
    FROM lezioni l
    WHERE l.ID_Corso = CORSO;
    IF(LEZIONE_PRECEDENTE <> NUM_LEZIONE - 1) THEN
        RAISE non_crescente;
    END IF;
END IF;

EXCEPTION
WHEN non_crescente THEN RAISE_application_error(-20003, 'Inserire le
lezioni in ordine crescente!');
END;
```

3.5 POPOLAMENTO DELLA BASE DATI

Per testare il corretto funzionamento del database, inseriamo dei valori di esempio, al fine di valutare la corretta gestione delle informazioni. Le funzioni di popolamento e svuotamento delle tabelle sono state inserite in un package, al fine di rendere piú intuitiva la loro chiamata.

```
CREATE OR REPLACE PACKAGE Sito
IS

PROCEDURE POPOLAMENTO;
PROCEDURE SVUOTAMENTO;

END Sito;
```

```
CREATE OR REPLACE PACKAGE BODY Sito
IS
PROCEDURE POPOLAMENTO
IS
BEGIN

    INSERT INTO STUDENTI VALUES(1, 'Emanuele', 'Riccio', 'RCCRCC9992223478',
'emanuele@email.it', '366266C716B5843854AF039851828F5E', '@lelettrone', 1000);

    INSERT INTO STUDENTI VALUES(2, 'Francesco', 'Mazza', 'eqweqrw534533531',
'francesco@email.it', '366266C716B5843854AF039851828F5E', '@framazzaa', 1000);

    INSERT INTO STUDENTI VALUES(3, 'Alessio', 'Paesano', '3142324534534535',
'alessio@email.it', '366266C716B5843854AF039851828F5E', '@alepaes', 1000);

    INSERT INTO DOCENTI VALUES(1, 'Angelo', 'Chianese',  '3245343524342543',
'angelo.chianese@unina.it', 'MzBFTE9ERQ==00000000000000000000','Docente Informatica',
hextoraw('453d7a34'));

    INSERT INTO DOCENTI VALUES(2, 'Vincenzo', 'Moscato', '4664765353535435',
'vinni.mosc@unina.it', 'MzBFTE9ERQ==00000000000000000000','Docente Informatica',
hextoraw('453d7a34'));

    INSERT INTO DOCENTI VALUES(3, 'Niels', 'Kowalzig',   '2232323232222221',
'niels.omast@unina.it', 'MzBFTE9ERQ==00000000000000000000','Docente Geometria',
hextoraw('2322324'));

    INSERT INTO DOCENTI VALUES(4, 'Francesco', 'Chiacchio',   '6666666666222222',
'francesco.chiacchio@unina.it', 'MzBFTE9ERQ==00000000000000000000','Docente
Matematica', hextoraw('2322324'));

    INSERT INTO CORSI VALUES(1, 'Basi di Dati', 1, 'Studio dei Database');

    INSERT INTO CORSI VALUES(2, 'Fondamenti di Informatica', 1, 'Introduzione all
informatica');

    INSERT INTO CORSI VALUES(3, 'Spionaggio industriale', 2, 'Non legale');

    INSERT INTO CORSI VALUES(4, 'Intreccio cesti in vimini', 2, 'Rilassante');

    INSERT INTO CORSI VALUES(5, 'Algebra e Geometria', 3, 'Principi di geometria e
algebra');

   INSERT INTO CORSI VALUES(6, 'Analisi 2', 4, 'Analisi in piu dimensioni');

   INSERT INTO LEZIONI VALUES(1, 1,'Modello ER');

   INSERT INTO LEZIONI VALUES(2, 1,'Modello Logico');

   INSERT INTO LEZIONI VALUES(3, 1,'SQL');
```

```
INSERT INTO LEZIONI VALUES(4, 1,'Trigger');

    INSERT INTO LEZIONI VALUES(1, 5, 'Vettori');

    INSERT INTO LEZIONI VALUES(2, 5, 'Matrici');

    INSERT INTO LEZIONI VALUES(3, 5, 'Spazi Vettoriali');

    INSERT INTO LEZIONI VALUES(4, 5, 'Teorema Spettrale e Jordanizzazione');

    INSERT INTO LEZIONI VALUES(1, 6, 'Funzioni multivariabile');

    INSERT INTO LEZIONI VALUES(2, 6,'Gradiente, Rotore, Divergenza');

    INSERT INTO LEZIONI VALUES(3, 6, 'Forme differenziali');

