Laboratorio B
Progetto SeatIn
Simple E-learning at Insubria
Anno accademico 2017/2018

Relazione della base di dati

P. Farina (727825) , D. Muraro (716298), A. Roshka (730662), C. Stroppolo (724781)

Introduzione

Lo scopo che si prefigge questo documento è quello di delineare le varie fasi della progettazione della basi di dati per il progetto Simple E-learning at Insubria, da ora denominato SeatIn, una piattaforma di e-learning nella quale verrà data la possibilità di creare, distribuire, selezionare, amministrare e diffondere gli strumenti utili alla formazione.

Metodologia

La metodologia seguita per la progettazione della base di dati si può riassumere in 4 fasi:

1. Raccolta ed analisi dei requisiti
   Il cui scopo è quello di definire in modo informale le caratteristiche che avrà la base di dati analizzando i requisiti forniti dal committente.
   Ha come risultato un documento in linguaggio naturale di specifica dei requisiti.

2. Progettazione concettuale
   La Progettazione Concettuale di una Base di Dati significa individuare gli oggetti (o entità) che la costituiscono e le relazioni (o operazioni o associazioni) tra un oggetto e l'altro.
   In questa fase si genera una prima rappresentazione formale del contenuto della base di dati a partire dal documento di specifica definito nella prima fase.

3. Progettazione logica
   Durante la fase della progettazione logica si traduce lo schema concettuale prodotto nella fase di progettazione concettuale, nel modello di rappresentazione dei dati adottato dall'ambiente database scelto per la programmazione. La traduzione, per essere formalmente corretta con i software in uso (per utilizzare così il minor numero di spazio e aumentare le velocità di recupero dei dati), porta ad utilizzare la tecnica della normalizzazione che si divide in tre (3) ulteriori livelli: 1ª, 2ª e 3ª forma normale. Il prodotto di questa fase è chiamato schema logico.

4. Progettazione fisica
   In questa fase vengono definiti alcuni dettagli sulla memorizzazione fisica dei dati, come, ad esempio, la definizione di strutture ausiliarie di accesso per la velocizzazione di alcune interrogazioni.

Glossario dei termini

Termine	Descrizione	Sinonimi	Collegamenti
Docente	Utente che tiene e monitora i propri corsi	Insegnante	Studenti, Monitoraggio
Studente	Utente che segue i		Docente, Corso

	corsi e sostiene gli esami		
Amministratore	Utente che amministra i profili degli utenti e le controlla le statistiche		Docente, Corso, Studente,Monitoraggio
Corso	Corso di studio		Docente, Studente, Amministratore,Sezione, Risorsa
Modulo	Ogni corso e' diviso in moduli		Corso, sezione
Sezione	Ogni modulo puo' essere diviso in sezioni		Docente, Studente, Amministratore, Corso, Modulo
Newsletter	La comunicazione attraverso la newsletter		Docente, Studente
Monitoraggio	Servizi di analisi per docenti titolari di un corso e amministratori		Studente, Corso, Amministratore

1. Analisi dei requisiti

1.1 Utenti

Per quanto riguarda la gestione degli utenti la base di dati permetterà di memorizzare le seguenti tipologie di utenti con le relative mansioni:

- Amministratore
  - ❖ abilita un profilo utente: studente, docente o amministratore
  - ❖ crea un'istanza di un corso
  - ❖ assegna docenti ad un corso
  - ❖ analizza l'uso della piattaforma
  - ❖ registra docenti ed altri amministratori
  - ❖ attribuisce uno o più docenti (precedentemente registrati) ad un corso

- Docente
  - ❖ gestisce la pubblicazione di materiale didattico e informazioni riguardante i corsi di cui è titolare

        - ❖ comunica con gli studenti dei suoi corsi attraverso newsletter
        - ❖ analizza l'accesso alle pagine del corso

- Studente
    - ❖ una volta iscritto ad un corso può visualizzare tutte le informazioni pubblicate
    - ❖ scarica il materiale didattico
    - ❖ invia email al docente

Si evidenzia che ogni tipologia di utente ha in comune i seguenti campi:

- numero di matricola
- cognome
- nome
- email
- codice di attivazione
- password

Per i docenti e gli amministratori abbiamo un ulteriore campo che riguarderà il dipartimento o la struttura di riferimento. Mentre per gli studenti dovranno inoltre essere presenti:

- il corso di laurea
- l'anno di immatricolazione
- lo stato della carriera dello studente (codifica l'iscrizione al I / II / III anno in corso / fuori corso)

1.2 Corsi

Il punto focale dell'applicazione è il catalogo dei corsi. Ogni corso è caratterizzato da

- codice identificativo
- nome
- anno accademico di attivazione
- corso di laurea di riferimento
- una breve descrizione
- contenuti strutturati in sezioni

Ogni corso a sua volta può essere diviso in sezioni, le quali sono caratterizzate da:

- titolo
- descrizione (opzionale)
- numero variabile (0..*) di risorse, ad esempio file con estensione .txt, .ppt, .pdf e jpg eventualmente organizzati in cartelle che rappresentano le risorse didattiche del corso
- numero variabile (0..*) di sottosezioni ognuna con possibili risorse

Ogni sezione ha associato un livello di visibilità che può essere privato, cioè visibile unicamente dal docente titolare, o pubblico, cioè accessibile da ogni utente (opzione di default), e un intervallo temporale di visibilità (opzionale), che specifica una restrizione di visibilità di un contenuto pubblico.

Se l’utente è uno studente o docente e non è iscritto al corso può solamente visualizzare: il titolo del corso, i docenti assegnati e una breve descrizione del corso.
I privilegi di scrittura sui contenuti di un corso sono concessi unicamente ai docenti titolari e agli amministratori.

1.3 Piano di studi

L'elenco degli esami di profitto che lo studente è tenuto a sostenere è basato sul piano di studi definito dallo stesso studente. Da notare che lo studente può iscriversi solamente a corsi presenti nel piano di studi.

