Progetto corso di Basi Di Dati

Matteo Varotto, Riccardo De Zen

22 Aprile 2020

A.A 2019/2020

Modellizzazione di una libreria

Indice

1 Descrizione del progetto

Si vuole realizzare una base di dati per una libreria che è in grado di vendere al dettaglio oppure ordinare libri per dei clienti; si vuole tenere traccia dei libri con i loro autori, generi, collane ed editori, degli acquisti, dei fornitori e degli ordini effettuati.

2 Requisiti strutturati

Frasi per Dipendente:
Ogni dipendente è identificato univocamente dal codice fiscale e possiede inoltre nel database nome, cognome, stipendio annuale e data di nascita. Un dipendente si occupa di registrare Acquisti ed Ordini.

Frasi per Cliente:
Ogni cliente è identificato univocamente da un ID numerico e possiede inoltre nel database nome, cognome e data di nascita. Ha inoltre almeno uno fra numero di telefono ed e-mail. Nel momento in cui compie l'acquisto, un cliente può scegliere di registrarsi, al fine di essere ricontattato in occasione di offerte future, ma è libero di non farlo. Nel momento in cui effettua una prenotazione è invece costretto a registrarsi, per poter essere ricontattato quando i libri sono disponibili.

Frasi per Acquisto:
Un Acquisto indica una generica transazione monetaria con flusso che parte dal Cliente e va verso libreria. Gli acquisti si differenziano in base alla tempistica della cessione del bene: immediata o successiva. Un Acquisto è univocamente identificato da un numero, ne rappresentiamo inoltre l'importo, la data in cui avviene e il cliente che lo effettua (se è registrato e si identifica). Se l'acquisto non è immediato (chiamato Prenotazione) rappresentiamo inoltre lo stato attuale dei libri (in transito verso la libreria, consegnati in libreria, ritirati dal cliente) e la data di ritiro dei libri da parte del cliente. Un Acquisto viene registrato da un Dipendente, e comprende uno o più libri.
La logistica di evasione delle Prenotazioni in sospeso è a discrezione del personale, che contatta i Clienti quando i libri sono disponibili. Si assume che tutti i libri vengano ritirati in una volta sola, esibendo lo scontrino dell'acquisto.

Frasi per Libro:
Ogni libro è identificato univocamente dal codice standard ISBN. Ne rappresentiamo nel database la quantità con cui è disponibile tra magazzino e scaffali, il prezzo ed il titolo. Un libro viene pubblicato da uno ed un solo Editore, e può appartenere ad una Collana di questo stesso Editore, si vuole inoltre rappresentare la sua data di pubblicazione. Un libro ha uno o più generi che lo caratterizzano e almeno un autore.
Si vuole però tenere traccia dei dati di tutti i libri a cui si ha accesso dalle aziende distributrici con cui si è in contatto, non solo di quelli attualmente presenti sugli scaffali.

Frasi per Collana:
Ogni collana è univocamente identificata dal proprio nome ed ne è rappresentata inoltre la descrizione. Una Collana viene pubblicata da un certo editore e comprende uno o più libri.

Frasi per Editore:
Ogni Editore è identificato univocamente dal proprio nome. Può facoltativamente inoltre presentare una sua descrizione o motto. Ogni editore pubblica almeno un libro. Ogni Editore può pubblicare più collane. Si vuole tenere traccia della data in cui un editore pubblica un libro.

Frasi per Genere:
Ogni genere è identificato univocamente dal nome e possiede inoltre nel database la propria descrizione. Ogni genere descrive uno o più libri.

Frasi per Autore:
Un Autore è univocamente identificato da un codice numerico. Di ogni autore si registra nome, cognome e

data di nascita, quest'ultima può essere ignota. Ogni Autore scrive almeno un Libro.

Frasi per Distributore:
Ogni distributore è univocamente identificato da una partita IVA, e possiede nel database nome e indirizzo della sede principale. Ogni distributore rende disponibili all'acquisto uno o più libri al negozio.

Frasi per Ordine:
Un Ordine di fornitura è univocamente identificato da un numero e possiede inoltre nel database la data in cui viene effettuato. Un ordine viene eseguito da un Dipendente presso un Distributore, su uno solo dei libri che questi rende disponibile. Per tale libro ordinato si vuole tenere traccia di quante copie sono state ordinate e a quale importo totale.

