UNIVERSITA' DEGLI STUDI DI NAPOLI
FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI BASI DI DATI I ANNO ACCADEMICO 2019/2020

Progettazione e sviluppo di una base di dati relazionale per la gestione di un negozio di abbigliamento che permetta di gestire vendite e disponibilità di magazzino

Autori:

Lorenzo Lama : N86002845

Aleks Nikolaev Nikolov : N86003002

Docente:

Prof. Adriano Peron

Descrizione del progetto

Introduzione

Per cominciare, andremo ad individuare le entità rilevanti del problema necessarie a soddisfare le esigenze della traccia, senza curarci della particolare implementazione che andremo poi ad adottare.

Individueremo le associazioni tra queste entità, gli attributi che le caratterizzano e eventuali enumerazioni adatte ad alcuni di essi, andando così a formare il nostro class diagram.

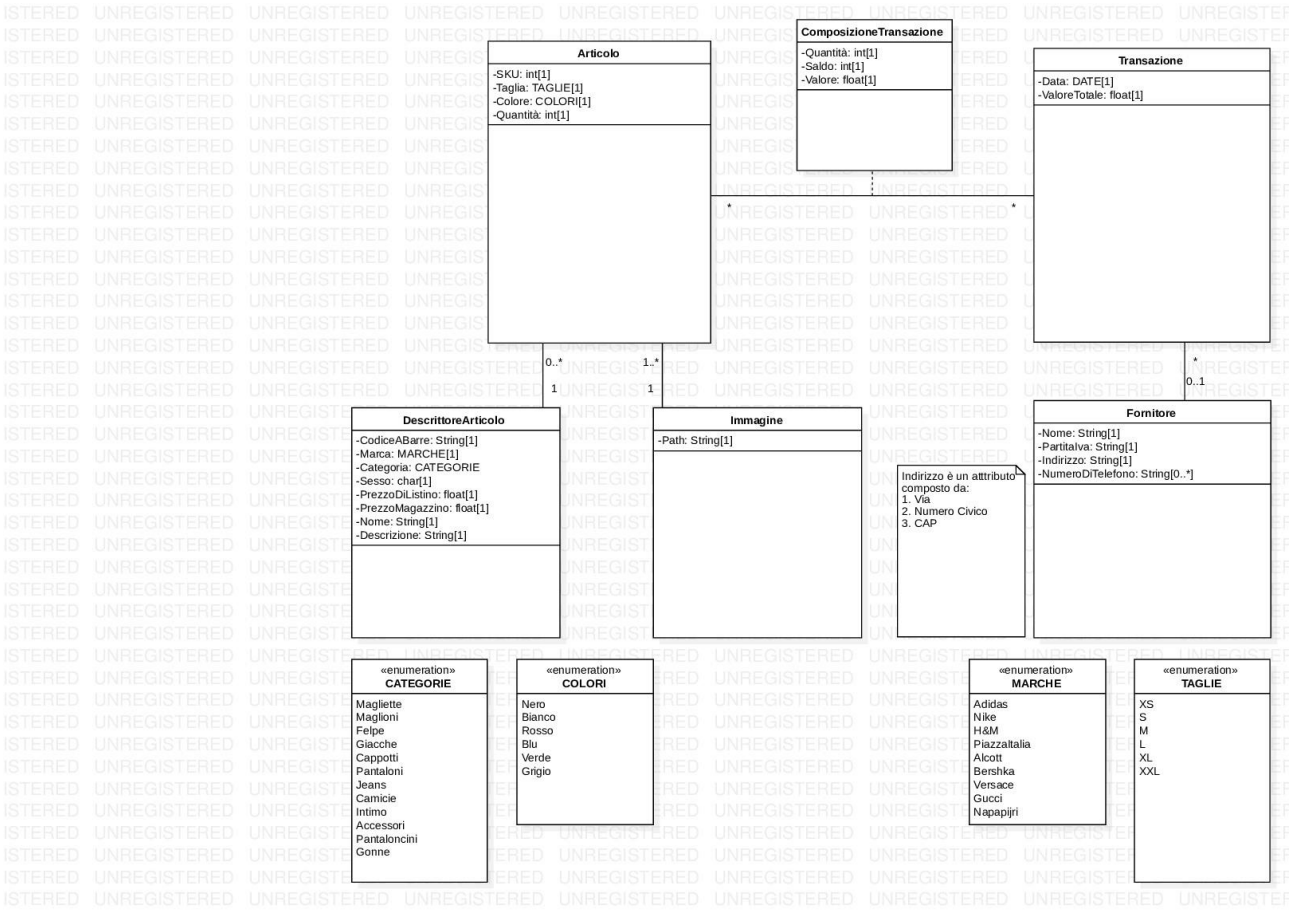
Successivamente, entreremo nella fase di ristrutturazione del class diagram, con eventuali analisi delle ridondanze, scomposizione di attributi composti o con molteplicità , sostituzione delle specializzazioni e introduzione di nuovi vincoli, in modo da prepararci alla traduzione in schema logico.