1.4 Comunicazione

Un docente titolare o un amministratore può decidere di notificare avvisi per email a tutti gli studenti, o a una selezione di studenti, iscritti al determinato corso. Il servizio può anche essere usato contestualmente alla pubblicazione di nuovi contenuti del corso, per notificare a tutti i partecipanti un aggiornamento rilevante.
Deve essere offerta la possibilità all’utente di selezionare come oggetto del messaggio il titolo di una sezione del corso.

1.5 Monitoraggio

Si deve offrire un servizio di analisi delle statistiche per i docenti titolari di un corso e gli amministratori. Le operazioni più comuni sono qui elencate:

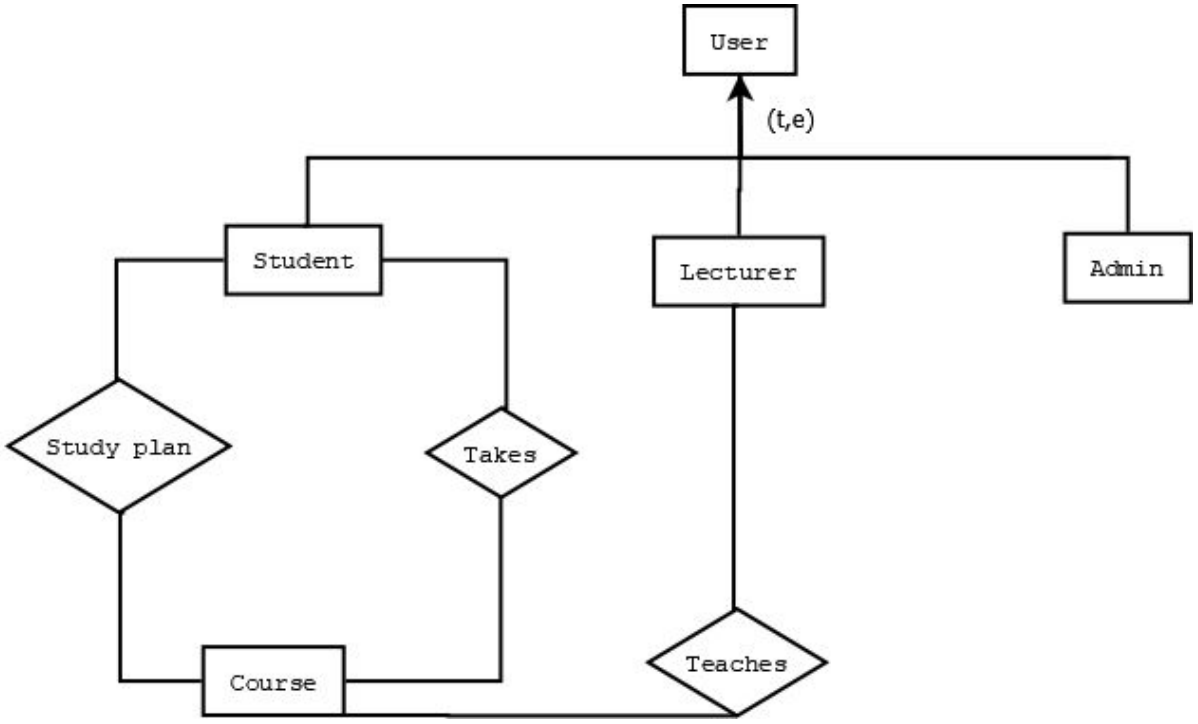
1. Mostrare il numero complessivo di utenti connessi che stanno visualizzando/interagendo con i contenuti del corso
2. Mostrare il numero complessivo di utenti che hanno effettuato il download di una o più risorse in intervalli temporali dati
3. Derivare il tempo medio di connessione degli studenti alle pagine del corso

Gli amministratori possono effettuare delle analisi di utilizzo, in particolare:
1. Mostrare il numero complessivo di utenti contemporaneamente connessi
2. Mostrare il numero complessivo accessi per corso in una fascia temporale
3. Derivare il tempo medio di connessione degli studenti per ogni corso
4. Derivare il numero complessivo di download per ogni corso

2. Progettazione concettuale: ER Diagram

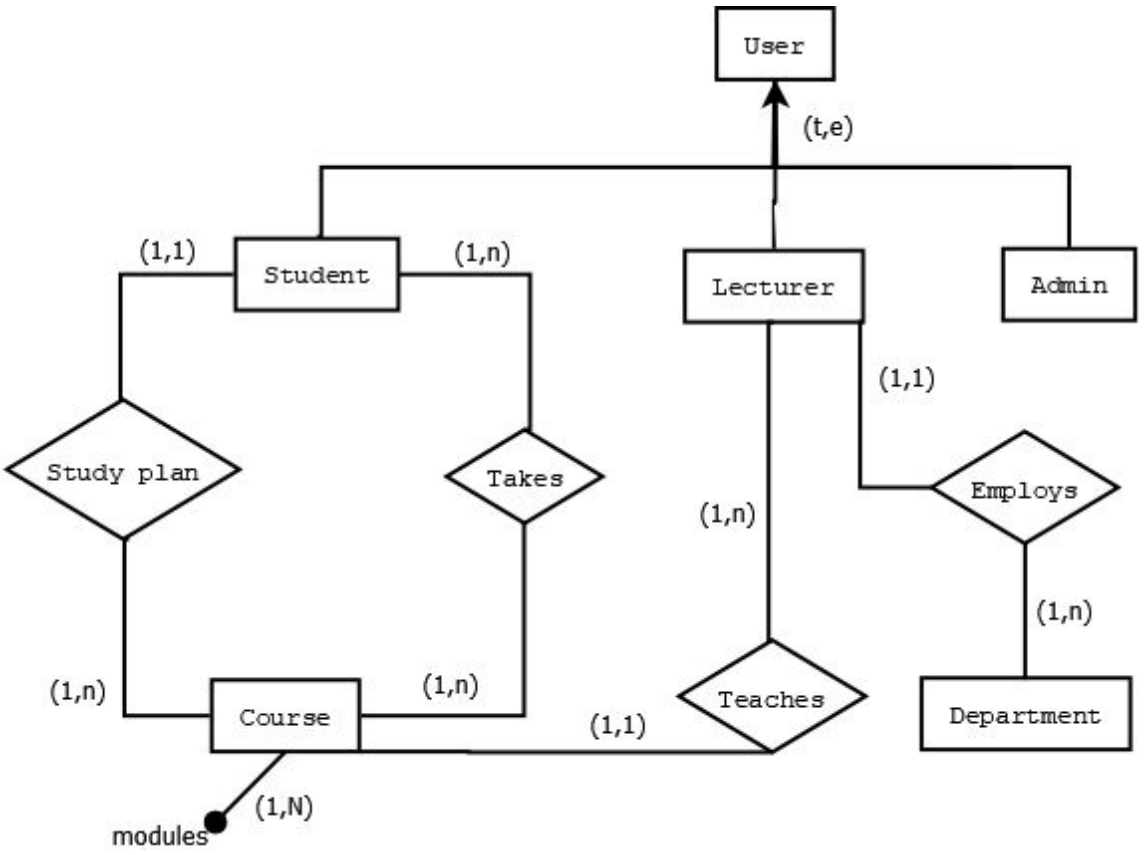
2.1 Creazione di uno schema scheletro o draft