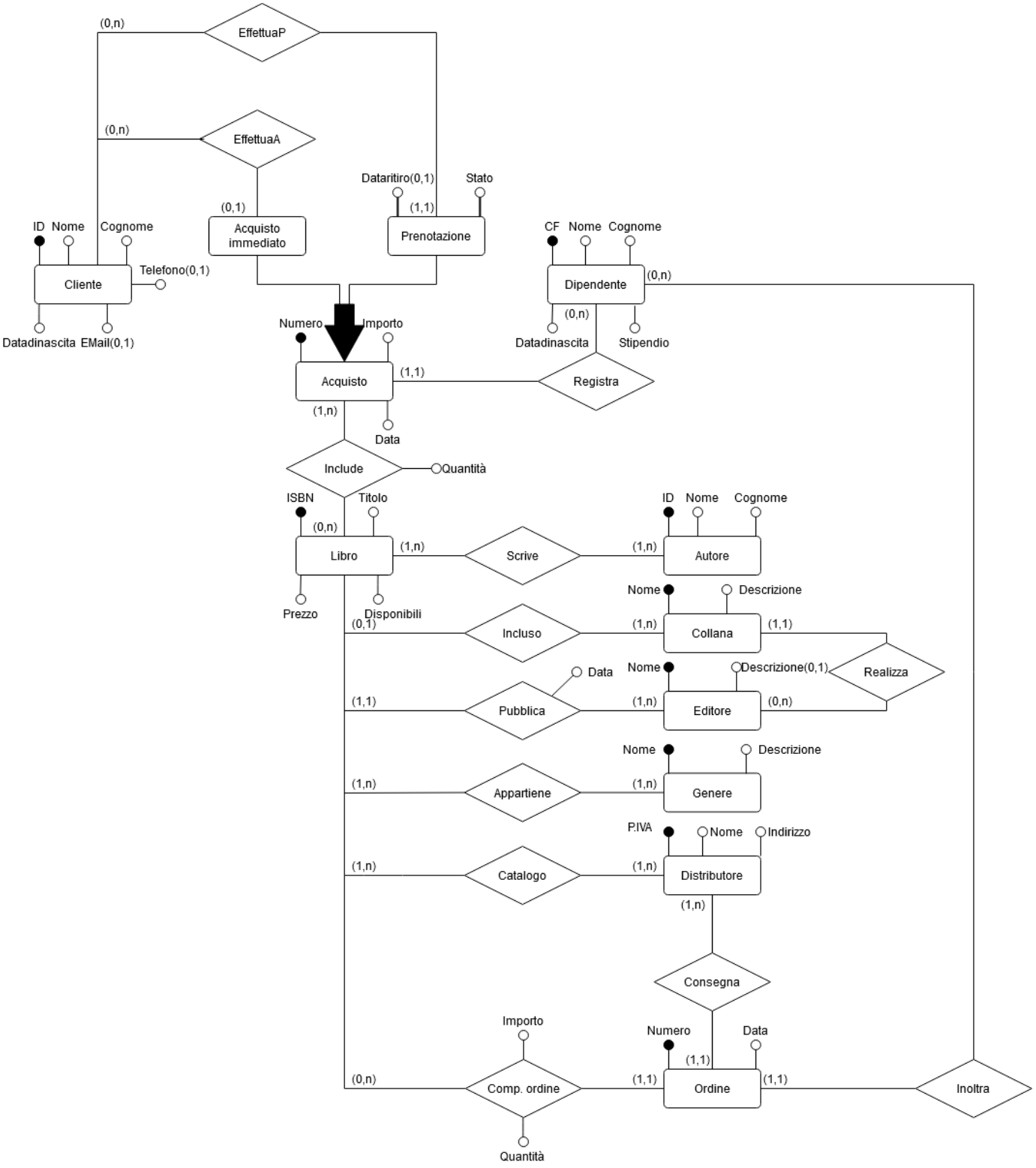
2.1 Operazioni sulla base di dati

Operazione	**Tipo**	**Frequenza**
Calcolo del bilancio per l'anno corrente	interrogazione[1]	1 volta al mese
Il genere di cui sono stati venduti più libri (di sempre)	interrogazione[1]	1 volta al mese
Lista di tutti i libri che appartengono ad un determinato genere	interrogazione[1]	1 volta al mese
Nomi delle case editrici di cui sono stati venduti libri in un certo giorno	interrogazione[1]	1 volta a settimana
Il dipendente che ha registrato più acquisti	interrogazione[1]	1 volta a settimana
Il fornitore che fornisce un libro al minor prezzo	interrogazione[1]	5 volte al giorno
Lista dei generi scritti da un autore	interrogazione[1]	10 volte al giorno
Lista dei generi che caratterizzano i libri di una collana	interrogazione[1]	10 volte al giorno
Aggiornamento dipendenti	modifica	1 volta al mese
Aggiornamento distributori	modifica	1 volta al mese
Aggiunta libri	modifica[2]	1 volta al mese
Aggiornamento generi,collane,autori,editori	modifica	1 volta al mese
Aggiornamento clienti	modifica	1 volta al giorno
Aggiornamento quantità libri disponibili	modifica[1]	1 volta al giorno
Inserimento nuovo cliente	modifica[1]	5 volte al giorno
Inserimento nuovo ordine	modifica[1 2]	10 volte al giorno
Inserimento nuovo acquisto	modifica[1 2]	30 volte al giorno

[1] Implementata nell'applicazione.
[2] Verrà trattata nel dettaglio in questo documento a causa di un coinvolgimento ritenuto non banale con le regole di vincolo.

3 Progettazione concettuale

3.1 Schema E-R



3.2 Dizionario dei dati

3.2.1 Dizionario delle entità

Entità	Attributi	Identificatore
Acquisto	Numero, Importo, Data	Numero
Acquisto immediato	-	Acquisto (ext.) [1]
Prenotazione	Dataritiro, Stato	Acquisto (ext.) [1]
Cliente	ID, Nome, Cognome, Telefono, EMail, Datadinascita	ID
Dipendente	CF, Nome, Cognome, Datadinascita, Stipendio	CF
Libro	ISBN, Titolo, Prezzo, Disponibili	ISBN
Autore	ID, Nome, Cognome	ID
Collana	Nome, Descrizione	Nome
Editore	Nome, Descrizione	Nome
Genere	Nome, Descrizione	Nome
Distributore	P.IVA, Nome, Indirizzo	P.IVA
Ordine	Numero, Data	Numero

[1] L'identificatore è implicito a causa della generalizzazione. É solo a seguito della ristrutturazione che diventa esplicito.