Una volta formato quest'ultimo, andremo a produrre il codice sql equivalente con la creazione delle tabelle e vincoli inerenti ai loro attributi.

Infine, andremo a produrre eventuali triggers procedure e viste utili al nostro database e/o applicazione java

Progettazione concettuale

Class Diagram



Fase di ristrutturazione

Nella classe “Fornitore” notiamo la presenza di due attributi problematici: “Indirizzo” e “NumeroDiTelefono”. Il primo è un attributo composto, il secondo ha molteplicità maggiore di 1.

Per quanto riguarda “Indirizzo”andremo a scomporlo nei suoi componenti Via, Numero civico e CAP nella classe fornitore. Decidiamo invece per quanto riguarda “NumeroDiTelefono” che il nostro negozio abbia al massimo un recapito telefonico.

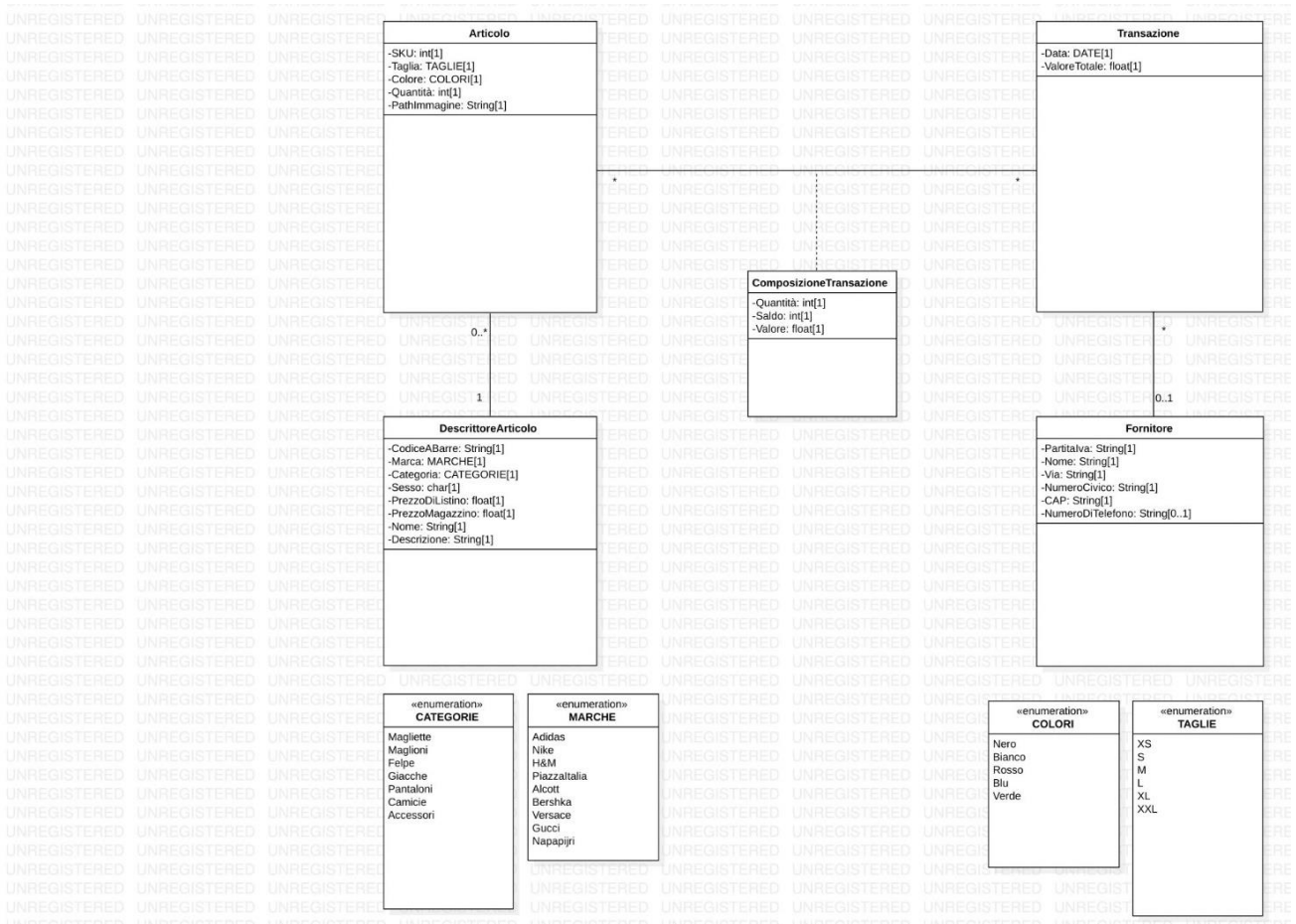
Non ci sono specializzazioni che ci possano infastidire.