Dopo una prima analisi vengono individuate le seguenti tipologie di utenti: i docenti, gli studenti e gli admin, questi vengono raggruppati in una gerarchia con entità generica User di tipo totale in quanto ogni istanza dell'entità padre (User) deve far parte di almeno un'entità figlia e esclusiva in quanto ogni istanza dell'entità padre deve far parte di una sola delle entità figlie (Fig 2.1.1.). Tale gerarchia costituisce dei vincoli che saranno definiti in seguito.



Schema scheletro - Fig.2.1.1

Dalla lettura delle specifiche sappiamo che l'università è composta da un numero di dipartimenti, ogni dipartimento offre dei corsi di laurea composti a loro volta da corsi suddivisi in moduli che sono gestiti da più professori che insegnano a svariati studenti (Fig. 2.1.2), le suddette entità dovranno essere memorizzate. Inoltre vi è la necessità di fornire delle statistiche di accesso che dovranno anch'esse essere memorizzate, ma verranno integrate nel paragrafo successivo essendo fini a loro stesse.



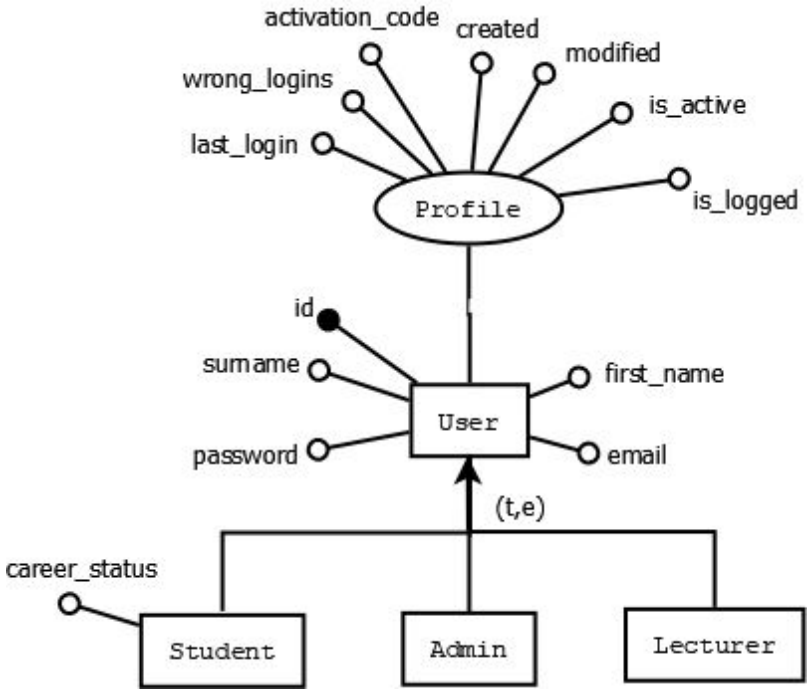
Schema scheletro - Fig.2.1.2

Si noti l'aggiunta dell'associazione Employs che è univoca da Lecture a Department, ovvero a un oggetto della classe Lecturer è associato un solo oggetto della classe Department; in tal modo si modella il fatto che un docente possa essere impiegato da un solo dipartimento. Inoltre, Employs è multivalore da Department a Lecturer, ovvero a un oggetto della classe Department possono essere associati più oggetti della classe Lecturer; in tal modo si modella il fatto che un dipartimento impiega molti docenti. Tra l'entità Student e l'entità Course esiste un'associazione individuata dal verbo Takes caratterizzata dal fatto che uno studente può frequentare un qualsiasi numero di corsi presente nel suo piano di studi ed un corso è frequentato da almeno uno studente. Inoltre sappiamo che un docente può insegnare uno o più corsi. Viene inoltre aggiunto un attributo multiplo modules in quanto ogni corso è composto da più moduli (Fig 2.1.2). Le associazioni e gli attributi sopra menzionate saranno sviluppate nelle fasi successive della relazione e costituiscono una parte fondamentale nella creazione della base di dati.