3.2.2 Dizionario delle associazioni

Associazione	Attributi	Entità collegate
EffettuaA	-	Acquisto immediato (0,1), Cliente (0,n)
EffettuaP	-	Prenotazione (1,1), Cliente (0,n)
Registra	-	Dipendente (0,n), Acquisto (1,1)
Include	Quantità	Acquisto (1,n), Libro (0,n)
Scrive	-	Autore (1,n), Libro (1,n)
Incluso	-	Collana (0,1), Libro (1,1)
Pubblica	Data	Editore (1,n), Libro (1,n)
Appartiene	-	Libro (1,n), Genere (1,n)
Catalogo	-	Libro (1,n), Distributore (1,n)
Consegna	-	Distributore (1.n), Ordine (1,1)
Comp. ordine	Importo, Quantità	Libro (0,n), Ordine (1,1)
Inoltra	-	Dipendente (0,n), Ordine (1,1)
Realizza	-	Collana (1,1), Editore (0,n)

3.2.3 Regole aggiuntive

- Un Ordine può richiedere ad un certo Distributore solo Libri presenti nel suo catalogo.
- Un Libro può appartenere solo ad una Collana realizzata dall'Editore che lo pubblica.
- Almeno un attributo tra "Telefono" e "EMail" di ogni istanza dell'entità Cliente non deve essere nullo.
- Gli attributi "Prezzo", "Importo", "Quantità", "Stipendio", "Disponibili" dello schema devono essere maggiori o uguali a zero. In particolare, "Quantità" e "Stipendio" devono essere strettamente maggiori di zero.

4 Progettazione logica

4.1 Eliminazione della generalizzazione Acquisto

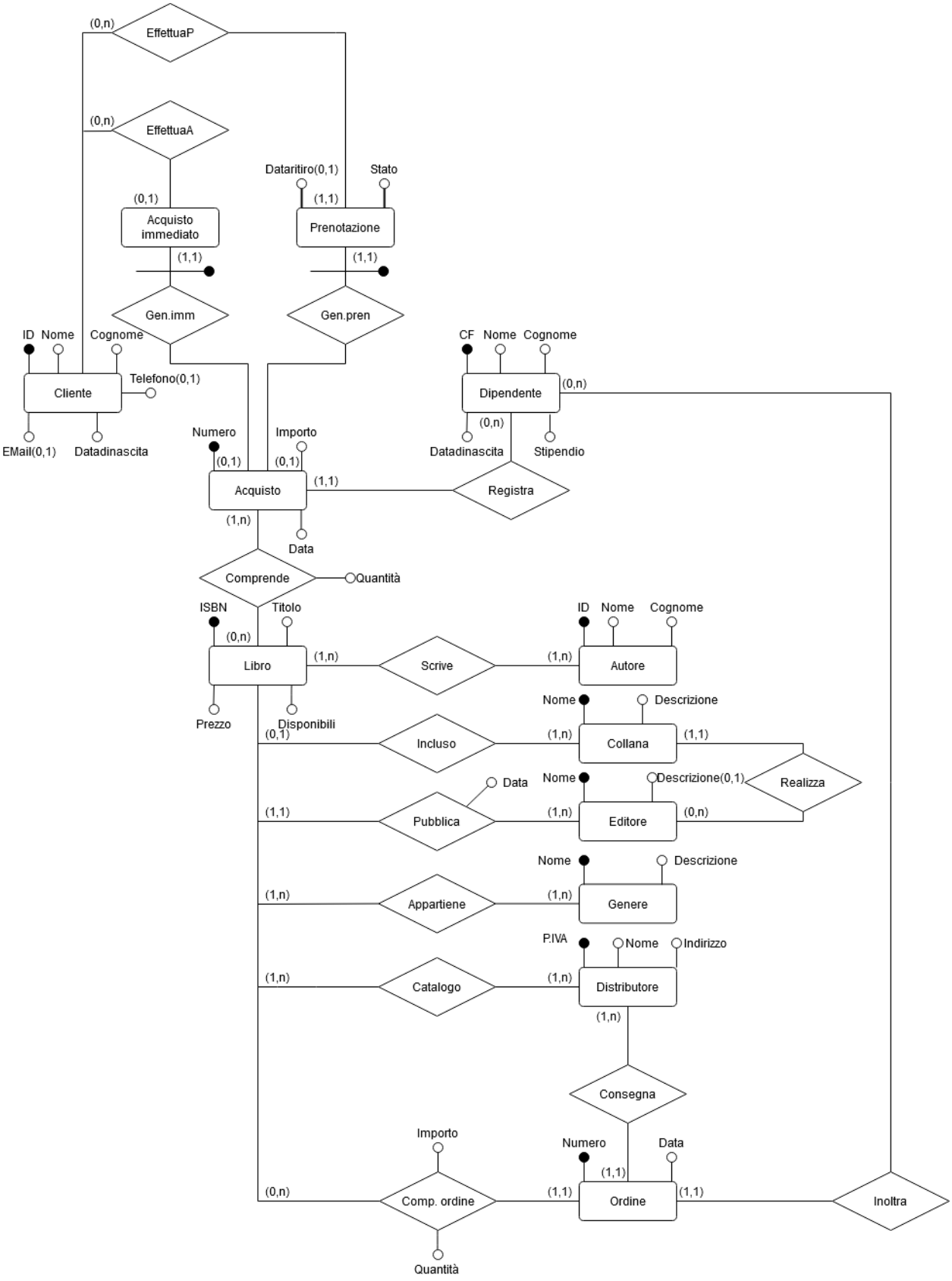
Per la rimozione della generalizzazione Acquisto, si è optato per la divisione in tabelle separate dei figli, mantenendo la tabella genitrice con gli attributi comuni. Questa scelta è stata fatta per tre motivi principali:

- (A favore della soluzione proposta) Leggibilità e modularità dei dati, poiché ogni istanza di Acquisto si occupa di tenere traccia solo degli aspetti economici di una operazione, senza preoccuparsi della risoluzione temporale della parte patrimoniale dello scambio, cioè della cessione dei beni.
- (Contro una divisione completa delle tabelle) Semplificazione delle operazioni di interrogazione sulla base di dati, poiché vi è una sola tabella (Comprende) che si occupa di tenere traccia dei libri inclusi in un acquisto, di nuovo senza preoccuparsi della risoluzione temporale. Questo è parso preferibile rispetto alla divisione in due tabelle distinte che non siano deboli verso una tabella comune, dato che rimuove la necessità di eseguire un unione insiemistica tra le due tabelle per eseguire operazioni che richiedano un conteggio o verifica dei libri acquistati, rendendo le query più intuitive e leggibili.
- (Contro un attributo "tipo" in Acquisto) L'introduzione di un attributo "tipo" in Acquisto comporterebbe una quantità di attributi nulli non trascurabile all'interno delle righe di Acquisto, anche in virtù dei valori riportati nella tabella dei volumi che segue.

