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

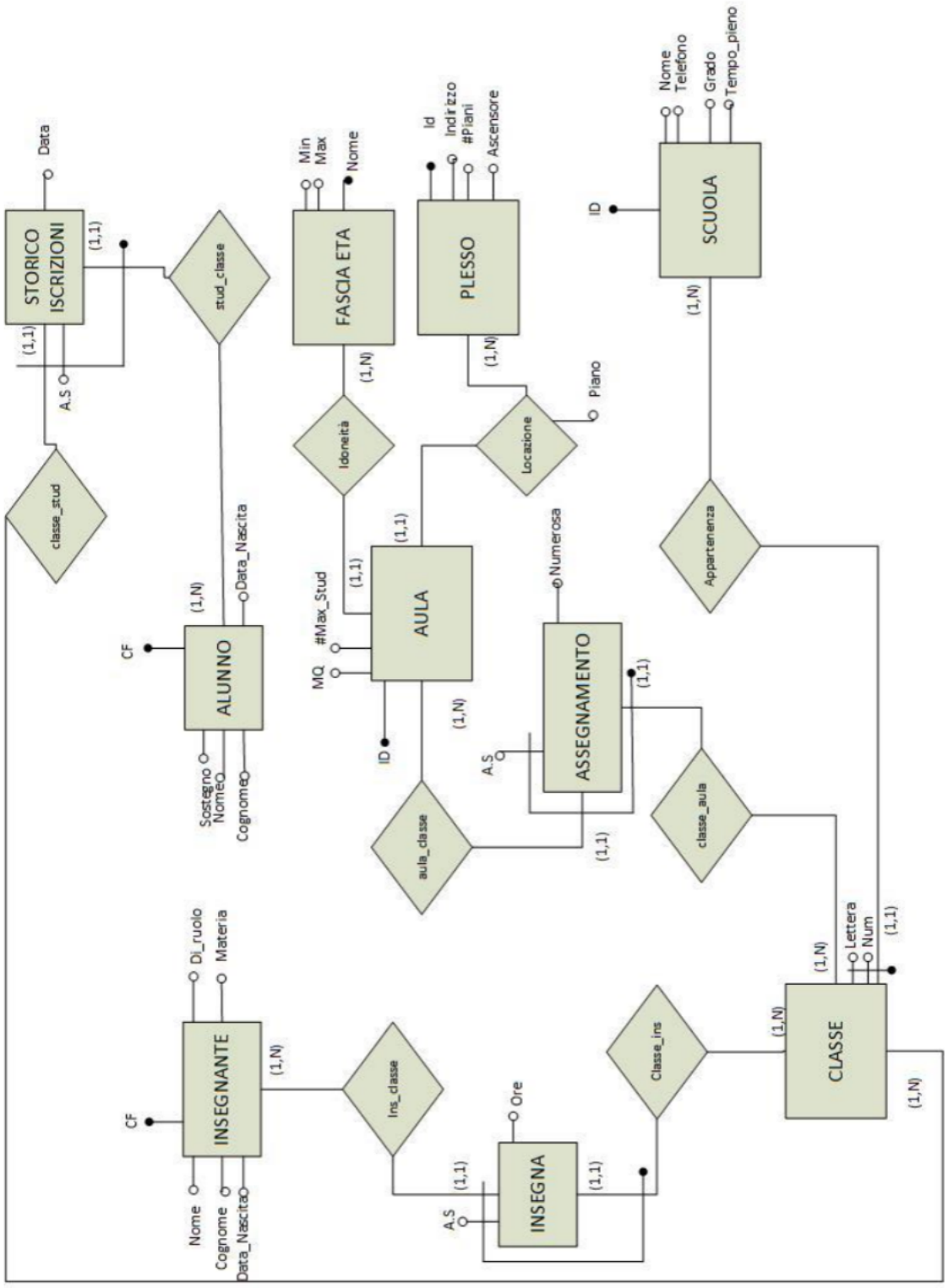
LABORATORIO: ING. G. LABOCCETTA,
DOTT.SSA D. ANGILICA

SISTEMA INFORMATIVO

PER LA GESTIONE DI

UN ISTITUTO COMPRENSIVO.

