# **Etivity Davide De Angelis**

**Caso di studio scelto**: Un sistema integrato per l’amministrazione di studenti, docenti, corsi, esami e attività di ricerca in un ateneo universitario.

## **Descrizione del caso di studio scelto:**

Il sistema gestisce:

- **Studenti:** iscrizione a corsi, registrazione voti, gestione tesi, tutoraggio.

- **Docenti:** assegnazione corsi, supervisione tesi, coordinamento dipartimenti, gestione laboratori.

- **Interazioni:** esami (studente-docente), progetti di ricerca (N:N tra studenti/docenti), assegnazione risorse (aule, laboratori).

- **Vincoli:** rispetto del range voti (CFU 18-30), controllo per laurea, disponibilità docenti.

## **Analisi e raccolta dei requisiti:**

* Registra studenti, docenti, corsi e dipartimenti, collegando iscrizioni, insegnamenti e relazioni organizzative. Traccia l’affiliazione dei docenti ai dipartimenti e l’offerta formativa delle facoltà. Semplifica il monitoraggio delle attività didattiche e amministrative.
* Il sistema gestisce studenti, docenti, corsi, dipartimenti e facoltà. Gli studenti possono iscriversi a più corsi e partecipare alle attività didattiche, mentre i docenti si occupano dell’insegnamento e del supporto alle tesi. Ogni corso è collegato a una facoltà, che interagisce con i dipartimenti. I docenti sono assegnati a un dipartimento, con un responsabile designato, e valutano gli studenti in base ai risultati nei corsi. Il modello facilita il monitoraggio delle relazioni accademiche e delle attività amministrative, semplificando l’organizzazione delle informazioni universitarie.
* Il database rappresenta un sistema universitario progettato per gestire studenti, docenti, corsi, dipartimenti e facoltà. Gli studenti possono iscriversi a più corsi e partecipare alle lezioni, mentre i docenti insegnano i corsi e supervisionano le tesi degli studenti. Ogni corso può avere prerequisiti e appartiene a una facoltà, che collabora con diversi dipartimenti. I docenti sono affiliati a un dipartimento e uno di loro ne è il direttore. Inoltre, i docenti valutano gli studenti in base alle loro prestazioni nei corsi. Questo modello consente di tracciare relazioni accademiche e amministrative, garantendo una gestione efficiente delle informazioni universitarie.

# **Glossario dei Termini**

|  |  |  |  |
| --- | --- | --- | --- |
| **Termine** | **Descrizione** | **Sinonimi** | **Collegamenti** |
| Facoltà | Unità accademica che offre corsi universitari. | Dipartimento | Corso, Studente, Docente |
| Dipartimento | Sezione amministrativa e accademica di una facoltà. | Sezione | Facoltà, Docente |
| Docente | Persona che insegna i corsi universitari. | Professore, Insegnante | Corso, Dipartimento, Studente |
| Assistente | Supporta i docenti nelle attività accademiche e didattiche. | Collaboratore | Docente |
| Studente | Persona iscritta ai corsi di studio universitari. | Matricola, Alunno | Corso, Facoltà, Test |
| Corso | Attività formativa offerta da una facoltà. | Insegnamento, Lezione | Facoltà, Docente, Studente |
| Test | Prova valutativa che uno studente sostiene per un corso. | Esame, Verifica | Studente, Docente |
| Prerequisiti | Requisiti necessari per accedere a un corso. | Requisiti | Corso |

## **Strutturazione dei requisiti in gruppi di frasi omogenee:**

**Studente**  
Identificato da una *Matricola* univoca, lo Studente include informazioni personali come *Nome*, *Cognome*, *DataNascita*, *Email* e l’*AnnoIscrizione*. Svolge un ruolo centrale nell’interagire con i Corsi: si iscrive a più corsi e partecipa alle lezioni e ha una relazione con un Docente per la *Supervisione Tesi*. La sua carriera accademica è inoltre valutata dai docenti tramite la relazione *Valutazione*.

**Docente**  
Riconosciuto da un *ID\_Docente*, il Docente include informazioni come *Nome*, *Cognome*, *Email* e *Ruolo*. È responsabile dell’insegnamento di più, appartiene a un Dipartimento e può diventare *Direttore* di quest’ultimo. Supervisiona le tesi degli Studenti e valuta il loro rendimento, agendo come punto di riferimento didattico e organizzativo.

**Corso**  
Definito da un *CodiceCorso*, il Corso include *Titolo*, *CFU*, *Semestre* e *Aula*. È offerto da una Facoltà e insegnato da un singolo Docente. Gli Studenti vi si iscrivono e partecipano alle lezioni.

**Dipartimento**  
Identificato dal *CodiceDipartimento*, contiene *Nome* ed *Edificio*. Organizza i ed è diretto da un Docente. Collabora con le Facoltà tramite la tabella *Collaborazioni* facilitando sinergie tra entità per l’offerta formativa.

