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 ER
Gallo Agostino 234378 MAT
Lorelli Giacomo 231352 MAT
Lo Riggio Nazzareno 230796 INF
Mobilio Giuseppe 230974 INF

1 Introduzione

La seguente relazione ha lo scopo di presentare alcuni dei passaggi che noi, componenti del gruppo, consideriamo rilevati nella costruzione del sistema informatico costruito durante le ore laboratoriali del corso al fine di dimostrare la nostra attiva partecipazione a queste. A tale scopo esplicheremo alcuni dei passaggi logici che ci hanno portato dalla raccolta dati alla completa proggettazione e mostreremo commentando alcuni aspetti pratici della realizzazione.

2 Tematica Progettuale

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

3 Raccolta Dei Requisiti

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.

4 Progettazione

Dopo la lettura della traccia è prodotto un primo schema Entita-Relazione, visionabale sul file *Progetto_ER_Relazionale* tale schema è stato modificato per facilitarne la trasformazione in uno schema logico, in particolare sono state rimosse le generalizzazioni su ***Scuola***, dove le tre entità figlie (***Scuola Dell'Infanzia, Scuola Primaria, Scuola Secondaria***) sono state immerse in ***Scuola*** e sostituite da un attributo *grado*, mentre per quanto riquarda quella su ***Persona*** è stata eliminata l'entita genitore e i suoi attributi sono stati inseriti nell'entite

figlie ***Insegnate e Alunno***.
Inoltre sempre allo scopo di rendere possibile la trasformazione in schema relazionale già in precedenza erano state reificate:

la relazione ***Insegna*** tra ***Classe e Insegnate***;

la relazione ***Asseganmento*** tra ***Classe e Aula***;

la relazione ***Storico Iscrizioni*** tra ***Classe e Alunno***;

queste ultime due, anche se non esplicitamente richiesto nelle raccolta dati, risultavano essere relazioni molti a molti dal momento che si è supposto utile tenere in memoria anno per anno delle iscrizioni degli alunni nelle diverse classi, e dell' associamento che vi è tra le alue e le classi anche questo anno per anno.

5 Creazione Del Database

Mostriamo ora lo sviluppo dell'entità ***Aula***, questa dopo la stesura dello schema relazionare risultava essere identificata da:

Aula(id, mq, max_stud, nome_fasciaEtà, id_plesso*,piano)*

tale entità è stata quindi costruita sulla *Shell* di MySQL in tal modo

```
CREATE TABLE Aula
    (
        id              int UNSIGNED NOT NULL AUTO_INCREMENT,
        mq              int UNSIGNED NOT NULL,
        max_stud        int UNSIGNED NOT NULL,
        fasciaEt         enum('infante', 'bambino','fanciullo', '
            adolescente'),
        piano           int UNSIGNED NOT NULL,
        id_plesso       int NOT NULL,
        PRIMARY KEY     (id),
        FOREIGN KEY     (fasciaEt ) REFERENCES RangeFasciaEt (
            nome),
        FOREIGN KEY     (id_plesso) REFERENCES Plesso(id)
    );
```

In primo luogo notiamo che la chiave primaria ***id*** può essere gestita da una funzione di *auto_increment* in quanto tale valore è stato da noi supposto come del tutto arbitrario e non rappresenta nessun concetto (non vi sono vincoli di alcun genere come ad esempio aule con numeri contigui si trovano nel medesimo adificio) la sua unica necessità è che soddisfi le condizioni tali per cui può essere una chiave primaria. Inoltre osserviamo che in questa entità troviamo diversi vincoli uno è quello tra fascia età e aula il quale è un vincolo d'integrità relazionale, questo vincolo è gestito internamente da MySQL tramite *FOREIGN KEY*.
Sul medesimo concetto logico ricade anche la creazione di un trigger che garantisce che in aula non vi siano studenti con un età non conforme con il range d'età assegnato all'aula, questo è stato sviluppato nel seguente modo

```
DELIMITER //

CREATE TRIGGER CheckEt
  BEFORE INSERT ON StoricoIscrizione
  FOR EACH ROW
  IF(((select YEAR(NOW())-YEAR(data_nascita)
       from Alunno
       where Alunno.cf = New.id_alunno) > (select x.max
                                           from RangeFasciaEt  as
                                               x, Assegnamento as y
                                               , Aula as z
                                           WHERE NEW.id_classe=y.
                                               id_assegnamento and
                                               y.id_aula=z.id and z
                                               .fasciaEt  = x.nome
                                               )) or
     ((select YEAR(NOW())-YEAR(data_nascita)
       from Alunno
       where Alunno.cf = New.id_alunno) < (select x.min
                                           from RangeFasciaEt  as
                                               x, Assegnamento as y
                                               , Aula as z
                                           WHERE NEW.id_classe=y.
                                               id_assegnamento and
                                               y.id_aula=z.id and z
                                               .fasciaEt  = x.nome
                                               )))
  THEN SIGNAL SQLSTATE '45000' set message_text = 'An error
      occurred';
  END IF;//

DELIMITER ;
```

