Progetto Object Orientation e Basi di Dati

Lorenzo Tecchia N86004446
Mirko Marciano N86004019
10/12/2022

Progettazione e sviluppo di un applicativo per la gestione di una Biblioteca Digitale

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Descrizione e Analisi del Progetto

1.1 Descrizione e risoluzione sintetica

Il progetto consiste in una base di dati relazionale unito ad un'interfaccia grafica costruita in Java per la gestione di una libreria digitale. Si è pensato di implementare una base di dati relazionali con server in locale tramite il dialetto PostgreSQL.

Si è pensato di articolare la base dati in cinque classi di oggetti per tenere traccia di articoli, libri, autori, collane e riviste.

E' stato messo in risalto il tracciamento per autori e argomenti tramite l'utilizzo di viste.

Si è pensato di controllare gli inserimento delle tuple nel database tramite implementazioni di trigger e domini che controllano ad ogni inserimento la validità degli elementi caratteristici degli oggetti consumabili all'interno della biblioteca, quali ISBN e ISSN per libri e collane rispettivamente. Per articoli e riviste vengono controllati gli inserimenti corretti di DOI e ISSN rispettivamente.

Tutti i tipi di codici menzionati sono stati scelti come vincoli di chiavi primarie per i rispettivi oggetti.

Capitolo 2

Progettazione concettuale

2.1 Class Diagram

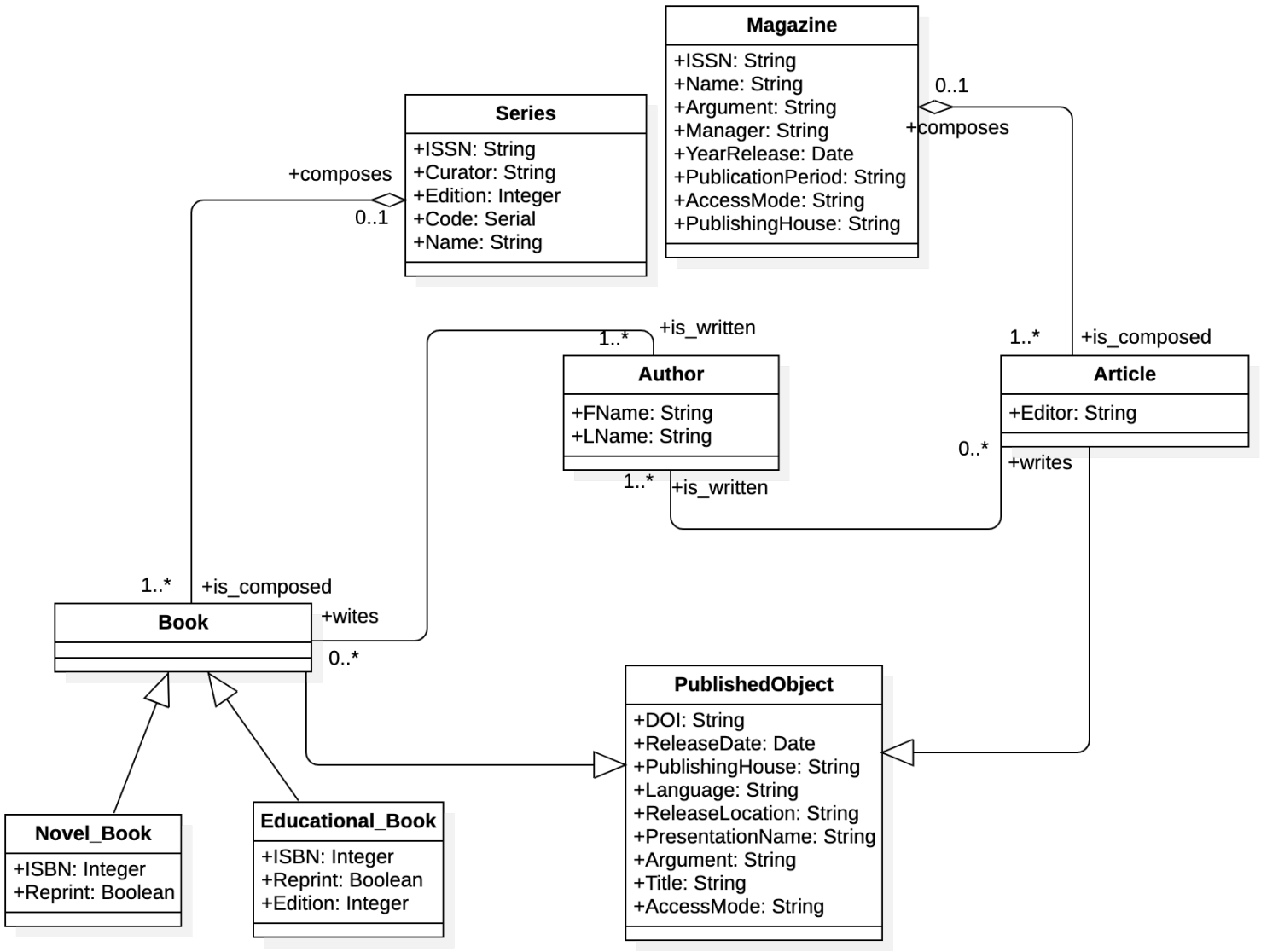

Figura 2.1: Class Diagram

2.2 Analisi della ristrutturazione del Class Diagram

2.2.1 Analisi delle ridondanze

Le ridondanze individuate nell class diagram sono presenti nella specializzazione della classe `Book` in: `Novel_Book` e `Educational_Book`.