    INSERT INTO LEZIONI VALUES(4, 6, 'Eq differenziali');

    INSERT INTO LEZIONI VALUES(1, 2 ,'Modello Von-Neumman');

    INSERT INTO LEZIONI VALUES(2, 2, 'Automa a stati finiti');

    INSERT INTO LEZIONI VALUES(3, 2, 'Ciclo for');

    INSERT INTO LEZIONI VALUES(4, 2, 'P = NP');

    INSERT INTO OFFERTEFORMATIVE VALUES(1, 300, 'Introduction to the language of
Science');

    INSERT INTO OFFERTEFORMATIVE VALUES(2, 100, 'Introduction to the language of
Information');

    INSERT INTO OFFERTEFORMATIVE_CORSI VALUES(5,1);

    INSERT INTO OFFERTEFORMATIVE_CORSI VALUES(6,1);

    INSERT INTO OFFERTEFORMATIVE_CORSI VALUES(2,2);

    INSERT INTO OFFERTEFORMATIVE_CORSI VALUES(1,2);

    INSERT INTO ISCRIZIONI VALUES(1, 1, TO_DATE('2016/05/03', 'yyyy/mm/dd'));

    INSERT INTO ISCRIZIONI VALUES(2, 1, TO_DATE('2020/02/23', 'yyyy/mm/dd'));

    INSERT INTO ISCRIZIONI VALUES(2, 3, TO_DATE('2019/09/17', 'yyyy/mm/dd'));

    INSERT INTO ISCRIZIONI VALUES(2, 2, TO_DATE('2018/08/11', 'yyyy/mm/dd'));

    INSERT INTO FEEDBACK VALUES(1,1,1,3);

    INSERT INTO FEEDBACK VALUES(2,1,1,5);

    INSERT INTO FEEDBACK VALUES(3,1,1,5);

    INSERT INTO FEEDBACK VALUES(4,1,1,2);

    INSERT INTO FEEDBACK VALUES(1,6,1,5);

    INSERT INTO FEEDBACK VALUES(2,6,1,5);

    INSERT INTO FEEDBACK VALUES(3,6,1,3);

    INSERT INTO FEEDBACK VALUES(4,6,1,5);

    INSERT INTO FEEDBACK VALUES(1,1,2,4);

    INSERT INTO FEEDBACK VALUES(2,1,2,4);

   INSERT INTO FEEDBACK VALUES(3,1,2,5);
```

```
INSERT INTO FEEDBACK VALUES(4,1,2,5);

    INSERT INTO FEEDBACK VALUES(1,2,3,5);

    INSERT INTO FEEDBACK VALUES(2,2,3,5);

    INSERT INTO FEEDBACK VALUES(3,2,3,1);

    INSERT INTO FEEDBACK VALUES(1,1,3,5);

    INSERT INTO FEEDBACK VALUES(2,1,3,5);

    INSERT INTO FEEDBACK VALUES(3,1,3,5);

    INSERT INTO FEEDBACK VALUES(4,1,3,5);

    INSERT INTO VALUTAZIONI VALUES(1,1,'Bel corso');

    INSERT INTO VALUTAZIONI VALUES(2,1,'Troppo semplice');

    INSERT INTO VALUTAZIONI VALUES(5,1,'Toccante');

    INSERT INTO VALUTAZIONI VALUES(2,2,'Docente molto preparato');

    INSERT INTO VALUTAZIONI VALUES(1,3,'SELECT * FROM Corsi');

    INSERT INTO VALUTAZIONI VALUES(2,3,'printf("Bel corso\n")');

    INSERT INTO UNITADIAPPRENDIMENTO(Numero,Numero_lezione,ID_Corso,ArgVideo)
VALUES(1,1,1,'Introduzione al corso');

    INSERT INTO UNITADIAPPRENDIMENTO(Numero,Numero_lezione,ID_Corso,ArgTesto)
VALUES(2,1,1,'La storia dei DataBase');

    INSERT INTO UNITADIAPPRENDIMENTO(Numero,Numero_lezione,ID_Corso,ArgVideo)
VALUES(1,2,1,'Il modello logico');

    INSERT INTO UNITADIAPPRENDIMENTO(Numero,Numero_lezione,ID_Corso,ArgVideo)
VALUES(2,2,1,'Traduzione dello schema ER');

    INSERT INTO UNITADIAPPRENDIMENTO(Numero,Numero_lezione,ID_Corso,ArgTesto)
VALUES(1,3,1,'SELECT');