Facciamo però la considerazione che per l’utilizzo che se ne prevede in questo contesto, non è particolarmente importante separare la classe “Articolo” da “Immagine”, specialmente considerando la cardinalità dell’associazione “Articolo-“Immagine” e che “Immagine” ha un solo attributo. Decidiamo quindi di collassare “Immagine” su “Articolo” perchè ciò risulta generalmente più comodo e non comporta particolari svantaggi.

Vediamo il class diagram ristrutturato:

Progettazione concettuale

Class Diagram Ristrutturato



Dizionario dei dati

Dizionario delle classi

Classe	***Attributi***	***Descrizione***
Articolo	*SKU*	*Lo "stock keeping unit" è un codice unico assegnato a un articolo per identificarlo in inventario*
	Taglia	*La taglia di un particolare articolo, il cui valore è uno di quelli che appare nell'enumerazione TAGLIE*
	Colore	*Il colore di un particolare articolo, il cui valore è uno di quelli che appare nell'enumerazione COLORI*
	Quantità	*Quantità del particolare articolo in magazzino. Non può essere minore di 0*
	PathImmagine	*Il path dell'immagine associata al particolare articolo. Ogni articolo ha una e una sola immagine(path dell'immagine "no image available" come default)*
DescrittoreArticolo	*CodiceABarre*	*Codice a barre di un prodotto*
	Marca	*Marca del particolare prodotto, il cui valore è uno di quelli che appare nell'enumerazione MARCHE*
	Categoria	*Categoria a cui appartiene il particolare prodotto, il cui valore è uno di quelli che appare nell'enumerazione CATEGORIE*
	Sesso	*Specifica se il prodotto è per uomini, donne, o entrambi. Può assumere valore 'M' , 'F', o 'U'*
	PrezzoDiListino	Il prezzo di listino di un particolare prodotto

Dizionario dei dati

Continuo dizionario delle classi

Classe	***Attributi***	***Descrizione***
//classe precedente	*PrezzoMagazzino*	*Prezzo del prodotto se acquistato da un fornitore*
	Nome	*Il nome di un particolare prodotto*
	Descrizione	*Breve descrizione del prodotto*
Transazione	*Data*	*Data in cui è avvenuta la transazione*
	ValoreTotale	*Uguale alla somma di ComposizioneTransazione.Valore di tutti gli articoli in quella transazione*
Fornitore	*PartitaIva*	*La partitaIva del fornitore*
	Nome	*Nome della ditta/azienda del particolare rifornimento*
	Via	*Via ove risiede la sede principale del fornitore*
	NumeroCivico	*Numero Civico della via ove risiede la sede principale del fornitore*
	CAP	*Codice avviamento postale del comune ove risiede la sede principale del fornitore*
	NumeroDiTelefono	*Numero di telefono di riferimento del fornitore*