Questo trigger agisce prima dell'inserimento dell'iscrizione di un alunno a una classe è nel caso in cui non soddisfi la condizionne necessaria ne impedisce l'inserimento, tale condizione viene verificata tramite due disequazioni, nessuna delle quali deve vericarsi, la prima si accerta che l'età dell'alunno che si vuole inserire non sia maggiore della massima età consentita dal range nell'aula dove si trova la sua classe,mentre la seconda disequazione è analoga ma verifica la validità sull'estremo inferiore.
In particolare l'età dell'alunno viene calcolata prendendo da sistema l'anno corrente e sottraendolo alla data di nascità dell'alunno che si vuole inserire disponibile nel database (notiamo che tale operazione non tiene conto del mese di nascita e che inoltre il convenzionale anno scolastico non coincide con l'anno Gregoriano). Questa viene poi messa a confronto con età massima(e età minima) disponibile in range fascia età nella colonna *max(min)* per arrivare a tali valori effettuiamo un join tra ***StoricoIscrizione*** e ***Assegnamento*** poi tra la tabella risultante del primo join e ***Aula*** e infine con l'entità ***RangeFasciaEtà***, questo ci permette di trovare la fascia d'età dell'aula alla quale è assegnata la classe dove è stato iscritto il nuovo alunno. Nel caso in cui la condizione necessaria non dovesse essere soddisfata il comando *SIGNAL SQLSTATE* poiché ha

codice 45000 (previsto per gli errori custom) fa si che il trigger interrompa la procedura di inserimento, restituendo un errore.
Altro vincolo logico associato all'aula è quello che assicura che il piano sulla quale questa è ubicata non dovrebbe essere superiore al numero di piani dell'edifico (del plesso) in cui si trova.

```
DELIMITER //

CREATE TRIGGER Piano
  BEFORE INSERT ON Aula
  FOR EACH ROW
  IF(NEW.piano>(SELECT piani
                FROM plesso
                WHERE plesso.id = NEW.id_plesso))
  THEN SIGNAL SQLSTATE '45000' set message_text = 'Errore nel piano
      ';
  END IF;//

DELIMITER ;
```

Anch'esso come il trigger sopra agisce prima di un inserimento in una tabella (in questo caso ***Aula***) tramite l'operatore maggiore si assicura che il piano dove si sta tentando di inserire la nuova aula non superi l'ultimo piano presente nel plesso da noi scelto, il numero di piani viene trovato grazie a una selection innestata e ad una join tra il plesso dell'aula che vogliamo inserire e l'insieme dei plessi presenti nel database. Tale trigger si basa sull'assunto fondamente che i piani siano tutti numerati (nessun piano terra la numerazione parte da 1) e che non vi siano piani interrati con numeri negativi all'interno del plesso (se ciò accadesse infatti potrebbe accadere che il numero di piani del plesso non coinciderebbe con il piano più alto)

6 Verifica Integrità

Per assicurarci l'integrità del database, da noi costruito, vi abbiamo caricato sopra dei dati di test e interrogandolo con delle query e ne abbiamo osservato il risultato confrontandolo con quello da noi atteso (calcolato a mano), una di queste e la sequente la quale trova gli studenti iscritti alla scuola secondaria nell'anno 2020.

```
select Id_alunno
from Storicoiscrizione as s, Classe, Scuola
Where AS=2020 and S.id_classe=Classe.ID and Classe.id_scuola=Scuola
    .ID and Scuola.tipo='SS'
```

Tale query costruisce una nuova tabella joinando le varie entità tramite una correlazione tra chiavi primarie e secondarie delle rispettive entità, di tale tabella

ne restituisce gli *Id* degli alunni appartenenti nel 2020 a classi di scuole secondarie.
Altra query di test e quella che trova gli insegnanti che hanno insegnato in più di una scuola nel corso della loro carriera(registrato nel database).

```
select id_ins
from Insegna as x, Insegna as y, Classe as c1, Classe as c2
where x.id_ins=y.id_ins and x.id_classe= c1.ID and y.id_classe= c2.
    ID and c1.id_scuola<>c2.id_scuola
```

Più complessa dal punto di vista logico questa query ci mostra il join sulla stessa entità tramite una ridenominazione della stessa e ciò accade sia per ***Insegna*** che per ***Classe***, il primo join tra le ridenominazioni di ***Insegna*** restituisce tutte le combinazioni delle possibili classi in cui c'è stato lo stesso insegnante, i successivi join permettono di stabilire a che scuola appartengono le varie classi e infine la disuguaglianza si assicura che le scuole si trovano le classi siano diverse

Tutti i codici utilizzati per la costruzione della base di dati sono visionabili presso *https://github.com/GiakElle/SQL_Exam*