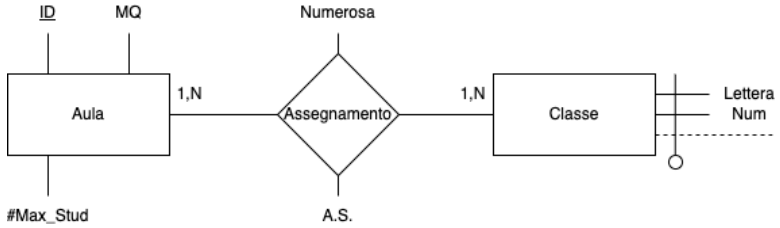
GRUPPO SQL6
<234496 Matteo Canino>
<234541 Pierfrancesco Napoli>


STORICO ISCRIZIONI
Data
A.S
classe_stud
stud_classe
ALUNNO
CF
Nome
Cognome
Sostegno
Data_Nascita
FASCIA ETA
Min
Max
Nome
Idoneità
#Max_Stud
MQ
PLESSO
Id
Indirizzo
#Piani
Ascensore
Locazione
Piano
AULA
ID
aula_classe
ASSEGNAMENTO
Numerosa
A.S
classe_aula
SCUOLA
ID
Nome
Telefono
Grado
Tempo_pieno
Appartenenza
INSEGNANTE
CF
Nome
Cognome
Data_Nascita
Di_ruolo
Materia
Ins_classe
INSEGNA
A.S
Ore
Classe_ins
CLASSE
Lettera
Num
(1,1)
(1,N)

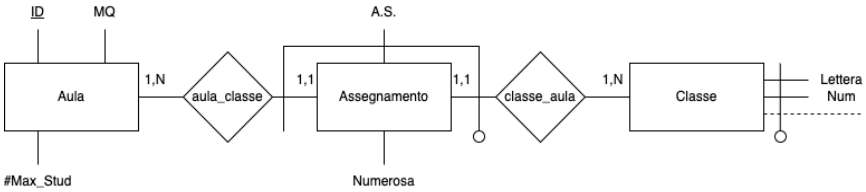
1. Progettazione Concettuale

In questo progetto abbiamo sviluppato un sistema informatico per la gestione di un istituto comprensivo. Una volta studiati i requisiti richiesti per la progettazione, siamo partiti dalla progettazione concettuale realizzando uno schema Entità-Relazione che rappresentasse la realtà d'interesse. Dopo aver rappresentato lo schema ER abbiamo apportato ulteriori ottimizzazioni allo schema applicando reificazioni delle associazioni e risolvendo eventuali generalizzazioni.

• **REIFICAZIONI**: a volte un concetto lega tra loro due concetti (entità). Pertanto, si presenta come relazione. Tuttavia, una relazione contiene coppie di occorrenze delle entità senza duplicazioni. Talvolta questo è un problema. Per rappresentare la ripetizione di uno stesso evento occorre reificare la relazione trasformandola in un'entità legata alle precedenti entità con due relazioni distinte. Nel nostro caso abbiamo applicato diverse reificazioni delle associazioni, un esempio è con il seguente schema:

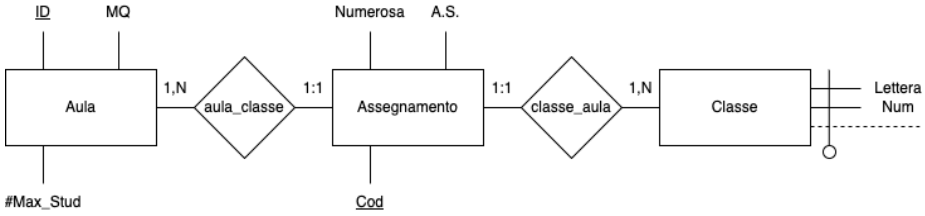


Dopo la reificazione dell'associazione "Assegnamento" avremo:



Tra Aula-Assegnamento-Classe abbiamo applicato un processo di reificazione. Abbiamo, quindi, sostituito la relazione n:m "Assegnamento" con l'entità "Assegnamento" e due relazioni 1:n. In questo modo la chiave primaria di "Assegnamento" sarà composta dall'unione delle chiavi primarie di "Aula" e "Classe" e dall'aggiunta dell'attributo "A.S.". In questo modo è possibile rappresentare il fatto che una classe può essere stata associata ad un'aula più volte in anni scolastici diversi. Possibili istanze dell'entità "Assegnamento" potrebbero, infatti, essere {<Classe1, Aula1, 2022>, <Classe1, Aula1, 2023>, …, <Classe2, Aula2, 2021>}.