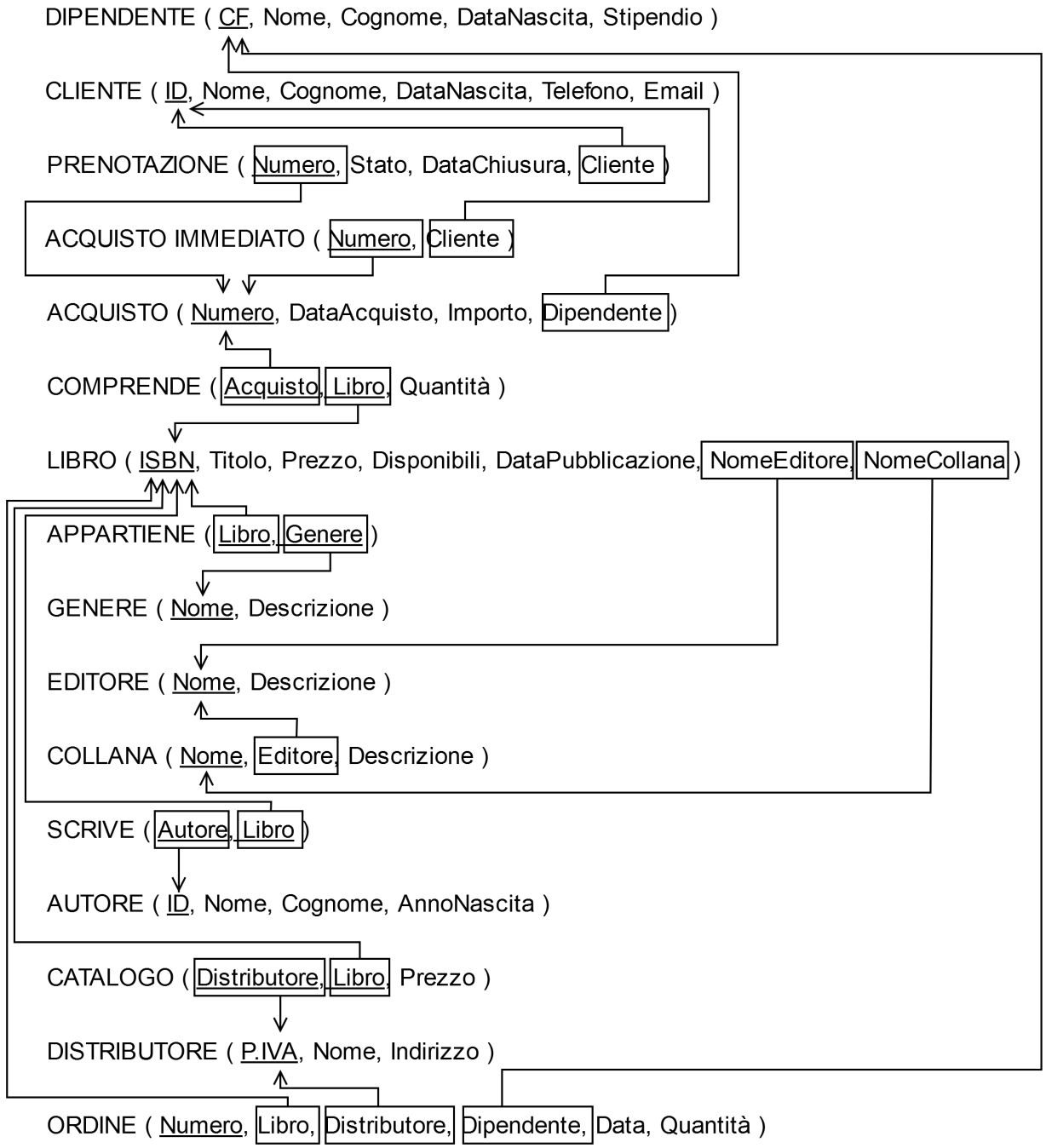
4.2 Volumi e rinominazione

Concetto	**Tipo**	**Volume**
Dipendente	Entità	10
Cliente	Entità	~ 500
Libro	Entità	~ 10000
Acquisto	Entità	~ 5000
Acquisto Immediato	Entità	~ 4000
Prenotazione	Entità	~ 1000
Autore	Entità	~ 5000
Editore	Entità	~ 50
Collana	Entità	~ 500
Genere	Entità	~ 50
Distributore	Entità	~ 10
Ordine	Entità	~ 1000
Comprende	Associazione	~ 15000
Scrive	Associazione	~ 20000
Appartiene	Associazione	~ 20000
Catalogo	Associazione	~ 15000

4.3 Schema E-R ristrutturato



4.4 Schema Logico



4.5 Regole di vincolo

4.5.1 Vincoli di dominio

- **RV1:** Almeno un attributo tra "Telefono" e "EMail" di ogni istanza dell'entità Cliente non deve essere nullo.
- **RV2:** Gli attributi "Prezzo", "Importo", "Quantità", "Stipendio", "Disponibili" dello schema devono

essere maggiori o uguali a zero. In particolare, "Quantità" e "Stipendio" devono essere strettamente maggiori di zero.

