# Università della Calabria

# Dipartimento di Matematica e Informatica

# Corso di Laurea in Informatica

# a.a. 2022/2023

# Progetto per il Corso di Basi di Dati

# Docente: Prof. P. Rullo

# Laboratorio: Ing. G. Laboccetta, Dott.ssa D. Angilica

# Sistema Informativo

# per la gestione di un

# Istituto Comprensivo

### GRUPPO222

<231536 Saverio Crea>

<220256 Federica De Miglio>

<223489 Giovanni Di Matteo>

<234447 Samuele Siciliano>

<219901 Francesco Vecchio>

## Tematica Progettuale

La progettazione del sistema informatico in esame riguarda la gestione di un istituto comprensivo.

## Raccolta e Analisi dei Requisiti

|  |  |
| --- | --- |
|  | **Requisiti Richiesti** |
| 1 2  3 4 5  6  7 8 9  10 11 12  13 14 15 16 17 | L’istituto comprensivo StudioOnline vuole riorganizzare il proprio database a seguito di alcuni aggiornamenti nella normativa.  Il complesso scolastico include diverse scuole appartenenti ognuna ad uno dei seguenti gradi: scuola dell’infanzia, scuola primaria e scuola secondaria inferiore. Ciascuna scuola è distribuita in diversi plessi (edifici) e ciascun plesso ospita anche più di una scuola.  Le iscrizioni avvengono nel mese di gennaio: di ognuno alunno viene registrata l’anagrafica.  Ogni studente, in un anno scolastico (01/09-19/06), è iscritto ad una delle scuole dell’istituto ma si vuole memorizzare per ciascun ragazzo lo storico delle iscrizioni. All’atto dell’iscrizione, ogni studente è associato ad una classe.  I plessi sono identificati da un indirizzo e sono composti da delle aule. Di ogni aula si conoscono i metri quadri, il numero massimo di studenti ospitabili e la fascia di età degli studenti ospitabili (ad esempio infanti dai 2 ai 3 anni, bambini dai 3 ai 5).  Ogni classe, che ogni anno può cambiare aula, è identificata da un numero (1-4 per l’infanzia, 1-5 per la primaria e 1-3 per le medie) e da una lettera dell’alfabeto (aule di scuole diverse possono avere la stessa coppia). Di ciascun insegnante si conosce, oltre all’anagrafica, ogni anno, il numero di ore lavorative settimanali e le classi a cui è assegnato. Ogni insegnante può insegnare in più classi anche di diverse scuole. |

## Fase preliminare

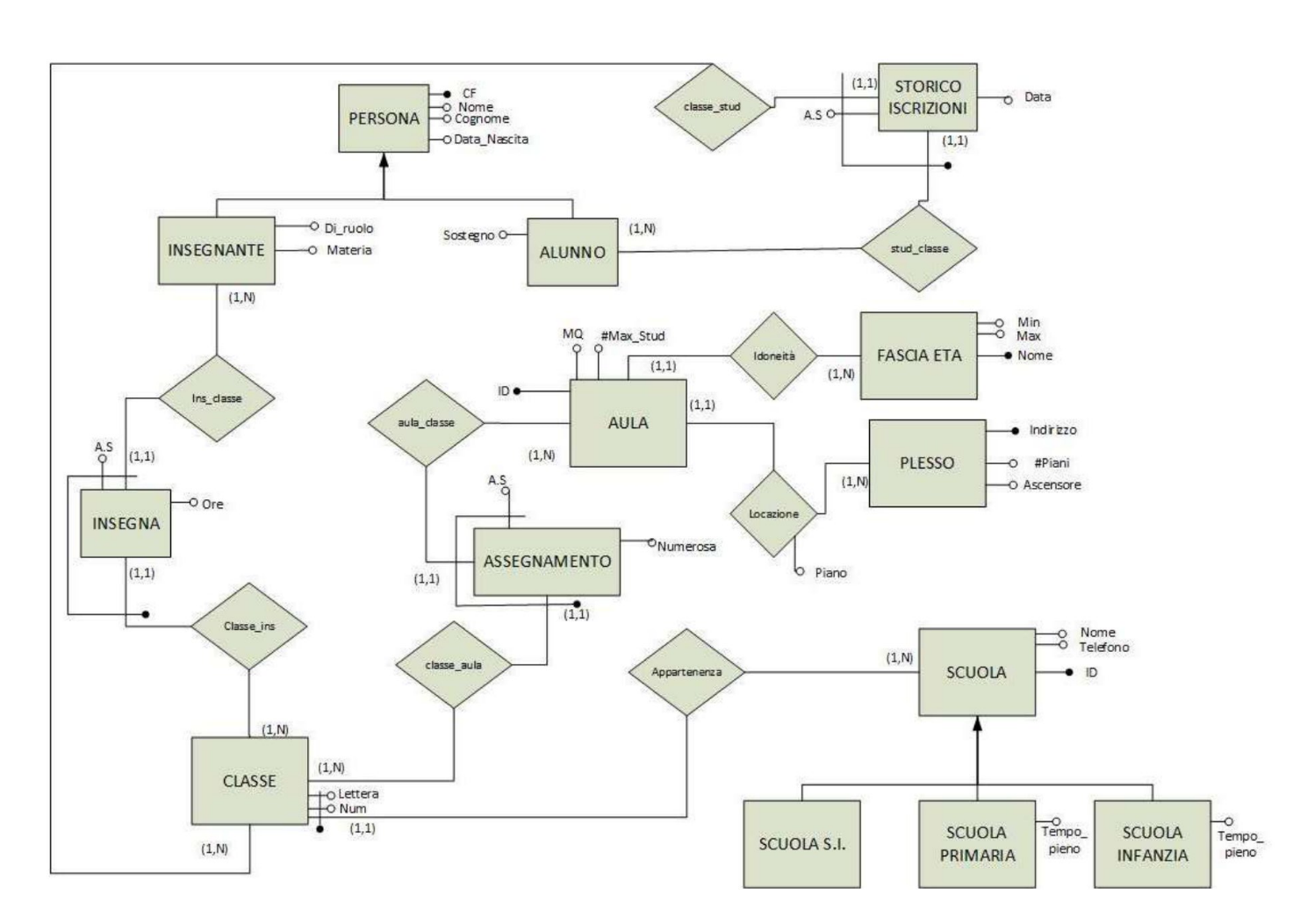
Una volta analizzati i dati si procede con l’individuazione ed eliminazione delle ambiguità. Nella seguente tabella vengono specificati le correzioni effettuate.