Dizionario dei dati

Continuo dizionario delle classi

Classe	*Attributi*	*Descrizione*
ComposizioneTransazione	*Quantità*	*Indica quante unità di articolo vi siano in una transazione*
	Saldo	*Percentuale di saldo corrente dell'articolo. Va da 0 a 100, ha 0 come valore di default*
	Valore	*Uguale a quantità * (PrezzoDiListino – (Prezzodilistino / saldo)) se la nuova istanza di composizionetransazione è una vendita,PrezzoMagazzino se un rifornimento.*

Dizionario delle associazioni

Associazione	*Descrizione*
Articolo-DescrittoreArticolo	*Indica da un lato quale descrittore sia associato all'articolo, dall'altro quali articoli abbiano quel descrittore*
Transazione-Fornitore	*Indica da un lato quale sia il fornitore di quella transazione, se c'è (e quindi la transazione è un rifornimento e non una vendita), dall'altro quali rifornimenti siano stati effettuati tramite quel fornitore*
Articolo-Transazione	*Indica da un lato, tramite la classe di associazione "ComposizioneTransazione", di quali articoli sia composta una particolare transazione, e dall'altro in quale transazione siano coinvolti particolari articoli*

Dizionario dei dati

Dizionario dei vincoli

Vincolo	***Descrizione***
valoreQuantitàArticolo	*La quantità di un articolo non può essere minore di 0*
valoreSaldo	*Il valore di saldo non può essere minore di 0 o maggiore di 100*
valoreQuantiàComposizioneTransazione	*La quantità di un articolo in una vendita è > 0 e <= di Articolo.quantità*
valoreValore	*Valore deve essere maggiore o uguale a 0*
uNumeroDiTelefono	*I numeri di telefono sono unici*
valoreValoreTotale	*ValoreTotale deve essere maggiore o uguale a 0*
valorePrezzoDiListino	*Il valore di PrezzoDiListino deve essere maggiore o uguale a 0*
valorePrezzoMagazzino	*Il valore di PrezzoMagazzino deve essere maggiore o uguale a 0*
valoreSesso	*Il valore di sesso deve essere 'M', 'F' o 'U'*

Progettazione logica

Schema logico

Scelto il modello relazionale come nostro modello dei dati e a partire dal class diagram ristrutturato andiamo ora a costruire lo schema logico che farà da base per la fase finale di progettazione fisica del database. Gli attributi sottolineati una volta sono primary keys. Gli attributi sottolineati due volte sono foreign keys.

Traduzione classi -> schema relazionali

Articolo (SKU, Taglia, Colore, Quantità, PathImmagine, CodiceABarre)

dove CodiceABarre si riferisce a DescrittoreArticolo.CodiceABarre

---

DescrittoreArticolo (CodiceABarre, Marca, Categoria, PrezzoDiListino,PrezzoMagazzino, Sesso, Nome, Descrizione)

Fornitore (PartitaIva, Nome, Via, NumeroCivico, CAP, NumeroDiTelefono)

Transazione (CodiceTransazione, Data, ValoreTotale, PartitaIva)

dove PartitaIva si riferisce a Fornitore.PartitaIva. Introduciamo inoltre una pk surrogata CodiceTransazione

---

ComposizioneTransazione (CodiceTransazione, SKU, Quantità, Saldo, Valore)

dove CodiceTrasazione si riferisce a Transazione.CodiceTransazione e SKU si riferisce a Articolo.SKU

Associazioni

L'associazione uno a molti Articolo-DescrittoreArticolo è espressa dalla presenza della foreign key CodiceABarre nello schema relazionale Articolo.

L'associazione uno a molti Transazione-Fornitore è espressa dalla presenza della foreign key PartitaIva nello schema relazionale Transazione.

L'associazione molti a molti Articolo-Transazione è espressa dalla presenza dello schema ponte ComposizioneTransazione.

Progettazione fisica

Note sull'implementazione

L'implementazione fisica sarà effetuata tramite la piattaforma di sviluppo PostgreSql 12.

Ciò comporterà probabilmente una diversa implementazione di alcune parti dell'elaborato rispetto all' SQL riportato di seguito (i trigger vanno costruiti in modo diverso, ad esempio).