Gli attributi: `ISBN` e `Reprint` di entrambe le specializzazioni vengono inserite all'interno della classe `Book`.

2.2.2 Analisi degli identificativi

Identificativo primario per la classe `Book` è l'attributo `DOI`, viene attribuito all'attributo `ISBN` il vincolo di chiave candidata. Per la classe `Author` viene implementato un attributo aggiuntivo: `ID_Author` che sarà chiave primaria. Per la classe `Article` viene usato l'attributo `DOI` come chiave primaria. Per le classi `Series` e `Magazine` viene usato come chiave primaria l'attributo `ISSN`.

2.2.3 Rimozione degli attributi multipli

Non sono presenti attributi multipli all'interno del Class Diagram.

2.2.4 Rimozione degli attributi composti

Non sono presenti attributi composti all'interno del Class Diagram.

2.2.5 Partizione/Accorpamento delle associazioni

In questo Class Diagram non sono presenti associazioni 1..1 da eliminare.

2.2.6 Rimozione delle gerarchie, delle composizioni

Nel Class Diagram vengono incorporati nella classe `Book` le specializzazioni `Novel_Book` e `Educational_Book` assieme ai rispettivi attributi. All'attributo `Argument` nel Class Diagram ristrutturato, viene specificato l'argomento educativo (Filosofia, Economia ecc.) oppure all'attributo viene assegnato il valore: Romanzo.
Dato l'interesse di tracciamento delle classi `Book` e `Article` in modo separato, viene eliminata la classe `PublishedObject` e i suoi attributi vengono inseriti all'interno delle classi menzionate precedentemente. Le composizioni che riguardano le classi `Series` e `Magazine`, vengono eliminate e sostituite da una semplice associazione.