|  |  |  |  |
| --- | --- | --- | --- |
| Riga | Termine | Sinonimi | Motivazione della correzione |
| 1-3 | complesso scolastico | istituto | Sinonimi |
| 4 | scuola dell’infanzia, scuola primaria e scuola secondaria inferiore | gradi | Sinonimi |
| 5 | edifici | plessi | Sinonimo |
| 6-9,  11-12 | studente, ragazzo | alunno | Sinonimi |

## Progettazione concettuale

Durante la progettazione concettuale si individuano le entità e le relazioni e si procede con la rappresentazione della base di dati “Istituto comprensivo” tramite lo Schema ER.

**Schema Relazionale**



In questa fase abbiamo ritenuto opportuno ottimizzare lo schema effettuando delle ottimizzazioni:

L’entità SCUOLA è stata ottimizzata rimuovendo le specializzazioni (*SCUOLA S. I., SCUOLA PRIMARIA, SCUOLA INFANZIA*) e aggiungendo l’attributo *grado*:

**SCUOLA(ID, Nome, Telefono) → SCUOLA(ID, Nome, Telefono, Grado, Tempo\_pieno)**

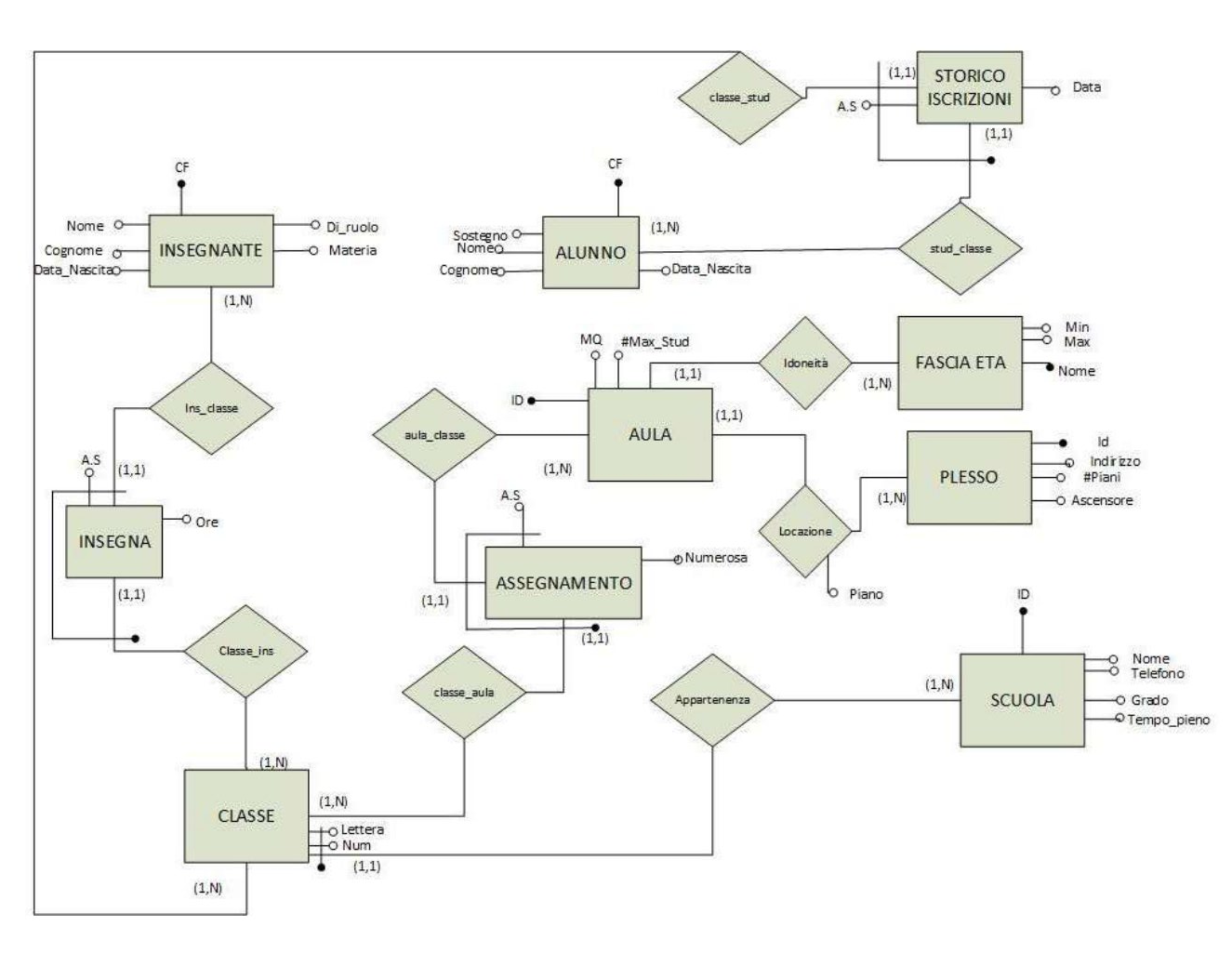
L’entità PLESSO è stata ottimizzata come segue:

**PLESSO(Indirizzo, #Piani, Ascensori) → PLESSO(id, Indirizzo, , #Piani, Ascensori)**

L’entità PERSONA è stata ottimizzata rimuovendo la sua generalizzazione e i suoi attributi sono stati accorpati nelle distinte entità ALUNNO e INSEGNANTE.