2.2 Schemi parziali

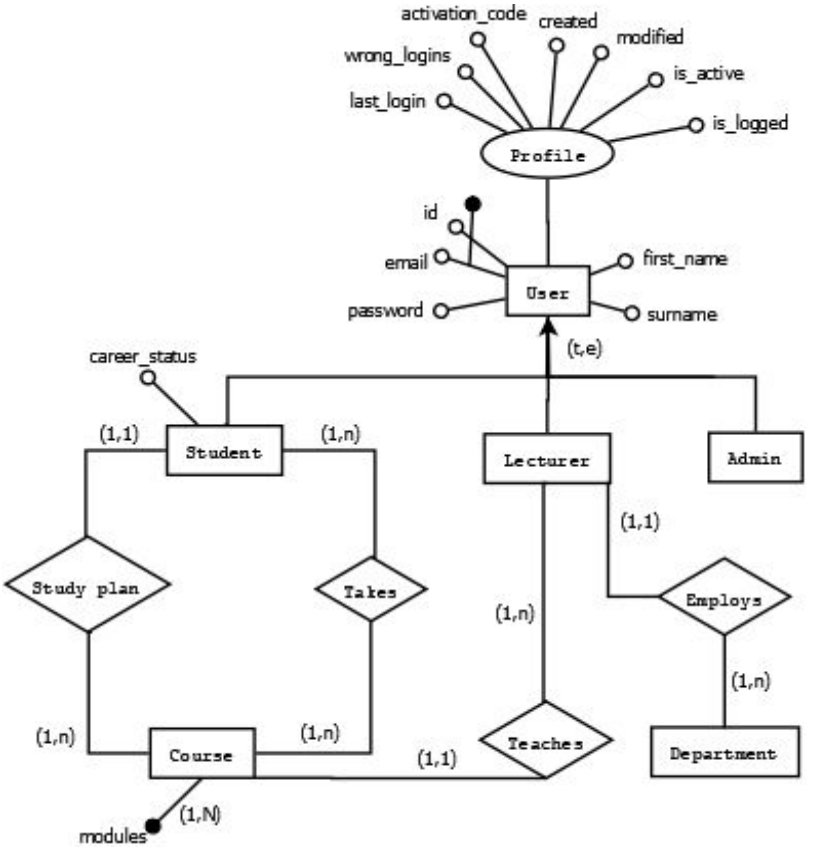
Vista la complessità delle entità riguardanti le varie tipologie di utenti (amministratori, studenti e docenti) e delle loro generalizzazioni prima di procedere con la definizione dello schema concettuale completo abbiamo preferito creare degli schemi parziali considerando anche i possibili attributi. Sono stati usati degli attributi composti perché ad esempio a livello applicativo le informazioni di registrazione e attivazione degli utenti saranno utilizzate esclusivamente durante le suddette fasi oppure durante la modifica del profilo, per tale motivo si può proattivamente pensare di organizzarli in fase di ristrutturazione in un'entità

separata da quella degli utenti in modo da usare queste informazioni solo nei casi di necessità ottimizzando le future prestazioni alleggerendo le queries. Questa suddivisione si evidenzia in Fig. 2.2.1 con l'aggiunta di Profile e dei relativi attributi, e sempre in Fig. 2.2.1 è ben distinguibile la gerarchia di generalizzazione di tipo totale esclusiva (t,e) dove ogni istanza di User è istanza di almeno un'entità tra Student, Admin o Lecturer.



Schema utenti - Fig. 2.2.1

In Fig 2.2.2 da notare l'aggiunta dell'associazione tra le entità Student e Course chiamata Study plan attraverso la quale ogni Student può creare uno e un solo Study plan (cardinalità 1,1 obbligatoria, una sola volta) nella quale sono presenti uno o più Course (1,n, obbligatoria, almeno una volta). Uno Study plan è composto da uno o più corsi (1,n), simile l'aggiunta dell'associazione tra Lecturer e Department dove ogni Lecturer fa parte di uno e un solo Department (cardinalità 1,1) e un Department è composto da uno o più Lecturer (1,n).