Definizione delle tabelle

```
CREATE TABLE Articolo (

    SKU INTEGER NOT NULL,
    Taglia ENUM ('XS', 'S', 'M', 'L', 'XL', 'XXL')NOT NULL,
    Colore ENUM ('Nero', 'Bianco', 'Rosso', 'Blu', 'Verde', 'Grigio')
    NOT NULL,
    Quantità INTEGER NOT NULL,
    PathImmagine VARCHAR(200) DEFAULT 'res\\images\\noimage.png' NOT NULL
,
    CodiceABarre CHAR(11) NOT NULL,
    CONSTRAINT valoreQuantitàArticolo CHECK (Quantità >= 0)
);


CREATE TABLE ComposizioneTransazione (
    SKU INTEGER NOT NULL,
    CodiceTransazione INTEGER NOT NULL,
    Quantità INTEGER NOT NULL,
    Saldo INTEGER DEFAULT 0 NOT NULL,
    Valore FLOAT DEFAULT 0,
    CONSTRAINT valoreSaldo CHECK (Saldo BETWEEN 0 AND 100),
    CONSTRAINT valoreValore CHECK (Valore >= 0),
    CONSTRAINT valoreQuantitàCTEstremoInferiore CHECK (Quantità > 0)
);


CREATE TABLE DescrittoreArticolo (
    CodiceABarre CHAR(11) NOT NULL,
    Marca ENUM ('Adidas', 'Nike', 'H&M', 'PiazzaItalia', 'Alcott',
'Bershka', 'Versace', 'Gucci', 'Napapijiri') NOT NULL,
    Categoria ENUM ('Magliette', 'Maglioni', 'Felpe', 'Giacche',
'Cappotti', 'Pantaloni', 'Jeans', 'Camicie', 'Intimo', 'Accessori',
'Pantaloncini', 'Gonne') NOT NULL,
    Sesso CHAR(1) NOT NULL,
    PrezzoDiListino FLOAT NOT NULL,
    PrezzoMagazzino FLOAT NOT NULL,
    Nome VARCHAR(25) NOT NULL,
    Descrizione VARCHAR(100) NOT NULL,
    CONSTRAINT valorePrezzoDiListino CHECK (PrezzoDiListino >= 0),
    CONSTRAINT valorePrezzoMagazzino CHECK (PrezzoMagazzino >= 0),
    CONSTRAINT valoreSesso CHECK ((Sesso = 'M') OR (Sesso = 'F') OR
(Sesso = 'U')) );
```

Progettazione fisica

Continuo definizione delle tabelle

```
CREATE TABLE Fornitore (
    PartitaIva CHAR(11) NOT NULL,
    Nome VARCHAR(50) NOT NULL,
    Via VARCHAR(30) NOT NULL,
    NumeroCivico VARCHAR(10) NOT NULL,
    CAP VARCHAR(5) NOT NULL,
    NumeroDiTelefono CHAR(10),
    CONSTRAINT uNumeroDiTelefono UNIQUE(NumeroDiTelefono)
);


CREATE TABLE Transazione (

    CodiceTransazione INTEGER NOT NULL,
    Data DATE NOT NULL,
    ValoreTotale FLOAT DEFAULT 0,
    PartitaIva CHAR(11),
    CONSTRAINT valoreValoreTotale CHECK (ValoreTotale >= 0)
);
```

Definizione delle primary keys

```
ALTER TABLE Articolo
    ADD CONSTRAINT pkArticolo PRIMARY KEY (SKU);


ALTER TABLE DescrittoreArticolo
    ADD CONSTRAINT pkDescrittoreArticolo PRIMARY KEY (CodiceABarre);


ALTER TABLE Fornitore
    ADD CONSTRAINT pkFornitore PRIMARY KEY (PartitaIva);


ALTER TABLE Transazione
    ADD CONSTRAINT pkTransazione PRIMARY KEY (CodiceTransazione);
```

Definizione delle foreign keys

```
ALTER TABLE ComposizioneTransazione
    ADD CONSTRAINT fk1ComposizioneTransazione FOREIGN KEY (SKU) REFER-
ENCES Articolo(SKU) ON DELETE NO ACTION;


ALTER TABLE ComposizioneTransazione
    ADD CONSTRAINT fk2ComposizioneTransazione FOREIGN KEY
(CodiceTransazione) REFERENCES Transazione(CodiceTransazione) ON DELETE
CASCADE;
```