A seguito delle ottimizzazioni il nuovo scherma ER della base di dati è il seguente:

**Schema Relazionale Ottimizzato**



## Progettazione Logica

**Schema Logico**

1. ALUNNO(**CF**, nome, cognome, data\_nascita, sostegno)
2. INSEGNANTE(**CF**, nome, cognome, data\_nascita, di\_ruolo, materia)
3. CLASSE(**numero, lettera, scuola\***)
4. AULA(**id**, mq, #max:stud, fasciaEtà\*, plesso\*,piano)
5. PLESSO(**id**, indirizzo,#piani, ascensore)
6. FASCIAETÀ(**nome**, min, max)
7. SCUOLA(**id**, nome, telefono, tipo, tempoPieno)
8. INSEGNA(**id\_ins\*, <lettera, numero, scuola\*>\*, AS**, ore)
9. ASSEGNAMENTO(**<lettera, numero, scuola\*>\*, id\_aula\*, AS**, numerosa)
10. STORICOISCRIZIONE(**id\_alunno\*, <lettera, numero, scuola\*>\*, AS**, data)

Al fine di ottenere una semplificazione dello schema e un risparmio in termini di occupazione di memoria, l’entità CLASSE è stata ottimizzata come segue:

**CLASSE(numero, lettera, scuola\*) → CLASSE(id, numero, lettera, scuola\*)**

con il ***vincolo di integrità***:

A id diversi corrispondono valori diversi della tripla *<numero, lettera, scuola>*.