- **RV3:** L'attributo Stato in Prenotazione è parte di una enumerazione con valori "Transito", "Consegnato", "Ritirato". Solo una Prenotazione con Stato = "Ritirato" dovrebbe avere una DataChiusura non nulla.

4.5.2 Vincoli di Integrità Referenziale

- **RV4:** Ogni istanza di Prenotazione deve avere il valore di "Cliente" non nullo.
- **RV5:** Ogni istanza di Acquisto deve avere il valore di "Dipendente" non nullo.
- **RV6:** Ogni istanza di Acquisto deve partecipare ad almeno un'istanza di Comprende.
- **RV7:** Ogni istanza di Libro deve partecipare al massimo ad un'istanza di Scrive,così come ogni istanza di Autore.
- **RV8:** Ogni istanza di Libro deve avere il valore di "NomeEditore" non nullo,inoltre deve partecipare ad almeno ad un'istanza di Appartiene e Catalogo.
- **RV9:** Ogni istanza di Genere deve partecipare ad almeno un'istanza di Appartiene.
- **RV10:** Ogni istanza di Distributore deve partecipare ad almeno un'istanza di Catalogo.
- **RV11:** Ogni istanza di Collana deve avere il valore di "Editore" non nullo.
- **RV12:** Ogni istanza di Ordine deve avere i valori di "Distributore","Libro" e "Dipendente" non nulli contemporaneamente.

4.5.3 Vincoli aggiuntivi

- **RV13:** Per ogni Acquisto A deve esistere, nell'insieme delle istanze di AcquistoImmediato e Prenotazione, un solo esemplare con attributo Numero che si riferisce ad A.
- **RV14:** Quando viene creato un Ordine, deve essere esistente in tale momento una istanza di Catalogo tale per cui gli attributi Distributore e Libro coincidono con quelli dell'Ordine.
- **RV15:** Per ogni Libro con attributo Collana non nullo, l'istanza di Collana corrispondente possiede in NomeEditore il valore dell'attributo Editore del Libro.

5 Implementazione SQL

5.1 Domini per i dati particolari

Si è ritenuto sensato restringere i domini di alcuni tipi di dati mediante espressioni regolari.
Ciò non costituisce una scelta eccessivamente vincolante, poiché il dominio può essere espanso alla sua versione meno restrittiva (varchar/text etc.) in qualunque momento senza causare alcuna perdita di dati.

```
CREATE DOMAIN tipo_codice_fiscale AS CHAR(16)
CHECK (VALUE SIMILAR TO '[A-z0-9]{16}');

CREATE DOMAIN tipo_codice_isbn AS CHAR(13)
CHECK (VALUE SIMILAR TO '[0-9]{13}');

CREATE DOMAIN tipo_numero_telefono AS character varying(15)
CHECK (VALUE SIMILAR TO '[0-9]{3,15}');
```

```
CREATE DOMAIN tipo_partita_iva AS CHAR(11)
CHECK (VALUE SIMILAR TO '[0-9]{11}');

CREATE DOMAIN tipo_email AS TEXT
CHECK (VALUE SIMILAR TO '[A-Za-z0-9.]{1,}@[A-Za-z0-9.]{1,}');

CREATE TYPE stato_prenotazione AS ENUM ('Transito', 'Consegnato', 'Ritirato');
```