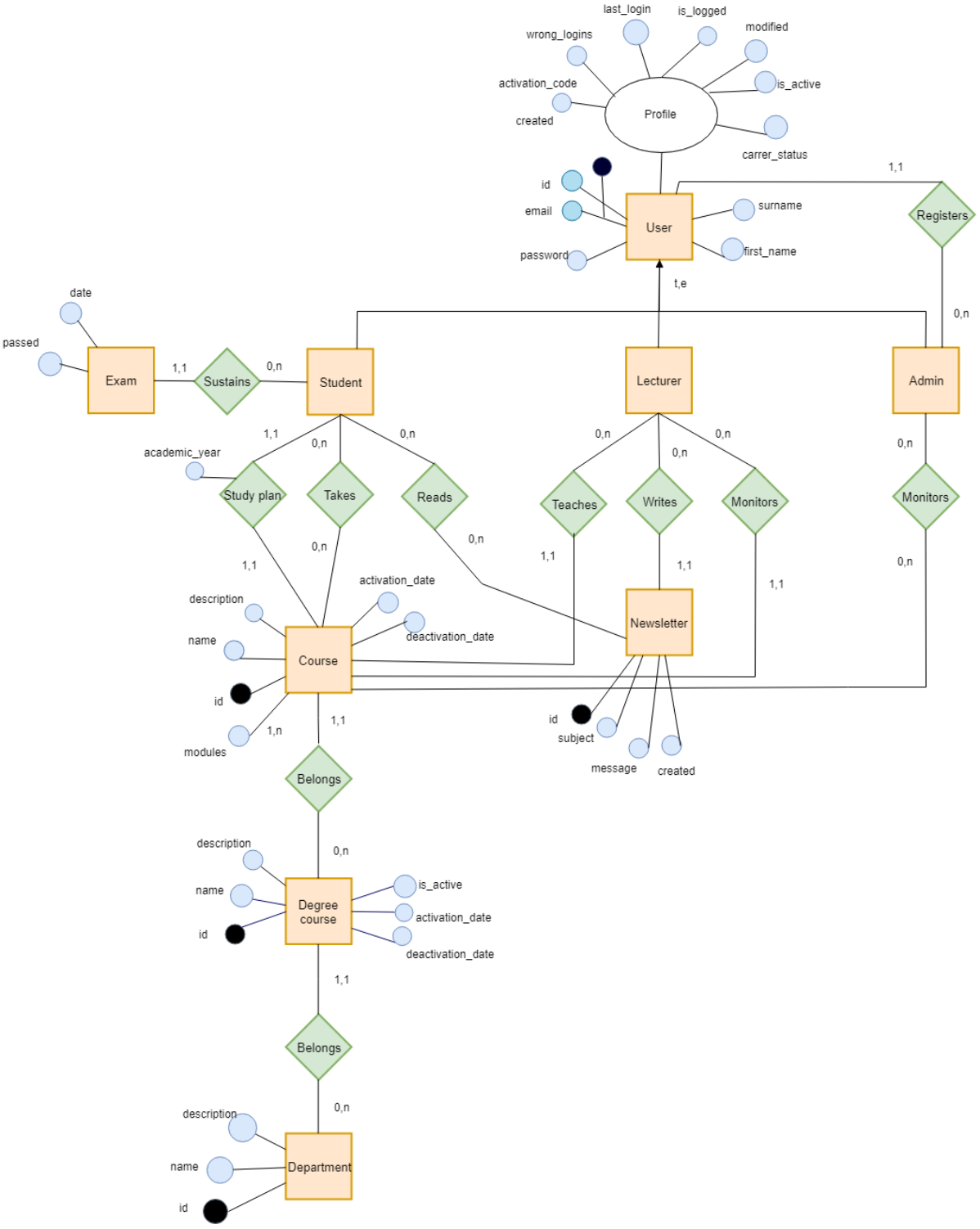


Schema student, lecturer e admin - Fig. 2.2.2

In Fig. 2.2.2 definiamo fin da ora la creazione di una chiave primaria per l'entità User identificata dagli attributi id e email che sarà utilizzata per l'ottimizzazione delle queries di estrazione per i dati utente. Viene aggiunto un attributo multivalore chiamato modules per l'entità Course con cardinalità (1,n) in quanto sappiamo che ogni corso è composto da più moduli, un modulo può avere come padre un altro modulo, lo stesso modulo è composto da più risorse che può avere più files, questa struttura sarà rielaborata nelle fase successiva. Chiaramente l'ottimale definizione delle suddette cardinalità sono la base fondamentale per la fase logica ovvero la creazione fisica delle base di dati modellata in base al RDBMS scelto che è PostgreSQL.

2.3 Schema concettuale completo

Lo schema concettuale completo è stato implementato usando un sito di condivisione di grafici e diagrammi chiamato draw.io, in tal modo i partecipanti al progetto hanno collaborato alla stesura del grafico in Fig 2.3.1. Il link fornito nella legenda dell'immagine è apribile usando draw.io da chiunque ne sia in possesso.
I vari schemi parziali sono stati unificati nello schema concettuale completo in Fig 2.3.1 dove vengono identificate le tutte le entità e le associazioni che fungeranno da input per la successiva e ultima fase di progettazione, ovvero quella logica.



Schema concettuale completo Fig. 2.3.1

https://drive.google.com/file/d/1gNZCbDtWuEi6XsgMRv-krBd4khyNcJXE/view?usp=sharing

Nella definizione dello schema concettuale completo vengono anche implementate a livello strutturale le richieste opzionali relative all'intervallo temporale di visibilità nelle quali si specifica una restrizione di visibilità di un contenuto pubblico. Una possibile gestione delle visibilità dei contenuti, ad esempio la propagazione del criterio di visibilità di una risorsa o di un modulo è possibile a livello di RDBMS usando i triggers che in maniera automatica in coincidenza di un evento come ad esempio il cambio della visibilità di una tupla in una

tabella a cascata possono inserire, cambiare o eliminare altri oggetti mantenendo l'integrità della base di dati. Per questa prima versione si e' preferito minimizzare l'implementazione dei triggers, ciò facendo si favorisce la portabilità della base di dati.

2.4 Dizionari

2.4.1 Entità

Nome	**Descrizione**	**Attributi**	**Identificatori**
Department	Dipartimento dell'università	id: int name: string description: string created: timestamp modified: timestamp	id
Degree course	Lista dei corsi di laurea	id: int name: string description: string activation_date: timestamp deactivation_date: timestamp is_active: boolean	id
Course	Lista dei corsi	id: int name: string description: string activation_date: timestamp deactivation_date: timestamp is_active: boolean degree_course_id: int	id
Module	Lista dei moduli che compongono un corso	id: int title: string description: string active_from: timestamp active_to: timestamp parent_id: int course_id: int	id
Resource	Lista delle risorse che compongono un modulo	id:int title: string description: string	id
User	Tabella degli utenti (campi comuni)	id: int surname: string first_name: string password: string	id, email

		email: string	
Admin	Utenti di tipo Admin		
Lecturer	Utenti di tipo Lecturer		
Student	Utenti di tipo Student		
Newsletter	Lista delle newsletters che un professore può inviare	id: int subject: string message: string user_id: int course_id: int created: timestamp modified: timestamp	id
Exam	Esame che uno studente può sostenere	study_plan_id,passed, date	

2.4.2 Associazioni

Nome	**Descrizione**	**Entità**
Department Has Lecturer	Ogni dipartimento ha molti professori	Department,Lecturer
Department Has Degree course	Ogni dipartimento ha molti corsi di laurea	Department, Degree course
Degree course Has Course	Ogni corso di laurea ha molti corsi	Degree course, Course
Course Has Module	Ogni corso ha più moduli	Course, Module
Module Has Resource	Ogni modulo ha più risorse	Module, Resource
Resource Has File	Ogni risorsa ha molti file	Resource, File
Admin Registers Users	Ogni Admin può registrare gli utenti	Admin, Lecturer, Student,
Admin Monitors Users	Ogni Admin può monitorare i corsi	Admin, Course
Lecturer Writes Newsletter	Ogni docente può scrivere delle newsletter	Lecturer,Newsletter

Lecturer Teaches Course	Ogni docente può insegnare più corsi	Lecturer, Course
Lecturer Monitors Course	Ogni docente può monitorare più corsi che gestisce	Lecturer, Course
Student Creates Study plan	Ogni studente crea il piano di studio	Student, Study plan
Student Reads Newsletter	Ogni studente legge più newsletter	Student,Newsletter
Student Sustains Exams	Ogni studente può sostenere più esami	Student, Exams

2.6 Elenco delle generalizzazioni

Entità padre	**Entità figlie**	**Tipologia**
Users	Lecturer, Student, Admin	Totale esclusiva