2.3 Class Diagram ristrutturato

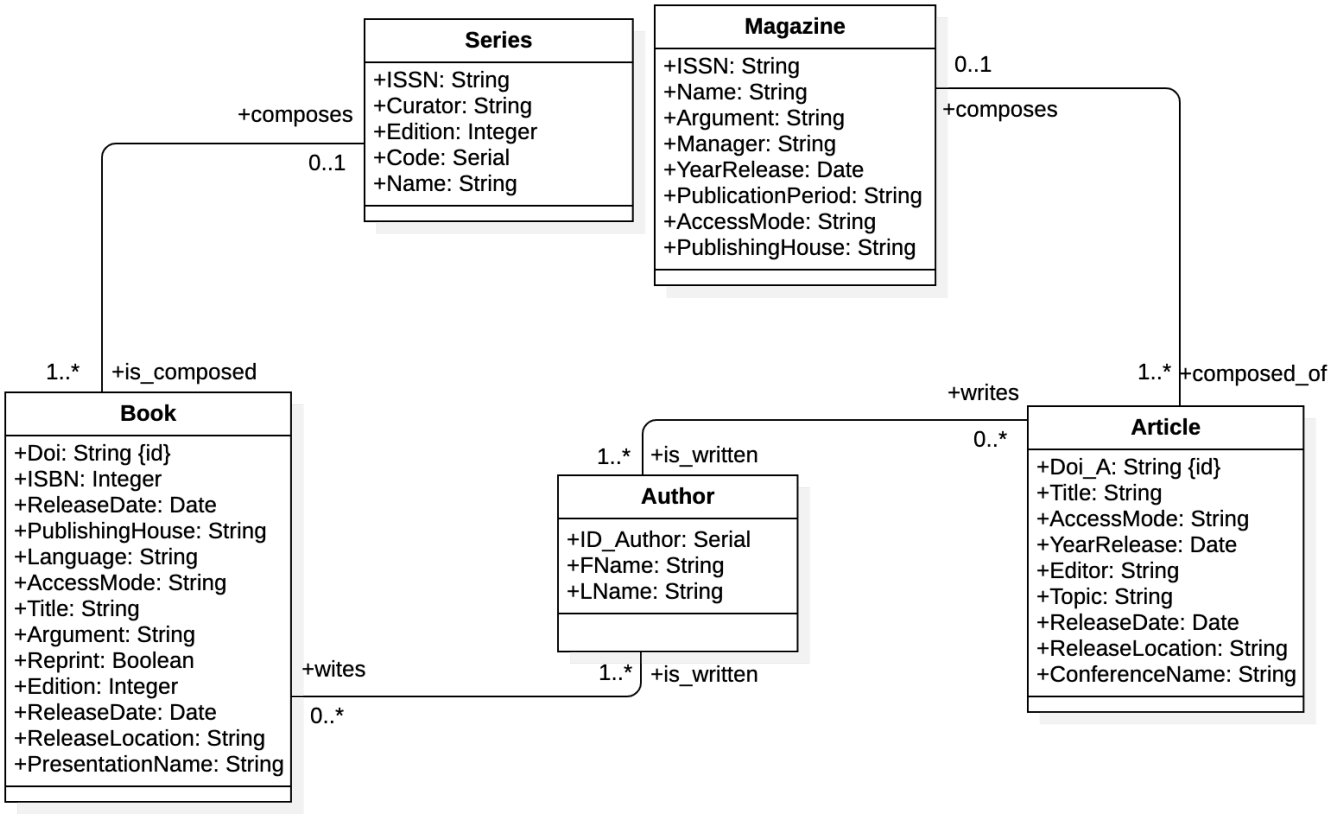

Figura 2.2: Class Diagram
Ristrutturato

2.4 Dizionario delle classi

Tabella 2.1: Dizionario delle Classi

Classe	Spiegazione	Attributi
Author	Autore di libri o articoli	ID_Author (Serial): Author's identification code. FName (String): Author's first name. LName (String): Author's last name.
Books	Oggetti legibili, romanzi o d'educazione	DOI(String): Digital object Identifier of the book. ISBN (Integer): Numerical classification sequence of the book. Edition (Integer): Edition number. AccessMode (AccessMode): Fruition method. ReleaseDate (Date): Publication date. PublishingHouse (String): Publishing house that printed the book. ReleaseLocation (String): Place of publication of the book. Language (String): Language in which the book is written. Title (String): Book title. Argument (String): Book topic. Reprint (Boolean): Parameter that identifies if the book is a reprint or not. PresentationName (String): Name of pesentation in which books are presented.
Series	Insieme di libri	ISSN (String):International number that identifies serial publications. Edition (Integer):Edition number. Curator (String):Curator of the series. Code (Serial):Code assigned to the series. Name (String): Series' name.
Magazine	Insieme di Articoli	(Integer): International number that identifies serial publications. Name (String): Magazine's name. Argument (String): Magazine topic. Manager (String): Event organizer. YearRelease (Date): Publication year. PublicationPeriod (String): Periodicity of publication. AccessMode (AccessMode): Fruition method.
Article	Articoli di ricerca Scientifica	DOI (String): Digital object Identifier of the book. Title (String): Book title. AccessMode (AccessMode): Fruition method. YearRelease (Date):Publication year. Editor (String):Article editor. ReleaseDate (Date):Publication date. ReleaseLocation (String):Place of publication of the book. ConferenceName (String): Name of pesentation in which books are presented.

2.5 Dizionario delle associazioni

Tabella 2.2: Tabella delle Associazioni

Nome	Descrizione	Classi Coinvolte
composes/is_composed	Una collana è composta da uno o più libri/ Un libro può comporre oppure no una collana	Series/Book
writes/is_written	Un libro è scritto da uno o più autori/ Un autore scrive molti oppure nessun libro	Book/Author
is_written/writes	Un autore scrive molti oppure nessun articolo/ Un articolo è scritto da uno o più autori	Author/Article
composes/is_composed	Un articolo puo comporre oppure no una rivista/ Una rivista è composta da uno o più articoli	Article/Magazine

Capitolo 3

Schema logico

3.1 Spiegazione Schema Logico

Il seguente schema logico riassume la composizione del database, evidenziando le chiavi primarie (in grassetto) e le chiavi esterne per il collegamento di una tabella con l'altra (sottolineate).