Progettazione fisica

Continuo definizione delle foreign keys

```
ALTER TABLE Articolo
    ADD CONSTRAINT fkArticolo FOREIGN KEY (CodiceABarre) REFERENCES
DescrittoreArticolo(CodiceABarre) ON DELETE NO ACTION;


ALTER TABLE Transazione
    ADD CONSTRAINT fkTransazione FOREIGN KEY (PartitaIva) REFERENCES
Fornitore(PartitaIva) ON DELETE NO ACTION;
```

Definizione dei trigger "ON UPDATE CASCADE"

```
CREATE TRIGGER updateCascadeSKU

AFTER UPDDATE OF SKU ON Articolo

FOR EACH ROW

BEGIN

UPDATE ComposizioneTransazione

SET SKU = NEW.SKU

WHERE SKU = OLD.SKU ;

END;


CREATE TRIGGER updateCascadeCodiceABarre

AFTER UPDDATE OF CodiceABarre ON DescrittoreArticolo

FOR EACH ROW

BEGIN

UPDATE Articolo AS A

SET A.CodiceABarre = NEW.CodiceABarre

WHERE A.CodiceABarre = OLD.CodiceABarre ;

END;
```

Progettazione fisica

Continuo definizione dei trigger "ON UPDATE CASCADE"

```
CREATE TRIGGER updateCascadeCodiceTransazione
AFTER UPDDATE OF CodiceTransazione ON Transazione
FOR EACH ROW
BEGIN
UPDATE ComposizioneTransazione
SET CodiceTransazione = NEW.CodiceTransazione
WHERE CodiceTransazione = CodiceTransazione ;
END;

CREATE TRIGGER updateCascadePartitaIva
AFTER UPDDATE OF PartitaIva ON Fornitore
FOR EACH ROW
BEGIN
UPDATE Transazione
SET PartitaIva = NEW.PartitaIva
WHERE PartitaIva = PartitaIva ;
END;
```

Progettazione fisica

Definizione sequenze e trigger associati

E' ideale definire sequenze per generare automaticamente i valori delle primary key CodiceTransazione e SKU in seguito a un inserimento.

Definizione sequenza SequenceCodiceTransazione

```
CREATE SEQUENCE SequenceCodiceTransazione
STARTWITH 0
MIN VAL 0
MAX VAL 1000000
INCREMENT BY 1
NO CACHE
NO CYCLE ;
```

Definizione trigger nCodiceTransazione

```
CREATE TRIGGER nCodiceTransazione
BEFORE INSERT ON Transazione
FOR EACH ROW
BEGIN
NEW.CodiceTransazione = SequenceCodiceTransazione.NEXTVAL ;
END;
```

Progettazione fisica

Continuo definizione sequenze e trigger associati

Facciamo lo stesso per SKU.

Definizione sequenza SequenceSKU

```
CREATE SEQUENCE SequenceSKU
STARTWITH 0
MIN VAL 0
MAX VAL 10000000
INCREMENT BY 1
NO CACHE
NO CYCLE ;
```

Definizione trigger nSKU

```
CREATE TRIGGER nSKU
BEFORE INSERT ON Articolo
FOR EACH ROW
BEGIN
NEW.SKU = SequenceSKU.NEXTVAL ;
END;
```

Progettazione fisica

Trigger

Definiamo di seguito una serie di trigger necessari al mantenimento della coerenza del database e/o utili alla nostra applicazione java.

Definizione del trigger aggiornaCodiceTransazione

Per comprendere la natura di questo trigger, bisogna fare riferimento al contesto per il quale lo stiamo realizzando. Nel contesto dellla nostra applicazione java,tutti gli inserimenti di composizionetransazione saranno sempre immediatamente successivi all' inserimento della transazione di cui fanno parte.

```
i.   CREATE TRIGGER aggiornaCodiceTransazione
ii.  BEFORE INSERT ON ComposizioneTransazione
iii. FOR EACH ROW
iv.  BEGIN
v.   NEW.ComposizioneTransazione =
vi.  SequenceCodiceTransazione.CURRENTVAL
vii. END;
```