Un metodo alternativo per risolvere la reificazione è quello di aggiungere un codice univoco all'interno dell'entità "Assegnamento" in modo tale da evitare la chiave composta:



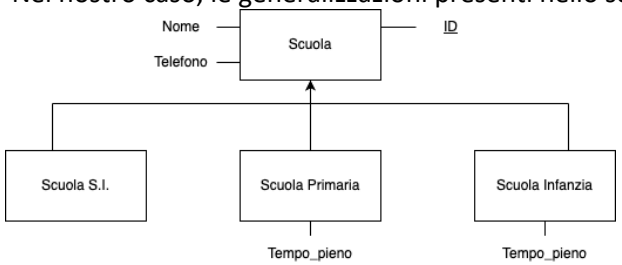
In questo caso, possibili istanze potrebbero essere:

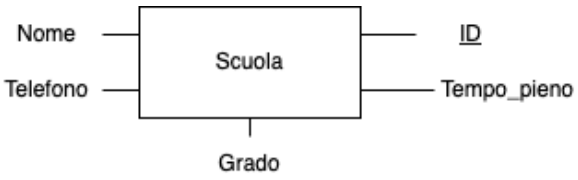
- Istanze di "Assegnamento": {<A1, True, 2022>, <A2, False, 2023>, ...}.
- Istanze di "aula_classe": {<Aula1, A1>, <Aula2, A2>, <Aula3, A1>, ...}.
- Istanze di "classe_aula": {<A1, Classe1>, <A2, Classe2>, ...}.

Possiamo notare come questo approccio non fornisce risultati corretti. Infatti, potremmo avere due aule che nello stesso anno sono state assegnate alla stessa classe. Quindi l'approccio migliore, in questo caso, risulta essere la chiave composta.

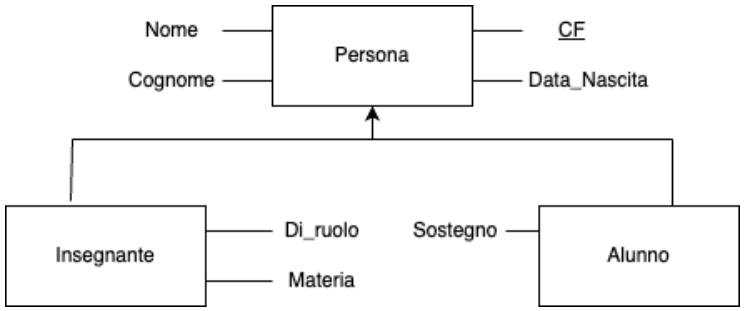
• **GENERALIZZAZIONI**: le generalizzazioni sono legami logici tra le entità. Un'entità genitore (generalizzazione) può avere uno o più legami con entità figlie (specializzazione).

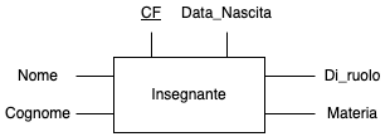
Nel nostro caso, le generalizzazioni presenti nello schema erano due. La prima è la seguente:

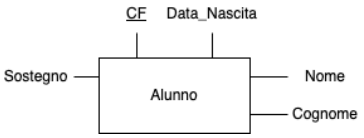




Questa prima generalizzazione è totale-disgiunta (è totale perché l'insieme delle entità figlie costituisce l'insieme del padre ed è disgiunta perché una entità può essere di un solo tipo di figlio) ed è stata risolta accorpando le entità figlie nell'entità padre. Nello specifico è stato aggiunto all'entità "Scuola" l'attributo "Tempo_pieno" (presente nelle entità "Scuola Primaria" e "Scuola Infanzia") e un nuovo attributo "Grado" che rappresenta il tipo di scuola (Scuola S.I., Scuola Primaria o Scuola Infanzia).
Abbiamo applicato questo metodo perché il padre ha la relazione appartenenza con classe che si sarebbe dovuta ripetere per ognuna delle figlie; quindi, è stato scelto di accorpare le figlie nel genitore perché gli accessi al genitore e alle figlie sono contestuali, cioè, fatti dalle stesse operazioni.
La seconda generalizzazione è la seguente:





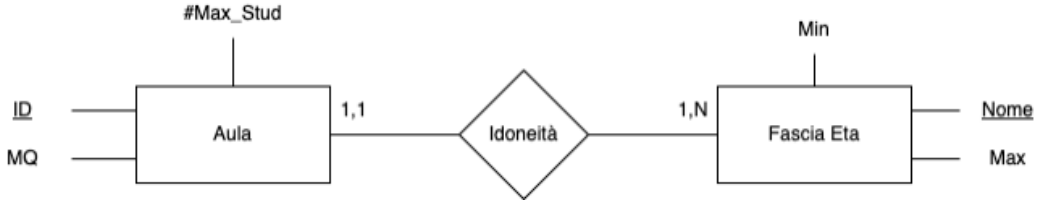


Questa seconda generalizzazione è parziale-disgiunta (è parziale perché esistono entità del padre che non sono di nessun tipo figlio ed è disgiunta perché una entità può essere di un solo tipo di figlio) ed è stata risolta accorpando il genitore nelle figlie. Nello specifico sono stati aggiunti gli attributi dell'entità "Persona" in "Alunno" e in "Insegnante". Abbiamo applicato questo metodo perché l'attributo "Sostegno" è unico dell'entità "Alunno" mentre gli attributi "Di_ruolo" e "Materia" sono unici dell'entità "Insegnante". Se avessimo accorpato le figlie nel padre, le tuple dell'entità padre avrebbero presentato dati ridondanti. Infatti, una tupla alunno avrebbe avuto informazioni relative agli attributi "Di_ruolo" e "Materia" (appartenenti ad insegnante) con valori posti a NULL e di conseguenza una tupla insegnante avrebbe avuto informazioni relative all'attributo "Sostegno" (appartenente ad alunno) con valore NULL. Inoltre, oltre agli attributi con valori nulli avremmo dovuto avere come vincoli di cardinalità 0:n nelle relazioni "Ins_classe" e "stud_classe." Quindi è stato scelto di accorpare il genitore alle figlie perché gli accessi alle figlie sono fatti da operazioni distinte

In tutti e due i casi non è stato scelto di sostituire la generalizzazione con relazioni perché gli accessi alle entità figlie non sono distinti dagli accessi al padre.

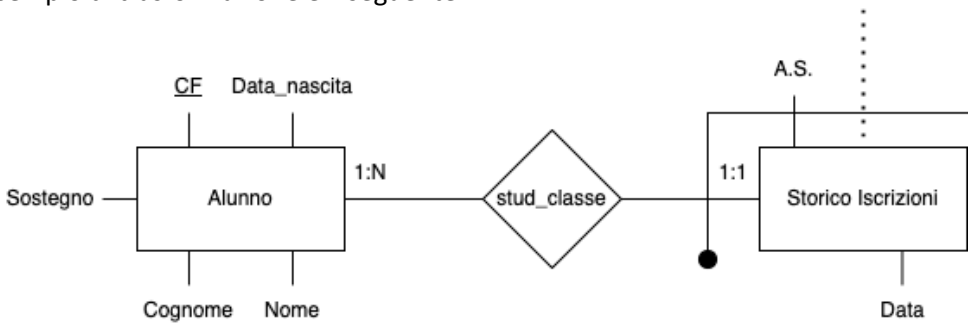
2. Progettazione Logica

Successivamente siamo passati alla Progettazione Logica che consiste in trasformare le relazioni presenti nello schema ER in schema logico. Un primo esempio di trasformazione è il seguente:



In questo caso, trovandoci davanti ad un'associazione di tipo 1:N, andiamo ad aggiungere ad Aula la chiave primaria dell'entità "Fascia Età" come chiave esterna. Avendo il valore minimo della cardinalità pari a 1 è presente il vincolo che vieta valori nulli. Inoltre, la chiave esterna non ha un vincolo di unicità. Infatti, se avessimo assegnato un vincolo di unicità alla chiave esterna "fasciaEtà" dell'entità "Aula" non saremmo stati in grado di rappresentare il fatto che ci possono essere più aule diverse assegnate contemporaneamente alla stessa fascia di età.

Un secondo esempio di trasformazione è il seguente:



In questo caso, trovandoci davanti ad un'associazione di tipo 1:N andiamo ad aggiungere a "Storico Iscrizioni" la chiave primaria dell'entità "Alunno" come chiave esterna. Essendo un'associazione 1:n con entità debole, la chiave secondaria che viene aggiunta va a formare la chiave primaria di "Storico Iscrizioni". Avendo il valore minimo della cardinalità pari a 1 è presente il vincolo che vieta valori nulli. Inoltre, la chiave esterna non ha vincolo di unicità. Infatti, se avessimo assegnato un vincolo di unicità alla chiave esterna "id_alunno" di "Storico Iscrizioni" non avremmo potuto rappresentare il fatto che uno studente, per esempio in caso di bocciatura, possa essere iscritto alla stessa classe in più anni scolastici diversi.