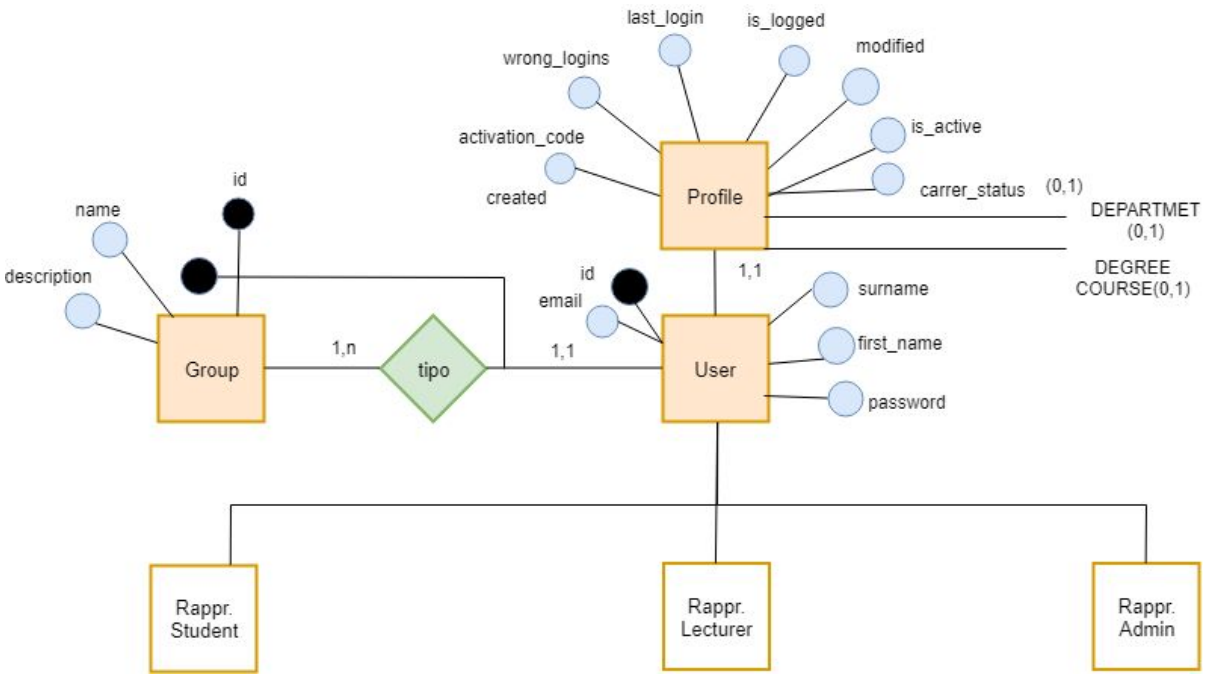
3. Progettazione logica

Nella progettazione logica viene tradotto lo schema concettuale in uno schema logico per il Relational Database Management System (RDBMS) prescelto, nel nostro caso PostgreSQL. La progettazione logica è stata suddivisa in due fasi:

- La ristrutturazione dello schema ER in cui si genera uno schema ER semplificato, ma equivalente a quello di partenza e che sia rappresentabile nel modello relazionale
- La traduzione dello schema ER in cui si traduce lo schema ristrutturato nell'equivalente schema relazionale

3.1 Ristrutturazione dello schema ER

Una prima ristrutturazione riguarda la tipologia degli utenti eliminando le generalizzazioni che non possono essere rappresentabili nel modello relazionale. Abbiamo applicato il cosiddetto collasso verso l'alto riunendo tutte le entità figlie Lecturer, Admin, Student nell'entità padre User, in Fig 3.1.1 per semplicità grafica viene mantenuta una rappresentazione delle entità sopra menzionate. Sostanzialmente le entità figlie vengono eliminate e le proprietà comuni vengono aggiunte all'entità padre User. L'entità Profile collegata a User tramite un'associazione di tipo uno-a-uno definisce le informazioni usate con meno frequenza essendo limitate alle fasi di creazione o di aggiornamento dell'utente. Gli attributi obbligatori delle entità figlie divengono opzionali nell'entità Profile, il tutto indicato dalla cardinalità (0,1) per carrer_status e degree_course (precedente entità Student) e per Department (precedente entità Lecturer) con la conseguenza che si avrà una certa percentuale di valori nulli, ma a livello prestazionale le queries saranno più semplici e veloci essendoci una minore interazione tra le tabelle.



Schema logico Fig.3.1.1

https://drive.google.com/file/d/1wZMXIRWFrNQ0am8z86Py6T64MjiANcU_/view?usp=sharing

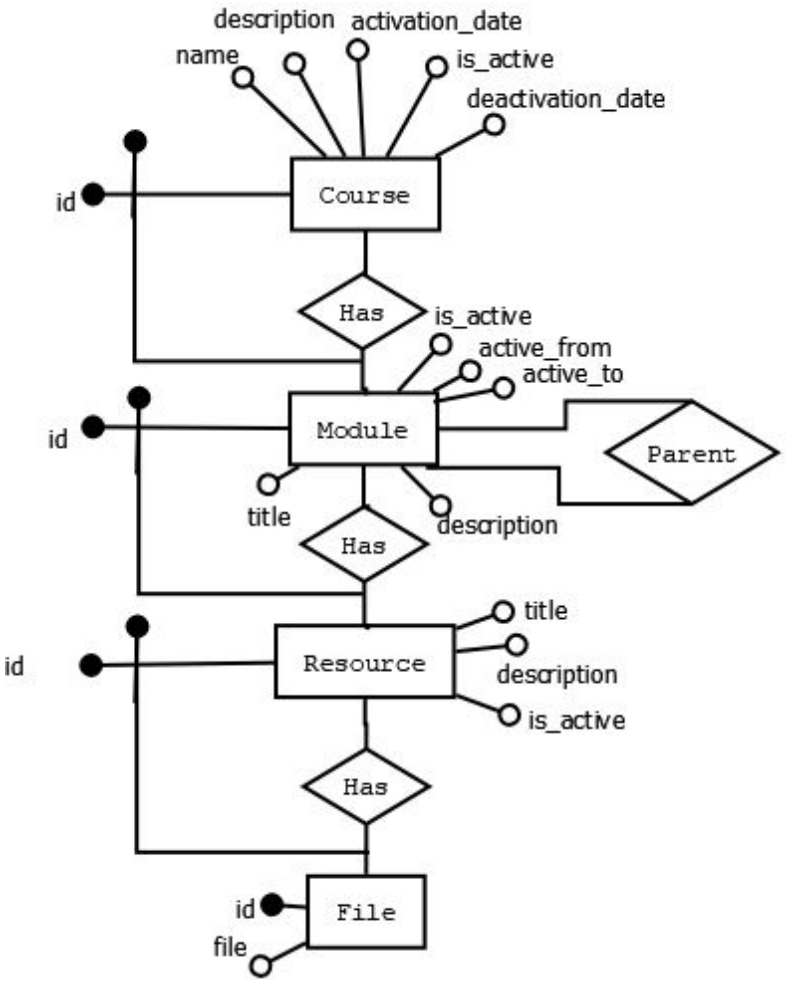
Dopo una prima fase di test dove le entità Lecturer, Admin e Student erano state create come tabelle distinte abbiamo notato che tale suddivisione andava ad appesantire le queries e di conseguenza impattava negativamente sulle performance. Un possibile esempio è rappresentato dalla query che ritorna tutti gli studenti online, nella prima versione era:

```
SELECT surname, first_name, is_logged_in
FROM users
INNER JOIN profiles ON users.id=profiles.user_id
INNER JOIN students ON users.id=students.user_id
WHERE profiles.is_logged_in
ORDER BY surname
```