    INSERT INTO UNITADIAPPRENDIMENTO(Numero,Numero_lezione,ID_Corso,ArgTesto)
VALUES(2,3,1,'CREATE TABLE');

    INSERT INTO UNITADIAPPRENDIMENTO(Numero,Numero_lezione,ID_Corso,ArgTesto)
VALUES(3,3,1,'ALTER TABLE');

    dbms_output.enable;
    dbms_output.put_line('###DataBase popolato con successo###');
    EXCEPTION WHEN OTHERS THEN dbms_output.put_line('###DataBase giÃ  popolato!###');
END;



PROCEDURE SVUOTAMENTO
IS
BEGIN
    DELETE FROM FEEDBACK;
    DELETE FROM ISCRIZIONI;
    DELETE FROM offerteformative_corsi;
    DELETE FROM unitadiapprendimento;
    DELETE FROM valutazioni;
    DELETE FROM lezioni;
    DELETE FROM studenti;
    DELETE FROM offerteformative;
    DELETE FROM corsi;
    DELETE FROM docenti;

    dbms_output.enable;
    dbms_output.put_line('###DataBase svuotato con successo###');
END;
END Sito;
```

3.6 TESTING DELLE QUERY

```
CREATE OR REPLACE PACKAGE query
IS

PROCEDURE numero(numero_corso SMALLINT);
PROCEDURE percentuale;

END query;
```

```
CREATE OR REPLACE PACKAGE BODY query
IS

PROCEDURE numero(numero_corso SMALLINT) IS



CURSOR cursore IS select l.Numero, l.Argomento, count(u.argvideo) as
numero_video, count(u.argtesto) as numero_testo
                from lezioni l join unitadiapprendimento u on
u.NUMERO_LEZIONE = l.Numero
                where l.ID_CORSO = numero_corso
                group by l.Numero,l.argomento;


Vettore cursore%ROWTYPE;

BEGIN
dbms_output.put_line('NUMERO DI ELEMENTI VIDEO E TESTUALI PER OGNI
LEZIONE DI UN CERTO CORSO');

FOR vettore in cursore
LOOP
dbms_output.put_line(vettore.numero ||' '|| vettore.argomento ||' '||
vettore.numero_video ||' '|| vettore.numero_testo);
END LOOP;

END;


PROCEDURE percentuale IS

CURSOR cursore IS select s.Nome,s.Cognome, count(v.ID_CORSO) /
count(c.ID) * 100 as Percentuale
                from ((((studenti s join iscrizioni i on s.ID =
i.ID_Studente)
                join offerteformative o on (o.ID = i.ID_Offerta))
                join offerteformative_corsi oc on o.ID =
(oc.ID_Offerta))
                join corsi c on (c.ID = oc.ID_Corso))
                left join valutazioni v on (v.ID_Corso = c.ID AND
v.ID_Studente = s.ID)
                group by (s.ID,s.nome,s.cognome);
```

```
Vettore cursore%ROWTYPE;

BEGIN

dbms_output.put_line('RAPPORTO, PER OGNI STUDENTE, DEI CORSI RECENSITI
DIVISO QUELLI ACQUISTATI');

FOR Vettore in cursore
LOOP
dbms_output.put_line(Vettore.nome || ' ' || Vettore.cognome || ' ' ||
Vettore.percentuale || '%' );
END LOOP;

END;


END query;
```

RISULTATI DELLE QUERY:

1.

	NOME	COGNOME	NOME_CORSO
1	Angelo	Chianese	Basi di Dati
2	Angelo	Chianese	Fondamenti di Informatica
3	Vincenzo	Moscato	Spionaggio industriale
4	Vincenzo	Moscato	Intreccio cesti in vimini
5	Niels	Kowalzig	Algebra e Geometria
6	Francesco	Chiacchio	Analisi 2

2.

	NUMERO_LEZIONE	ARGOMENTO	NUMERO_VIDEO	NUMERO_TESTO
1	1	Modello ER	1	1
2	3	SQL	0	3
3	2	Modello Logico	2	0

3

	ID	NOME	MEDIA
1	6	Analisi 2	4,5
2	1	Basi di Dati	4,41...

4

	NOME	COGNOME	NUMERO_CORSI
1	Vincenzo	Moscato	2
2	Angelo	Chianese	2

5

	NOME	COGNOME	PERCENTUALE
1	Alessio	Paesano	100
2	Emanuele	Riccio	75
3	Francesco	Mazza	50