Lo schema logico relativo al nostro schema ER è il seguente:

1. ALUNNO(**CF**, nome, cognome, data_nascita, sostegno)
2. INSEGNANTE(**CF**, nome, cognome, data_nascita, di_ruolo, materia)
3. CLASSE(**id**, numero, lettera, id_scuola*)
4. AULA(**id**, mq, #max:stud, nome_fasciaEtà*, id_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)

3. Progettazione Fisica

3.1 Trovare gli insegnanti che insegnano in più di una scuola:

```sql
SELECT DISTINCT id_ins
FROM insegna AS i1, classe AS c1
WHERE id_classe=id AND AnnoS=YEAR(CURDATE()) AND EXISTS (SELECT *
                                                         FROM insegna, classe
                                                         WHERE id_classe=id and i1.id_ins=id_ins AND c1.id_scuola<>id_scuola
                                                             AND AnnoS=i1.AnnoS)
```

Rappresentazione della query tramite una espressione algebrica equivalente:

$$Insegna' = \rho_{id\_ins' \leftarrow id\_ins,\ id\_classe' \leftarrow id\_classe,\ AS' \leftarrow AS,\ ore' \leftarrow ore}(Insegna)$$

$$Classe' = \rho_{id' \leftarrow id,\ numero' \leftarrow numero,\ lettera' \leftarrow lettera,\ id\_scuola' \leftarrow id\_scuola}(Classe)$$

$$\pi_{id\_ins}(\sigma_{AS=AS' \wedge id\_ins=id\_ins' \wedge id\_scuola!=id\_scuola'}((\sigma_{AS=2023}Insegna) \bowtie_{id\_classe=id} Classe) \bowtie (\sigma_{AS=2023}Insegna' \bowtie_{id\_classe=id} Classe')))$$

3.2 Trovare gli studenti che hanno frequentato almeno una classe per ogni grado dell'istituto:

```sql
SELECT CF
FROM alunno
WHERE NOT EXISTS (SELECT DISTINCT tipo
                  FROM scuola AS s1
                  WHERE NOT EXISTS (SELECT *
                                    FROM storicoiscrizioni AS si, classe AS c, scuola AS s2
                                    WHERE si.id_alunno=CF AND c.id=si.id_classe AND c.id_scuola=s2.id AND s2.tipo=s1.tipo));
```

3.3 Impedire l'iscrizione di uno studente ad una classe associata ad un'aula non compatibile con la sua età:

```sql
delimiter //
CREATE TRIGGER `check_età_alunno` AFTER INSERT ON `storicoiscrizioni`
FOR EACH ROW
BEGIN
DECLARE AGE INT;
SELECT YEAR(CURDATE())-YEAR(data_nascita) INTO AGE FROM alunno WHERE NEW.id_alunno=CF;

    IF (NOT EXISTS ( SELECT *
                FROM  assegnamento AS a, aula, fasciaetà AS fe
                WHERE a.id_classe=NEW.id_classe AND a.AnnoS=NEW.AnnoS AND a.id_aula=id
                AND nome_fasciaEtà=fe.nome AND AGE BETWEEN fe.min AND fe.max;))
    THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT='Errore. L\'alunno non è compatibile con la fascia d\'età dell\'aula';
    END IF;
END; //
delimiter ;
```

3.4 All'iscrizione di uno studente con sostegno, se alla classe non è assegnato un insegnante con materia "sostegno", sollevare un warning con messaggio: "E' necessario associare un docente di sostegno alla classe":

```sql
delimiter //
CREATE TRIGGER `check_docente_sostegno` AFTER INSERT ON `storicoiscrizioni`
FOR EACH ROW
BEGIN
DECLARE alunno_sostegno BINARY(1);
SELECT sostegno INTO alunno_sostegno FROM alunno WHERE NEW.id_alunno=CF;

IF ((alunno_sostegno=TRUE) AND (NOT EXISTS (SELECT *
                                        FROM insegna, insegnante
                                        WHERE NEW.id_classe=id_classe AND NEW.AnnoS=AnnoS AND id_ins=CF AND materia='sostegno';)))
    THEN SIGNAL SQLSTATE '01000' SET MESSAGE_TEXT ='E\' necessario associare un docente di sostegno alla classe';
END IF;
END; //
delimiter ;
```