Progettazione fisica

Definizione del vincolo (tramite trigger) valoreQuantitàCTEstremoSuperiore

Se la transazione di cui fa parte è una vendita e non un rifornimento, Quantità all'interno di ComposizioneTransazione non può assumere un valore maggiore di Quantità all'interno di Articolo al momento dell'inserimento per ovvie ragioni.

```
 i.   CREATE TRIGGER valoreQuantitàCTEstremoSuperiore
 ii.  BEFORE INSERT ON ComposizioneTransazione
 iii. FOR EACH ROW
 iv.  BEGIN
 v.   DECLARE
 vi.  riga1 Articolo%ROWTYPE ;
 vii. riga2 Transazione%ROWTYPE ;
 viii. QuantitàMaggioreDiDisponibilitàException EXCEPTION;
 ix.  BEGIN
 x.   SELECT * INTO riga1
 xi.  FROM Articolo AS A
 xii. WHERE A.SKU = NEW.SKU ;
 xiii. SELECT * INTO riga2
 xiv. FROM Transazione AS T
 xv.  WHERE T.CodiceTransazione = SequenceCodiceTransazione.CURRENTVAL ;
 xvi. IF (riga2.PartitaIva IS NULL) THEN
 xvii. IF (NEW.Quantità > riga1.Quantità) THEN
 xviii. RAISE (QuantitàMaggioreDiDisponibilitàException);
 xix. END IF;
 xx.  END IF;
 xxi. EXCEPTION
 xxii. WHEN QuantitàMaggioreDiDisponibilitàException
 xxiii. RAISE_APPLICATION_ERROR(-20001, 'Quantità inserita maggiore di
 xxiv. disponibilità');
 xxv. END;
 xxvi. END;
```

Progettazione fisica

Definizione del trigger aggiornaQuantitàArticolo

Inserendo una nuova istanza di ComposizioneTransazione, la quantità dell'articolo va diminuita in modo adeguato se la transazione è una vendita, aumentata se un rifornimento. Questo trigger si occupa di questo.

```
 i.   CREATE TRIGGER aggiornaQuantitàArticolo
 ii.  AFTER INSERT ON ComposizioneTransazione
 iii. FOR EACH ROW
 iv.  BEGIN
 v.   DECLARE
 vi.  riga Transazione%ROWTYPE ;
 vii. BEGIN
 viii. SELECT * INTO riga
 ix.  FROM Transazione AS T
 x.   WHERE T.CodiceTransazione = NEW.CodiceTransazione ;
 xi.  IF (riga.PartitaIva IS NULL) THEN
 xii. UPDATE Articolo
 xiii. SET Quantità = Quantità – NEW.Quantità
 xiv. WHERE SKU = NEW.SKU ;
 xv.  ELSE
 xvi. UPDATE Articolo
 xvii. SET Quantità = Quantità + NEW.Quantità
 xviii. WHERE SKU = NEW.SKU ;
 xix. END IF;
 xx.  END;
 xxi. END;
```

Progettazione fisica

Definizione del trigger calcolaValore

Inserendo una nuova istanza di ComposizioneTransazione, il campo dell'attributo valore dovrà essere calcolato(questo campo non viene infatti inserito dall'utente che utilizza la nostra applicazione java) a partire dai valori del campo saldo e quantità.

Nota: articolicondescrittori è una vista definita in seguito come natural join di articolo e descrittorearticolo.

```
 i.   CREATE TRIGGER calcolaValore
 ii.  AFTER INSERT ON ComposizioneTransazione
 iii. FOR EACH ROW
 iv.  BEGIN
 v.   DECLARE
 vi.  riga articolicondescrittori%ROWTYPE ;
 vii. rigaTransazione Transazione%ROWTYPE ;
 viii. BEGIN
 ix.  SELECT * INTO riga
 x.   FROM articolicondescrittori AS A
 xi.  WHERE A.SKU = NEW.SKU ;
 xii. SELECT * INTO rigaTransazione
 xiii. FROM Transazione AS T
 xiv. WHERE T.CodiceTransazione = NEW.CodiceTransazione ;
 xv.  IF(rigaTransazione.PartitaIva IS NULL) THEN
 xvi. NEW.Valore = NEW.Quantità *(riga.PrezzoDiListino –
 xvii. (riga.PrezzoDiListino * (NEW.Saldo / 100)))
 xviii. ELSE
 xix. UPDATE ComposizioneTransazione
 xx.  NEW.Valore = NEW.Quantità * (riga.PrezzoMagazzino –
 xxi. (riga.PrezzoMagazzino *(NEW.Saldo / 100)))
 xxii. END IF;
 xxiii. END;
 xxiv. END;
```