**Schema Logico Ottimizzato**

1. ALUNNO(**CF**, nome, cognome, data\_nascita, sostegno)
2. INSEGNANTE(**CF**, nome, cognome, data\_nascita, di\_ruolo, materia)
3. CLASSE(**id**, numero, lettera, scuola\*)
4. AULA(**id**, mq, #max:stud, fasciaEtà\*, plesso\*,piano)
5. PLESSO(**id**, indirizzo,#piani, ascensore)
6. FASCIAETÀ(**nome**, min, max)
7. SCUOLA(**id**, nome, telefono, tipo, tempoPieno)
8. INSEGNA(**id\_ins\*, id\_classe\*, AS**, ore)
9. ASSEGNAMENTO(**id\_classe\*, id\_aula\*, AS**, numerosa)
10. STORICOISCRIZIONE(**id\_alunno\*, id\_classe\*, AS**, data)

**Vincoli di integrità**

Se il tipo scuola di una istanza di SCUOLA è SI, allora l’attributo tempoPieno deve assumere valore NULL.

Ad ogni classe, ogni anno, non può essere iscritto un numero di alunni superiore al numero massimo di studenti ospitabili dall’aula associata alla classe per quell’A.S..

## Progettazione fisica

In questa fase si procede con la creazione delle tabelle all’interno del database, successivamente le tabelle vengono popolate e infine vengono svolte delle interrogazioni al database tramite il linguaggio SQL.

**Esempio di creazione della tabella *Alunno***

CREATE TABLE Alunno(

CF CHAR(16) NOT NULL,

nome VARCHAR(20),

cognome VARCHAR(20),

data\_nascita DATE,

sostegno BOOLEAN,

PRIMARY KEY(CF)

);

**Esempio di popolazione della tabella *Alunno***

INSERT INTO Alunno (CF, nome, cognome, data\_nascita, sostegno)

VALUES (GVNDLC88T23G316K, Giovanni, DeLuca, 23/03/1988, True);

## Specifica delle Operazioni sui dati previste

1. Trovare gli insegnanti che insegnano in più di una scuola.

SELECT cf, count(classe.scuola)

FROM insegnante, insegna, classe, scuola

WHERE insegna.cf\_ins = insegnante.cf AND insegna.id\_classe = classe.id AND scuola.id = classe.scuola

GROUP BY cf

HAVING COUNT(classe.scuola) > 1;

1. Trovare per ogni anno scolastico, il numero di ore assegnate a ciascun insegnante.

SELECT cf\_ins, anno\_scolastico, SUM(ore)

FROM insegnante, insegna

WHERE cf = insegna.cf\_ins

GROUP BY cf\_ins, anno\_scolastico

HAVING SUM(ore) > 0;

1. Trovare gli studenti iscritti alla scuola secondaria nell’anno 2020.

SELECT cf, anno\_scolastico

FROM alunno, storicoiscrizioni

WHERE cf = cf\_alun

GROUP BY cf, anno\_scolastico

HAVING anno\_scolastico = 2020;

1. Controllare che ad un’aula non vengano assegnati più iscritti di quanti ne possa ospitare.

SET @num\_stud := (SELECT num\_max\_stud

FROM aule

LIMIT 1);

CREATE ASSERTION CheckNumStudent CHECK(

NOT EXIST(

SELECT c.id

FROM storicoiscrizioni AS si, aule AS a, classe AS c, assegnamenti AS asg

WHERE c.id = si.id\_classe AND asg.id\_classe = c.id AND asg.id\_aula = a.id

GROUP BY c.id, si.anno\_scolastico

HAVING COUNT(\*) >= @num\_stud

)

)

1. Alla scadenza delle iscrizioni annuali (1 febbraio) si memorizzi il numero di nuovi iscritti per grado e classe (1-4 infanzia, 1-5 primaria e 1-3 medie).

-- Creazione della tabella --

CREATE TABLE reg\_eventi (

id INT PRIMARY KEY AUTO\_INCREMENT,

anno\_scolastico VARCHAR(10),

grado INT,

classe VARCHAR(10),

numero\_iscritti INT,

data\_scadenza DATE

);

-- Creazione dell'evento schedulato --

CREATE EVENT inserimento\_reg\_eventi

ON SCHEDULE

EVERY '1' YEAR

STARTS '2024-02-01 00:00:00'

DO

BEGIN

-- Calcola il numero di nuovi iscritti per ogni grado e classe --

INSERT INTO reg\_eventi (anno\_scolastico, grado, classe, numero\_iscritti, data\_scadenza)

SELECT

SI.AS,

c.grado,

c.classe,

COUNT(SI.id\_alunno) AS numero\_iscritti,

'2024-02-01' AS data\_scadenza

FROM STORICOISCRIZIONE SI

INNER JOIN CLASSE c ON SI.id\_classe = c.id

WHERE SI.data >= '2023-02-01' AND SI.data < '2024-02-01'

GROUP BY SI.AS, c.grado, c.classe;

END;