5.2 Tabelle

```
CREATE TABLE a_immediato (
    numero integer NOT NULL,
    cliente integer,
    CONSTRAINT immediato_pkey PRIMARY KEY (numero),
    CONSTRAINT immediato_fkey_acquisto FOREIGN KEY (numero)
        REFERENCES acquisto (numero) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT immediato_fkey_cliente FOREIGN KEY (cliente)
        REFERENCES cliente (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE acquisto
(
    numero SERIAL,
    data_acquisto date NOT NULL,
    importo numeric(11,2) NOT NULL,
    dipendente tipo_codice_fiscale NOT NULL,
    CONSTRAINT acquisto_pkey PRIMARY KEY (numero),
    CONSTRAINT acquisto_dipendente_fkey FOREIGN KEY (dipendente)
        REFERENCES dipendente (cf) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT acquisto_importo_positivo CHECK (importo >= 0)
);

CREATE TABLE appartiene
(
    libro tipo_codice_isbn NOT NULL,
    genere character varying(20) NOT NULL,
    CONSTRAINT appartiene_pkey PRIMARY KEY (libro, genere),
    CONSTRAINT appartiene_libro_fkey FOREIGN KEY (libro)
        REFERENCES libro (isbn) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT appartiene_genere_fkey FOREIGN KEY (genere)
        REFERENCES genere (nome) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE autore
(
```

```
    id SERIAL,
    nome character varying(20) NOT NULL,
    cognome character varying(20) NOT NULL,
    data_nascita date,
    CONSTRAINT autore_pkey PRIMARY KEY (id)
);

CREATE TABLE catalogo
(
    distributore tipo_partita_iva NOT NULL,
    libro tipo_codice_isbn NOT NULL,
    prezzo numeric(11,2) NOT NULL,
    CONSTRAINT catalogo_pkey PRIMARY KEY (distributore, libro),
    CONSTRAINT catalogo_distributore_fkey FOREIGN KEY (distributore)
        REFERENCES distributore (piva) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT catalogo_libro_fkey FOREIGN KEY (libro)
        REFERENCES libro (isbn) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE cliente
(
    id SERIAL,
    nome character varying(20) NOT NULL,
    cognome character varying(20) NOT NULL,
    data_nascita date NOT NULL,
    telefono tipo_numero_telefono,
    email tipo_email,
    CONSTRAINT cliente_pkey PRIMARY KEY (id),
    CONSTRAINT informazione_di_contatto CHECK (telefono IS NOT NULL OR email IS NOT NULL)
);

CREATE TABLE collana
(
    nome character varying(50) NOT NULL,
    nome_editore character varying(50) NOT NULL,
    descrizione text,
    CONSTRAINT collana_pkey PRIMARY KEY (nome),
    CONSTRAINT collana_nome_editore_fkey FOREIGN KEY (nome_editore)
        REFERENCES editore (nome) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE comprende
(
    libro tipo_codice_isbn NOT NULL,
    acquisto integer NOT NULL,
    quantita smallint NOT NULL,
    CONSTRAINT include_pkey PRIMARY KEY (libro, acquisto),
    CONSTRAINT include_acquisto_fkey FOREIGN KEY (acquisto)
        REFERENCES acquisto (numero) MATCH SIMPLE
```

```
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT include_libro_fkey FOREIGN KEY (libro)
        REFERENCES libro (isbn) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT quantita_positiva CHECK (quantita > 0)
);

CREATE TABLE dipendente
(
    cf tipo_codice_fiscale NOT NULL,
    nome character varying(20) NOT NULL,
    cognome character varying(20) NOT NULL,
    data_nascita date NOT NULL,
    stipendio numeric(11,2) NOT NULL,
    CONSTRAINT dipendente_pkey PRIMARY KEY (cf),
    CONSTRAINT stipendio_positivo CHECK (stipendio > 0)
);

CREATE TABLE distributore
(
    piva tipo_partita_iva NOT NULL,
    nome character varying(50) NOT NULL,
    indirizzo character varying(50) NOT NULL,
    CONSTRAINT distributore_pkey PRIMARY KEY (piva)
);

CREATE TABLE editore
(
    nome character varying(50) NOT NULL,
    descrizione text,
    CONSTRAINT editore_pkey PRIMARY KEY (nome)
);

CREATE TABLE genere
(
    nome character varying(20) NOT NULL,
    descrizione text,
    CONSTRAINT genere_pkey PRIMARY KEY (nome)
);

CREATE TABLE libro
(
    isbn tipo_codice_isbn NOT NULL,
    titolo character varying(50) NOT NULL,
    prezzo numeric(11,2) NOT NULL,
    disponibili smallint NOT NULL,
    data_pubblicazione date NOT NULL,
    nome_editore character varying(50) NOT NULL,
    nome_collana character varying(50),
    CONSTRAINT libro_pkey PRIMARY KEY (isbn),
    CONSTRAINT libro_nome_editore_fkey FOREIGN KEY (nome_editore)
        REFERENCES editore (nome) MATCH SIMPLE
        ON UPDATE CASCADE
```

```
        ON DELETE RESTRICT,
    CONSTRAINT libro_nome_collana_fkey FOREIGN KEY (nome_collana)
        REFERENCES collana (nome) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT prezzo_positivo CHECK (prezzo > 0),
    CONSTRAINT disponibili_non_negativo CHECK (disponibili >= 0)
);

CREATE TABLE ordine
(
    numero SERIAL,
    data_ordine date NOT NULL,
    quantita smallint NOT NULL,
    importo numeric(11,2) NOT NULL,
    libro tipo_codice_isbn NOT NULL,
    distributore tipo_partita_iva NOT NULL,
    dipendente tipo_codice_fiscale NOT NULL,
    CONSTRAINT ordine_pkey PRIMARY KEY (numero),
    CONSTRAINT ordine_dipendente_fkey FOREIGN KEY (dipendente)
        REFERENCES dipendente (cf) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT ordine_libro_fkey FOREIGN KEY (libro)
        REFERENCES libro (isbn) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT ordine_distributore_fkey FOREIGN KEY (distributore)
        REFERENCES distributore (piva) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT ordine_quantita_positiva CHECK (quantita > 0),
    CONSTRAINT ordine_importo_positivo CHECK (importo >= 0)
);

CREATE TABLE prenotazione
(
    numero integer NOT NULL,
    stato stato_prenotazione NOT NULL,
    cliente integer NOT NULL,
    data_ritiro date,
    CONSTRAINT prenotazione_pkey PRIMARY KEY (numero),
    CONSTRAINT prenotazione_fkey_acquisto FOREIGN KEY (numero)
        REFERENCES acquisto (numero) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT prenotazione_fkey_cliente FOREIGN KEY (cliente)
        REFERENCES cliente (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT prenotazione_data_se_chiusa CHECK (
        (stato = 'Ritirato' AND data_ritiro IS NOT NULL) OR
        (stato <> 'Ritirato' AND data_ritiro IS NULL)
    )
);
```

```
CREATE TABLE scrive
(
    autore integer NOT NULL,
    libro tipo_codice_isbn NOT NULL,
    CONSTRAINT scrive_autore_fkey FOREIGN KEY (autore)
        REFERENCES autore (id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    CONSTRAINT scrive_libro_fkey FOREIGN KEY (libro)
        REFERENCES libro (isbn) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);
```

5.3 Interrogazioni

Le interrogazioni sono state implementate, nel concreto, sotto forma di "Funzioni", costrutto specifico di PostgreSQL. La scelta è stata fatta in modo da poter utilizzare, nell'applicazione, un approccio quanto più possibile "a scatola chiusa". Questo non altera in nessun modo la struttura delle query, poiché vengono semplicemente racchiuse nella dichiarazione della funzione e degli eventuali parametri.