3.5 Scrivere una Stored Function che ricevuto un codice fiscale come parametro di input, restituisca l'anno scolastico della sua prima iscrizione. Scrivere quindi una query che restituisca l'anagrafica di tutti gli studenti con l'aggiunta dell'a.s. della loro prima iscrizione.

```
delimiter //
CREATE  FUNCTION `Anno_prima_iscrizione`(`CF` CHAR(16))
BEGIN
DECLARE MIN_YEAR, var INT;
DECLARE finito INT DEFAULT 0;
DECLARE cur_1 CURSOR FOR SELECT AnnoS FROM storicoiscrizioni WHERE id_alunno=CF;
DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'
    SET finito=1;
OPEN cur_1;
FETCH cur_1 INTO MIN_YEAR;
ciclo: WHILE NOT finito DO
    FETCH cur_1 INTO var;
    IF var < MIN_YEAR THEN
        SET MIN_YEAR = var;
    END IF;
END WHILE ciclo;
RETURN MIN_YEAR;
END; //
delimiter ;
```

Definizione e risultati della Query:

```
SELECT CF, nome, cognome, data_nascita, sostegno, Anno_prima_iscrizione(CF)
FROM alunno ;
```

CF	nome	cognome	data_nascita	sostegno	Anno_prima_iscrizione(CF)
28WUW2B2UWN2UW	Luigi	Capalbo	2014-09-03	0	2018
NPLPFR02H22D086I	Pierfrancesco	Napoli	2010-06-22	1	2022
RFGVEE43G32E859H	Antonio	Milazzo	1960-05-30	1	1998
SVYRSS34S18C322H	Agostina	Frugiuele	2003-09-04	0	2015

3.6 Scrivere una Stored Procedure che ricevuti in input una scuola, un numero indicante la classe e un anno scolastico restituisca in output il numero di sezioni di quella scuola per quella classe in quell'anno scolastico:

```
delimiter //
CREATE PROCEDURE `Numero_sessioni`(IN `Scuola_id` INT,IN `Numero_classe` INT,IN `Anno_S` INT,OUT `Numero_Sessioni` INT)
BEGIN
 SELECT COUNT(*) INTO Numero_Sessioni
                    FROM assegnamento, classe
                    WHERE id_scuola=Scuola_id AND id_classe=id AND AnnoS=Anno_S AND numero=Numero_classe;
END; //
delimiter ;
```

3.7 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).
Definiamo la table 'contenitore':

```
CREATE TABLE `nuove_iscrizioni` (
    `AnnoS` YEAR NOT NULL,
    `grado_scuola` ENUM('SI','SS','SP') NOT NULL,
    `numero_classe` TINYINT(4) NOT NULL,
    `numero_iscritti` INT(11) NOT NULL DEFAULT '0',
    PRIMARY KEY (`AnnoS`, `grado_scuola`, `numero_classe`)
)ENGINE=InnoDB DEFAULT CHARSET='latin1_swedish_ci';
```

Definiamo la procedura che inserirà le tuple nella table:

```
delimiter //
CREATE  PROCEDURE `mem_numero_iscrizioni`()
BEGIN
DECLARE  b INT DEFAULT 0;
DECLARE numero TINYINT;
DECLARE grado ENUM('SI', 'SS' ,'SP');
DECLARE cur_1 CURSOR FOR SELECT s.tipo, c.numero
                         FROM storicoiscrizioni AS si, classe AS c, scuola AS s
                         WHERE si.AnnoS=YEAR(CURDATE()) AND si.id_classe=c.id AND c.id_scuola=s.id;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET b = 1;
OPEN cur_1;
REPEAT
    FETCH cur_1 INTO grado, numero;
    IF b = 0 THEN
        IF (EXISTS (SELECT *
                    FROM nuove_iscrizioni
                    WHERE AnnoS=YEAR(CURDATE()) AND grado_scuola=grado AND numero_classe=numero))
                    THEN UPDATE nuove_iscrizioni
                            SET numero_iscritti=numero_iscritti + 1
                            WHERE AnnoS=YEAR(CURDATE()) AND grado_scuola=grado AND numero_classe=numero;
        ELSE INSERT INTO nuove_iscrizioni(AnnoS, grado_scuola, numero_classe, numero_iscritti) VALUES (YEAR(CURDATE()), grado, numero, 1);
        END IF;
    END IF;
UNTIL b = 1
END REPEAT;
CLOSE cur_1;
END; //
delimiter ;
```

Definiamo l'evento:

```
CREATE  EVENT `conta_iscrizioni`
    ON SCHEDULE EVERY 1 YEAR STARTS '2024-02-01'
    DO CALL `mem_numero_iscrizioni`()
```