**Facoltà**  
Caratterizzata da *CodiceFacoltà*, *Nome* e *Sede*, la Facoltà coordina l’offerta dei Corsi e gestisce collaborazioni con i Dipartimenti. Funge da hub organizzativo, collegando risorse didattiche e strutturali per erogare i programmi accademici.

## **Normalizzazione:**

Nel contesto dello schema concettuale, il processo di normalizzazione non risulta applicabile in quanto non si riscontrano ridondanze significative nei dati. La struttura attuale delle entità e delle loro relazioni è già conforme ai principi della modellazione relazionale ottimizzata, garantendo coerenza e minimizzando la duplicazione delle informazioni, pertanto, non vi è la necessità di ulteriori decomposizioni o ristrutturazioni per il raggiungimento delle forme normali.

## **Interrogazioni in algebra relazionale:**

#### SELEZIONI

* π∗​(assistente)
* π∗​(docente)
* π∗​(corso)
* π∗​(dipartimento)
* π∗​(studente)

#### PROIEZIONI

* πNome,Email(Assistente)
* πNome,Cognome,CodiceDipartimento(Docente)
* πTitolo,CFU​(Corso)
* πEdificio,Nome​(Dipartimento)
* πMatricola,Cognome,AnnoIscrizione​(Studente)

#### RIDENOMINAZIONI (in relazione alle proiezioni precedenti)

* ρ("Nome", "e-Mail", assistente)
* ρ("Nome Docente", "Cognome Docente", "Codice Dipartimento", docente)
* ρ("Titolo del Corso", "Crediti", corso)
* ρ("Nome Edificio", "Facoltà", dipartimento)
* ρ("Matricola Studente", "Cognome Studente", "Anno Iscrizione", studente)

#### JOIN

* docente|X|assistente
* docente|X|leftassistente
* docente|X|rightassistente
* docente|X|fullassistente
* docente|X|id\_assistente=id\_assistenteassistente

## **Modello Relazionale:**

* assistente (id\_assistente [PK], Nome, Cognome, Email)
* corso (idcorso [PK], Titolo, CFU, Semestre, Aula, idDocente [FK], idFacolta [FK])
* dipartimento (codiceDipartimento [PK], Edificio, Nome)
* docente (idDocente [PK], Nome, Cognome, Email, id\_assistente [FK], codiceDipartimento[FK])
* facolta (idfacolta [PK], Nome, Sede)
* studente (matricola [PK], Nome, Cognome, DataNascita, Email, AnnoIscrizione)
* tesi (idTesi [PK], idDocente [FK], matricola [FK], voto)

## **CRUD:**

#### CREATE

meta = MetaData()  
  
  
  
facolta = Table( 'facolta', meta, Column ('idfacolta', Integer, primary\_key = True, autoincrement = True), Column ('Nome', String(100)), Column('Sede', String(255)), )  
  
  
  
meta.create\_all(engine)

#### READ

connSelect = engine.connect()  
  
selezione = facolta.select()  
  
selezionex = "facolta.select()"  
  
risultatoSelect = connSelect.execute(selezione)  
  
riga = risultatoSelect.fetchall()  
  
rigaex = "risultatoSelect.fetchall()"  
  
print(riga)  
  
print('\v')  
  
time.sleep(1)  
  
print(f'Risultato della query {selezionex} utilizzando {rigaex}')

#### UPDATE

conn4 = engine.connect()  
  
updateRecordFacolta = facolta.update().values({facolta.c.Sede:'Via del Tritone 75'}).\  
where(facolta.c.idfacolta == 15)  
  
  
conn4.execute(updateRecordFacolta)  
conn4.commit()  
  
print('Record aggiornati correttamente' )

#### DELETE

conn2 = engine.connect()  
  
eliminaRecordFacolta = facolta.delete().where(facolta.c.idfacolta == 15)  
  
conn2.execute(eliminaRecordFacolta)  
conn2.commit()  
  
print('Record eliminati' )  
  
time.sleep(3)  
  
print('\v')

## **Interrogazioni MySQL:**

#### Select

SELECT \* FROM studente;

#### Select + Where

SELECT \* FROM docente

WHERE idDocente > 7

#### Select + AS + Join + Order By

SELECT docente.idDocente AS idDocente, docente.Nome, docente.Cognome, assistente.id\_assistente, assistente.Nome AS "Nome Assistente", assistente.Cognome AS "Cognome Assistente"

FROM docente

JOIN assistente ON docente.id\_assistente = assistente.id\_assistente

ORDER BY assistente.id\_assistente ASC

#### Select nidificata

SELECT \*

FROM studente

WHERE Nome = ANY (SELECT Nome FROM docente WHERE Nome = 'Simone')