Progettazione fisica

Definizione del trigger calcolaValoreTotale

Una volta inserite tutte le istanze di composizioneTransazione associate a una transazione (o man mano che queste vengono inserite), il campo valoreTotale della transazione corrispondente va calcolato/aggiornato.

```
 i.   CREATE TRIGGER calcolaValoreTotale
 ii.  AFTER UPDATE OF Valore ON ComposizioneTransazione
 iii. FOR EACH ROW
 iv.  BEGIN
 v.   UPDATE Transazione
 vi.  SET ValoreTotale = ValoreTotale + NEW.Valore
 vii. WHERE CodiceTransazione = NEW.CodiceTransazione ;
 viii. END;
```

Progettazione fisica

Viste

Definiamo di seguito alcune viste di servizio per il db e/o utili alla nostra applicazione java.

Definizione della vista ArticoliConDescrittori

La seguente vista fornisce una tabella con le colonne di Articolo e DescrittoreArticolo.

```
 i.   CREATE VIEW ArticoliConDescrittori
 ii.  SELECT *
 iii. FROM Articoli AS A NATURAL JOIN DescrittoreArticolo
 iv.  ORDER BY A.SKU ;
```

Definizione della vista TransazioniConComposizioni

La seguente vista fornisce una tabella con le colonne di Transazione e ComposizioneTransazione.

```
 i.   CREATE VIEW TransazioniConComposizioni
 ii.  SELECT *
 iii. FROM Transazione AS T NATURAL JOIN ComposizioneTransazione
 iv.  ORDER BY T.CodiceTransazione ;
```

Progettazione fisica

Definizione della vista Vendite

La seguente vista fornisce una tabella contenente tutte le vendite effettuate con una serie di campi pescati da varie tabelle.

```
 i.   CREATE VIEW Vendite (ID, Contenuto, Data, Costo)
 ii.  SELECT X.CodiceTransazione, LISTAGG((X.Quantità || 'x ' ||Y.Nome),
 iii. ',') WITHIN GROUP (ORDER BY Y.Nome), X.Data, X.ValoreTotale
 iv.  FROM (Transazione NATURAL JOIN ComposizioneTransazione) AS X JOIN
 v.   (DescrittoreArticolo NATURAL JOIN Articolo) AS Y ON
 vi.  X.SKU = Y.SKU
 vii. WHERE X.PartitaIva IS NULL
 viii. GROUP BY X.CodiceTransazione, X.Data, X.ValoreTotale
 ix.  ORDER BY X.CodiceTransazione ;
```

Progettazione fisica

Definizione della vista Rifornimenti

La seguente vista fornisce una tabella contenente tutti i rifornimenti effettuati con una serie di campi pescati da varie tabelle.

```
 i.   CREATE VIEW Rifornimenti(ID, Contenuto, Fornitore, Data, Costo)
 x.   SELECT Z.CodiceTransazione, LISTAGG((Z.Quantità || 'x ' || Z.Nome,
xi.   ',') WITHIN GROUP (ORDER BY Z.Nome),F.Nome, Z.Data, Z.ValoreTotale
 ii.   FROM ((Transazione NATURAL JOIN ComposizioneTransazione) AS Y JOIN
iii.   (DescrittoreArticolo NATURAL JOIN Articolo) AS X ON
 iv.   X.SKU = Y.SKU) AS Z JOIN Fornitore AS F ON Z.PartitaIva =
  v.   F.PartitaIva
 vi.   GROUP BY Z.CodiceTransazione, Z.Data, Z.ValoreTotale,
vii.   F.Nome
viii.  ORDER BY Z.CodiceTransazione ;
```

Progettazione fisica

Definizione della vista Fornitori

La seguente vista fornisce una tabella contenente tutti i fornitori del negozio con campi pescati da varie tabelle.

```
 i.   CREATE VIEW Fornitori (Fornitore, PartitaIva, Transazioni,
ii.   ValoreTotale)
iii.  SELECT F.Nome, F.PartitaIva, LISTAGG(T.CodiceTransazione, ‘, ’)
iv.   WITHIN GROUP (ORDER BY T.CodiceTransazione), SUM(T.ValoreTotale)
 v.   FROM Fornitore AS F NATURAL JOIN Transazione AS T
vi.   GROUP BY F.Nome, F.PartitaIva
vii.  ORDER BY F.Nome ;
```