3.1.1 Schema

- Author

  (**IDAuthor**, FName, LName)

- Book

  (**DOI_B**, ISBN, PublishingHouse, Language, AccessMode, Title, Argument, Reprint, Edition, ReleaseDate, ReleaseLocation, PresentationName, FK_Author, FK_Series)

  $FK\_Author \rightarrow Author(IDAuthor)$
  $FK\_Series \rightarrow Series(ISSN\_S)$

- Article

  (**DOI_A** , Title, AccessMode, YearRelease, Editor, Topic, ReleaseDate, ReleaseLocatione, ConferenceName , FK_Author, FK_Magazine)

  $FK\_Author \rightarrow Author(IDAuthor)$
  $FK\_Magazine \rightarrow Series(ISSN\_M)$

- Series

  (**ISSN_S**, Curator. Edition, Code, Name)

- Magazine

  (**ISSN_M**, Name , Argument, Manager, YearRelease, PublicationPeriod, AccessMode, PublishingHouse)

Capitolo 4

Progettazione Fisica

4.1 Creazione delle Tabelle

```
drop schema mtl cascade;
create schema mtl;

create table mtl.author
(
    CodAuthor serial
        primary key,
    FName    varchar(20),
    LName    varchar(20)
);

create table mtl.series
(
    ISSN_S issn primary key,
    Curator names,
    Edition int,
    Code_S varchar(10),
    Name_S names
);

create table mtl.magazine
(
    ISSN_M          issn primary key,
    Name_M          names,
    Argument        names,
    Manager         names,
    YearRelease     timestamp,
    PublicationPeriod names,
    AccessMode      access,
    PublishingHouse names
);

create table mtl.book
(
    Doi_B          doi
        primary key,
    ISBN_B         varchar(13)
        unique,
    PublishingHouse names,
    Language       names,
    AccessMode     access,
    Title          varchar(30),
    Argument       names,
    Reprint        boolean,
    Edition        int,
```

```
    ReleaseDate    timestamp,
    ReleaseLocation location,
    PresentationName names,
    FK_Author      serial,
    FK_Series      issn,

    constraint BookFK_2 foreign key (Fk_Author) references mtl.Author (CodAuthor) on delete cascade,
    constraint BookFK_3 foreign key (FK_Series) references mtl.Series (ISSN_S) on delete set null
);

create table mtl.article
(
    Doi_A         doi
       primary key,
    Title         varchar(40),
    AccessMode    access,
    YearRelease   timestamp,
    Editor        names,
    Topic         names,
    ReleaseDate   timestamp,
    ReleaseLocation location,
    ConferenceName varchar(50),
    FK_Author     serial,
    FK_Magazine   issn,

    constraint ArticleFK_1 foreign key (FK_Author) references mtl.Author (CodAuthor) on delete cascade,
    constraint ArticleFK_2 foreign key (FK_Magazine) references mtl.Magazine (ISSN_M) on delete set null
);
```

Docum
creazio
tabelle.

4.2 Creazione dei domini

```
   create domain issn as varchar(9)
check ( value like '%-%' );

create domain isbn as varchar(17)
check ( value like '%-_-%-%-_');

create domain doi as varchar(30)
check ( value like'10.%/%');

create domain access as varchar(20)
check ( value <> '' and value not similar to '%[0-9]+%'
       and value not similar to '%[@!# $ %&]+%');

create domain names as varchar(30)
check ( value not similar to '%[@!# $ %&]+%');

create domain location as varchar(50)
check ( value like '%,[0-9],%,[0-9],%');
```

Docum
per la
zione
domini.