Query 1: calcolo del bilancio di un anno, inteso come la differenza tra la somma di tutti i ricavi e la somma di tutte le spese(acquisto libri e stipendi).

```
SELECT
(SELECT SUM(importo) FROM acquisto WHERE date_part('year', data_acquisto) = XXXX) -
(SELECT SUM(importo) FROM ordine WHERE date_part('year', data_ordine) = XXXX) -
(SELECT SUM(stipendio) FROM dipendente)
AS "Bilancio";
```

Query 2: trovare il genere più venduto e restituirne anche nome,descrizione e quantità venduta di libri.

```
CREATE VIEW vendite_per_genere AS (
    SELECT AP.genere AS genere, SUM(CO.quantita) AS quantita
    FROM (
        acquisto AS AQ
        JOIN comprende AS CO
        ON AQ.numero = CO.acquisto
        JOIN appartiene AS AP
        ON AP.libro = CO.libro
    )
    GROUP BY genere
);

SELECT G.nome AS "Genere più venduto", G.descrizione AS "Descrizione", V.quantita AS "Vendite"
FROM vendite_per_genere AS V
JOIN genere AS G
ON G.nome = V.genere
WHERE V.quantita IN (
    SELECT MAX(quantita) FROM vendite_per_genere
);
```

Query 3: trovare tutti i libri appartenenti a un determinato genere.

```
SELECT DISTINCT L.isbn AS "ISBN", L.titolo AS "Titolo", L.nome_collana AS "Collana"
FROM appartiene AS A
```

```
JOIN libro AS L
ON A.libro = L.isbn
WHERE A.genere = XXXX;
```

Query 4: trovare i nomi delle case editrici che hanno venduto libri in un giorno.

```
SELECT DISTINCT L.nome_editore AS "Casa Editrice"
FROM libro AS L
JOIN comprende AS C
ON L.isbn = C.libro
JOIN acquisto AS A
ON C.acquisto = A.numero
WHERE A.data_acquisto = XXXX;
```

Query 5: trovare il dipendente che ha registrato più acquisti

```
CREATE VIEW acquisti_per_dipendente AS (
    SELECT dipendente, COUNT(*) AS acquisti
    FROM acquisto
    GROUP BY dipendente
);

SELECT A.dipendente AS "Codice Fiscale", D.nome AS "Nome", D.cognome AS "Cognome"
FROM acquisti_per_dipendente AS A
JOIN dipendente AS D
ON A.dipendente = D.cf
WHERE acquisti IN (SELECT MAX(acquisti) FROM acquisti_per_dipendente);
```

Query 6: trovare il fornitore che fornisce un libro al minor prezzo

```
SELECT distributore AS "Distributore", prezzo AS "Prezzo"
FROM catalogo
WHERE libro = XXXX
AND prezzo IN (
    SELECT MIN(prezzo)
    FROM catalogo
    WHERE libro = XXXX
);
```

Query 7: trovare tutti i generi scritti da un determinato autore.

```
SELECT DISTINCT genere AS "Genere"
FROM scrive NATURAL JOIN appartiene
WHERE autore = XXXX;
```

Query 8: trovare tutti i generi racchiusi in una determinata collana.

```
SELECT DISTINCT genere AS "Genere"
FROM libro JOIN appartiene
ON libro.isbn = appartiene.libro
WHERE nome_collana = XXXX;
```

6 Inserimenti e Vincoli

Tutte le regole di vincolo sono gestite tramite i costrutti SQL CONSTRAINT e DOMAIN, ad eccezione delle tre regole di vincolo citate alla sottosezione **Vincoli aggiuntivi**. Di seguito verranno trattati brevemente gli inserimenti contrassegnati alla sezione **Operazioni sulla base di dati**, poiché il loro coinvolgimento con

le sopracitate regole di vincolo non è banale.

Inserimento Libro
Ribadiamo la regola di vincolo in questione (RV 15): *Per ogni Libro con attributo Collana non nullo, l'istanza di Collana corrispondente possiede in NomeEditore il valore dell'attributo Editore del Libro.*.
Sia dato il Libro L, appartenente alla Collana C e pubblicato dall'Editore E. L rispetta la RV 15 se e solo se Il risultato dell'interrogazione:

```
SELECT nome_editore FROM collana WHERE nome = 'C';
```

contiene E.

Inserimento Ordine
Ribadiamo la regola di vincolo in questione (RV 14): *Quando viene creato un Ordine, deve essere esistente in tale momento una istanza di Catalogo tale per cui gli attributi Fornitore e Libro coincidono con quelli dell'Ordine. TESTO DELLA RV*
Sia dato l'Ordine O, che riguarda il rifornimento del Libro L presso il Distributore D. L'Ordine O rispetta la RV 14 se e solo se Il risultato dell'interrogazione:

```
SELECT libro FROM catalogo WHERE distributore = 'F';
```

contiene "L".

Inserimento Acquisto
Ribadiamo la regola di vincolo in questione (RV 13): *Per ogni Acquisto* A *deve esistere, nell'insieme delle istanze di AcquistoImmediato e Prenotazione, un solo esemplare con attributo Numero che si riferisce ad* A.
Per rispettare tale regola, è sufficiente inserire sempre coppie di tuple in *acquisto* - *a_immediato* o *acquisto* - *prenotazione* (in base al caso) atomicamente, come singola transazione.

1. Inserimento della tupla in *acquisto*, lasciando l'assegnazione del numero acquisto al database e recuperandolo tramite *RETURNING numero*

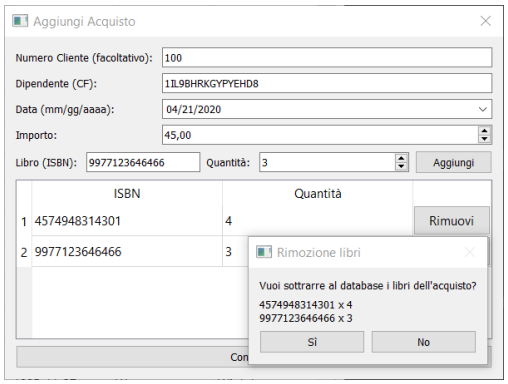
```
INSERT INTO acquisto (data_acquisto, importo, dipendente) VALUES (X, X, X) RETURNING numero;
```