Mentre con la creazione dell'entità Group (Fig 3.1.1) che specifica se una istanza di User appartiene a una delle sottoentità Lecturer, Admin e Student alla precedente query rimuoviamo la JOIN della tabella students ora rappresentata dal comando WHERE group_id=3, dove l'ID del gruppo student è uguale al numero tre:

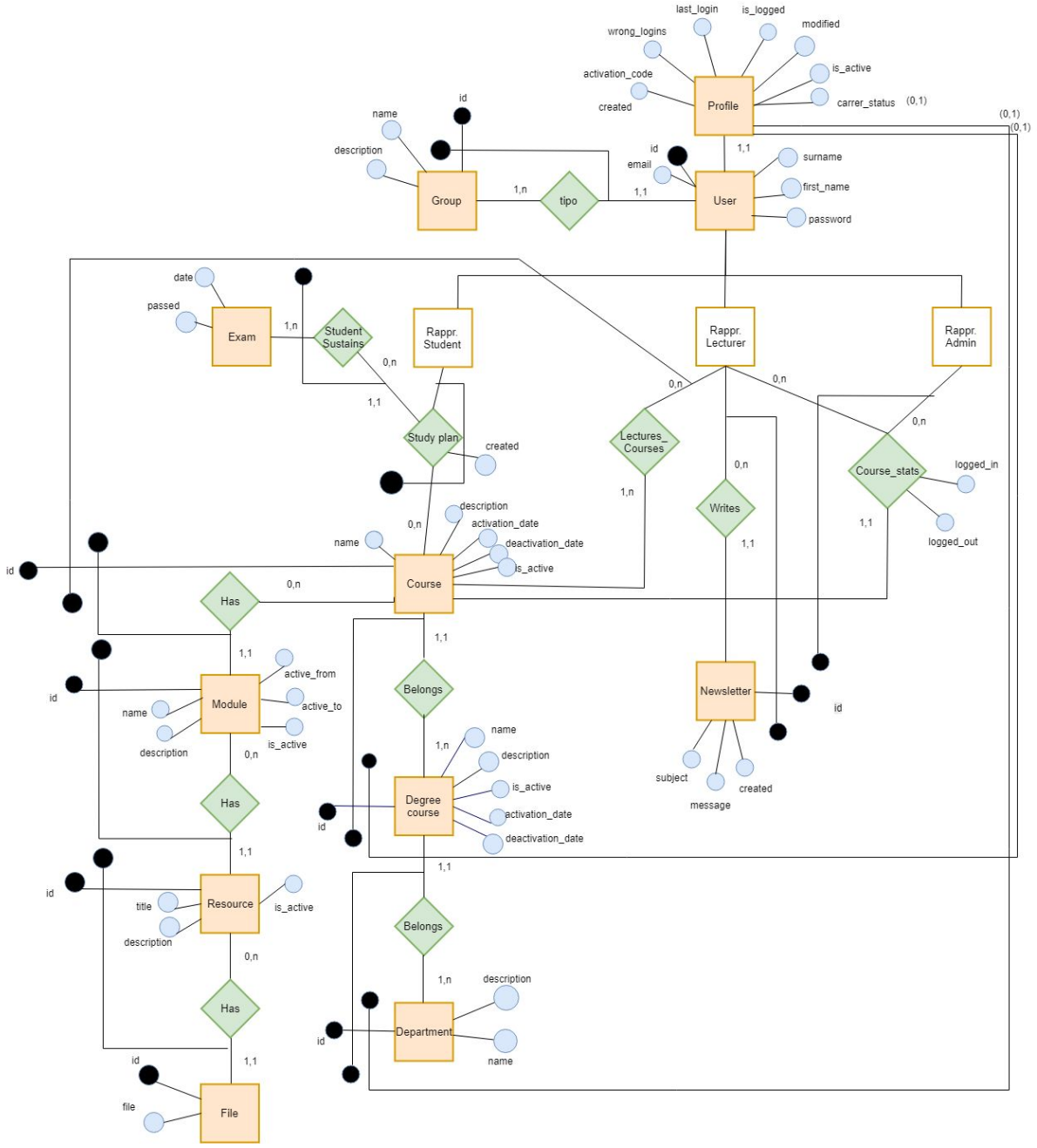
```
SELECT surname, first_name, profiles.is_logged_in
FROM users
INNER JOIN profiles ON users.id=profiles.user_id
WHERE group_id=3 AND profiles.is_logged_in
```

La seconda ristrutturazione è quella riguardante i moduli che compongono un corso, che essendo un attributo multiplo non è rappresentabile nel modello relazionale. Essendo la composizione dei corsi multilivello si è voluto delineare in Fig 3.1.2 una possibile gerarchia dove è possibile vedere le varie associazioni tra le entità: un corso è composto da più moduli, un modulo può avere come padre un altro modulo, lo stesso modulo è composto da più risorse che può avere più files.



Schema composizione dei corsi ristrutturato - Fig. 3.1.2

Il risultato di unificare gli schemi e le ristrutturazioni sopra menzionate è espresso nello schema logico finale in Fig. 3.1.3.