4.3 Creazione delle viste

```
create view mtl.bibliography as
select distinct b.Title,b.ReleaseDate,a.lname
from mtl.book b join mtl.author a on b.fk_author = a.codauthor
order by b.releasedate desc;

create view mtl.history as
select distinct a.fname, a.lname, ar.title,ar.yearrelease,ar.editor
```

```
from mtl.author a join mtl.article ar on a.codauthor = ar.fk_author
order by ar.yearrelease desc;

create view mtl.digital_goods as
select distinct b.title from mtl.book b where accessmode = 'Digital'
union
select distinct a.title from mtl.article a where accessmode = 'Digital'
union
select distinct m.name_m from mtl.magazine m where accessmode = 'Digital'
union
select distinct s.name_s from mtl.series s join mtl.book b on s.issn_s = b.fk_series where
    b.accessmode='Digital';

create view mtl.paper_goods as
select distinct b.title from mtl.book b where accessmode = 'Paper'
union
select distinct a.title from mtl.article a where accessmode = 'Paper'
union
select distinct m.name_m from mtl.magazine m where accessmode = 'Paper'
union
select distinct s.name_s from mtl.series s join mtl.book b on s.issn_s = b.fk_series where
    b.accessmode='Digital';

create view mtl.audio_goods as
select distinct b.title from mtl.book b where accessmode = 'Audio'
union
select distinct a.title from mtl.article a where accessmode = 'Audio'
union
select distinct m.name_m from mtl.magazine m where accessmode = 'Audio'
union
select distinct s.name_s from mtl.series s join mtl.book b on s.issn_s = b.fk_series where
    b.accessmode='Digital';

create view mtl.presentation as
select b.title,a.fname,a.lname, b.presentationname, b.releaselocation, b.releasedate
from mtl.book b join mtl.author a on a.codauthor = b.fk_author;

create view mtl.discussion as
select ar.title,a.fname,a.lname, ar.conferencename, ar.releaselocation, ar.releasedate
from mtl.article ar join mtl.author a on a.codauthor = ar.fk_author
order by a.lname;
```

Docum
per la
zione
viste.

4.4 Creazione di funzioni e trigger

```
   create or replace function mtl.function_1() returns trigger as
$$
declare
    stringa_in  varchar(13) = new.isbn_b;
    sum         integer    := 0;
    var_appoggio integer;
    resto       integer;
begin
    stringa_in := replace(stringa_in, '-', '');
    for i in 1..13
        loop
            var_appoggio = cast(substring(stringa_in from i for 1) as int);
            if (i % 2 = 0) then
                sum := sum + var_appoggio * 3;
            else
                sum := sum + var_appoggio;
            end if;
```

```
        end loop;
    resto = sum % 10;
    if (resto != 0) then
        delete from mtl.book where doi_b = new.doi_b;
    end if;
    return new;
end
$$
    language plpgsql;

create trigger validity_isbn
    after insert
    on mtl.book
    for each row
execute procedure mtl.function_1();

create or replace function mtl.function_2() returns trigger as
$$
declare
    stringa_in  varchar(13) = new.issn_s;
    sum         integer    := 0;
    var_appoggio integer;
    resto       integer;
begin
    stringa_in := replace(stringa_in, '-', '');
    for i in 1..8
        loop
            if substr(stringa_in, 8, 1) = 'X' then
                sum = sum + 10;
            end if;
            var_appoggio = cast(substring(stringa_in from i for 1) as int);
            if (i = 8) then
                sum = sum + 0;
            else
                sum := sum + var_appoggio * (9 - i);
            end if;
        end loop;
    resto = sum % 11;
    if (resto != 0) then
        delete from mtl.series where issn_s = new.issn_s;
    end if;
    return new;
end
$$
    language plpgsql;

create trigger validity_issn_s
    after insert
    on mtl.series
    for each row
execute procedure mtl.function_2();

create or replace function mtl.function_3() returns trigger as
$$
declare
    stringa_in  varchar(13) = new.issn_m;
    sum         integer    := 0;
    var_appoggio integer;
    resto       integer;
begin
    stringa_in := replace(stringa_in, '-', '');
    for i in 1..8
        loop
            if substr(stringa_in, 8, 1) = 'X' then
```

```
            sum = sum + 10;
        end if;
        var_appoggio = cast(substring(stringa_in from i for 1) as int);
        if (i = 8) then
            sum = sum + 0;
        else
            sum := sum + var_appoggio * (9 - i);
        end if;
    end loop;
    resto = sum % 11;
    if (resto != 0) then
        delete from mtl.magazine where issn_m = new.issn_m;
    end if;
    return new;
end
$$
    language plpgsql;

create trigger validity_issn_m
    after insert
    on mtl.magazine
    for each row
execute procedure mtl.function_3();
```

Docum
per la
zione
funzion

Capitolo 5

Caso d'uso e manuale