2. Inserimento della tupla appropriata in *a_immediato*, dove *NUM* è il risultato dell'operazione 1:

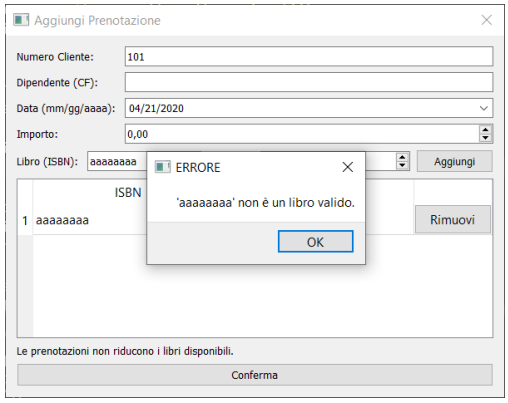
```
INSERT INTO a_immediato (numero, cliente) VALUES (NUM, X, X);
```

O in *prenotazione* (assumendo che sia una nuova prenotazione, i cui libri non sono stati ritirati e la cui data di conclusione sia quindi nulla):

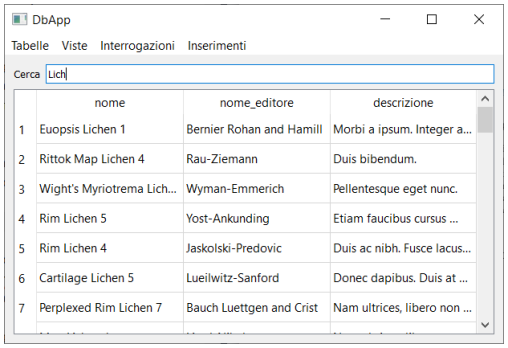
```
INSERT INTO prenotazione (numero, cliente, stato) VALUES (NUM, X, 'Transito');
```



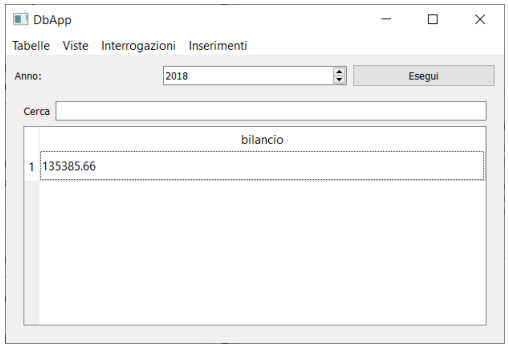
(a) Inserimento di un Acquisto Immediato, in fase di conferma della riduzione dei libri.



(b) Verifica dei dati prima dell'inserimento.



(a) Le tabelle possono essere filtrate su qualsiasi colonna (simil-full-text search).



(b) I risultati delle interrogazioni vengono trattati come tabelle.

7 Applicazione

7.1 Struttura e caratteristiche

L'applicazione mira a fornire tutte le operazioni indispensabili per poter essere usata all'interno di un contesto pratico quotidiano nella libreria. Pertanto implementa i quattro inserimenti menzionati alla sottosezione **Operazioni sulla base di dati**: Cliente, Acquisto Immediato, Prenotazione, Ordine. Acquisto Immediato, Prenotazione e Ordine sono implementati secondo il rispetto delle regole di vincolo, come descritto alla sezione precedente. Inoltre, viene data particolare attenzione al rispetto dei domini dei dati, in particolare l'inserimento di Acquisto Immediato controlla che le quantità di libri acquistate siano inferiori o uguali alle quantità di libri disponibili, e se lo sono, offre la possibilità, dopo aver inserito le tuple nelle tabelle, di aggiornare il valore dell'attributo Disponibili nelle tuple di Libro coinvolte nell'Acquisto, tramite la seguente query:

```
-- Per ogni libro Li, acquistato in quantità Qi
UPDATE libro SET disponibili = (
    (SELECT disponibili FROM libro WHERE isbn = 'Li') - Qi
) WHERE isbn = 'Li';
```

Se la quantità non è disponibile, non impedisce la registrazione dell'Acquisto, ma avvisa che non sarà possibile scalare la quantità acquistata se si prosegue. Cliente è la più banale: si limita a verificare che i domini dei dati siano rispettati.

Oltre agli inserimenti, l'applicazione permette di consultare il contenuto di tutte le tabelle (incluse le viste create a supporto delle interrogazioni), e di eseguire le otto interrogazioni descritte in precedenza, con possibilità di filtrare i le tabelle e i risultati in base a una stringa (su tutte le colonne).

L'applicazione permette di connettersi ad una arbitraria copia del database, locale o remota, a patto che

non vi siano server intermediari, o alla versione caricata su dbstud, nel database *dezenricca* via ssh (solo attraverso *login.dei.unipd.it*, poichè non è stato possibile testare all'interno della rete DEI).

7.2 Strumenti utilizzati

- Python 3.8.2 (Windows 64-bit)
- Librerie Python: sshtunnel (0.1.5), psycopg2 (2.8.4), PySide2 (5.14.2)
- Overleaf (Editor Latex)
- PostgreSQL e PgAdmin

7.3 Note

- L'applicazione è sviluppata in Python. Può essere eseguita partendo dal sorgente Python su una qualsiasi macchina che lo supporti, previa installazione dei package richiesti. Questi sono inclusi in un file di testo all'interno della repository GitHub con il materiale.
- È stata compilata una versione per Windows a 64 bit, tuttavia viene mostrata ugualmente una finestra del terminale, a causa di un difetto con l'opzione per nasconderlo dei compilatori.

Tutto il materiale del progetto, compresa l'applicazione e un backup del database, è disponibile all'interno della seguente repository GitHub: https://github.com/RicDeZen/dezen-varotto-db