Schema logico finale Fig.3.1.3
https://drive.google.com/file/d/1H_w4p456_s12J8ue0U4ZpJ_Via6-E2y4/view?usp=sharing

3.2 Traduzione dello schema ER

In questa fase ogni entità viene tradotta in una relazione avente lo stesso nome e ogni attributo appartenente all'entità viene tradotto in un attributo della relazione. Ogni identificatore interno diventa chiave primaria per la relazione che viene definita.

3.2.1 Elenco dei vincoli d’integrità intrarelazionale, interrelazionale e di autorizzazione

I vincoli di integrità sono delle proprietà da assicurare ai dati, in termini di correttezza e protezione. Ogni vincolo può essere visto come un predicato cioè una semplice funzione booleana che associa ad ogni istanza il valore vero o falso. Se il predicato assume il valore vero l’istanza soddisfa il vincolo. Se l’istanza della base di dati soddisfa tutti i vincoli, allora essa è corretta: lecita, valida e ammissibile. Nella seguente tabella vengono elencati i vincoli:

1	Gli utenti in seguito a 10 tentativi di autenticazione consecutivi falliti vengono bloccati. Una volta bloccato, il profilo potrà essere riattivato unicamente da un utente amministratore.
2	Lo studente non può registrarsi a un corso dove si è già precedentemente registrato.
3	Il nome del dipartimento deve essere univoco.
4	Il nome del corso di laurea deve essere univoco.
5	Il nome dei gruppi utente deve essere univoco.
6	L’email dell’utente deve essere univoca.
7	Un docente non può insegnare lo stesso corso più volte.
8	Uno studente non può seguire lo stesso corso più volte.
9	Una risorsa in uno specifico modulo deve avere un nome univoco.
10	Il path di ogni file presente in una risorsa deve essere univoco.
11	La data di disattivazione di un corso di laurea deve essere maggiore della data di attivazione.
12	La data di disattivazione di un corso deve essere maggiore della data di attivazione.
13	La data di disattivazione di un modulo deve essere maggiore della data di attivazione.
14	L’ora in cui l’utente effettua il logout deve essere maggiore dell’ora in cui lo stesso utente ha eseguito un login.

3.2.2 Schema relazionale

departments (id, name, description, created, modified)
degree_courses (id, name, description, activation_date, deactivation_date, is_active)
groups (id, name, description)
users (id, surname, first_name, password, email, group_id)
profiles (id, last_login, wrong_logins, activation_code, created, modified, is_active, is_logged_in, department_id, degree_course_id, career_status, user_id)
courses (id, name, description, activation_date, deactivation_date, is_active, degree_course_id)
lectures_courses(id, user_id, course_id)
study_plans (id, user_id, course_id, created)
exams (study_plan_id, passed,date)
modules (id, parent_id, title, description, is_active, active_from, active_to, course_id)
resources (id, title, description, is_active, module_id)
files (id, file, resource_id)
newsletters (id, subject, message, created, modified, course_id, user_id)
course_stats (id, logged_in, logged_out, course_id, user_id)
file_downloads (id, date, file_id, user_id)

4. Note di progetto

Il progetto è stato svolto da un team distribuito in varie locazioni geografiche beneficiando di strumenti orientati al lavoro da remoto e condiviso. I diagrammi concettuali e logici sono stati disegnati usando un sito web chiamato draw.io dove i partecipanti potevano vedere, commentare e proporre i cambiamenti ai vari schemi creati dal responsabile dell'area della base di dati durante le fasi di progettazione, si è anche utilizzato il software Dia per la definizione di alcuni schemi. Questa relazione e il codice SQL sono stati creati e condivisi usando Google Doc un programma gratuito e basato su web di elaborazione e condivisione di testi. L'uso di Slack (https://laboratorio-b.slack.com/) è stato fondamentale per la sincronizzazione delle attività.
La versione di PostgreSQL utilizzata è la 10 rilasciata il 5 ottobre del 2017 nello specifico viene utilizzata la minor release 10.6 avente come final release novembre 2022. Sapendo che il 18 Ottobre del 2018 veniva rilasciata la versione 11 dopo un'analisi delle differenze disponibile all'indirizzo https://www.postgresql.org/about/featurematrix/ abbiamo convenuto di utilizzare la versione 10 per il primo rilascio del progetto SeatIn. PostgreSQL 11 fornisce in generale un miglioramento delle prestazioni specificatamente per quanto riguarda basi di dati con un'alta richiesta computazionale in quanto è stata migliorata la tabella di partizionamento dei dati al livello del Relational Database Management System (RDBMS). La base di dati ha una backward compatibility testata dalla versione 9.6 alla 10.

4.1 Viste

Per le queries più usate specialmente quelle dove sono presenti operazioni di join tra varie tabelle si è pensato di aggiungere delle viste che contengono i risultati di un'interrogazione o un sottoinsieme di righe e/o colonne di una tabella o il risultato di una unione (join) andando indubbiamente a migliorare le performance ricordando che il compilatore SQL esegue di volta in volta delle operazioni ad esempio la validazione della sintassi che per una view viene eseguita solo alla prima esecuzione. Ad esempio è stata creata una view specifica che carica tutti i files presenti in un determinato corso:

```
CREATE OR REPLACE VIEW view_course_files AS
SELECT modules.course_id AS course_id, modules.id, resources.title, files.file
FROM resources
INNER JOIN modules ON resources.module_id=modules.id
INNER JOIN files ON resources.id=files.resource_id
```

E l'utilizzo della stessa diviene:

```
SELECT view_course_files WHERE course_id=<courseID>
```

Lo stesso criterio è applicato nella creazione di una view che carica tutte le risorse per un corso:

```
CREATE OR REPLACE VIEW view_module_resources AS
SELECT modules.course_id AS course_id, resources.id, resources.title
```

FROM resources
INNER JOIN modules ON resources.module_id=modules.id

E l’utilizzo della stessa diviene:
SELECT view_module_resources WHERE course_id=1

5. Riferimenti

"Sistemi di gestione dati" di B. Catania, E. Ferrari e G. Guerrini edizione CittàStudi 1a edizione 2006.
"Mastering PostgreSQL 9.6: A comprehensive guide for PostgreSQL 9.6 developers and administrators" di Hans-Jurgen Schonig.