Università degli studi di Trieste

Intelligenza Artificiale e Data Analytics

Progetto di Basi di Dati

Database di un negozio di carte collezionabili

Tonet Lorenzo

22 maggio 2024

Indice:

Presentazione progetto

Si vuole realizzare una base di dati a supporto di venditori, giocatori e collezionisti di carte da gioco collezionabili. L'obiettivo di tale database sarà quello sia di permettere a venditori di raggiungere una platea di clienti più ampia e affine ai propri target ma anche aiutare giocatori e collezionisti nella ricerca di determinate carte senza aver bisogno di comprare bustine in cui non è noto il contenuto.
L'utente sarà dunque in grado di sia mettere in vendita che a sua volta comprare copie singole. Non essendo le carte (la maggior parte) stampate in singola copia, sarà possibile scegliere tra diverse offerte da parte di utenti che vendono le loro copie della stessa carta. Ciò che identifica una carta sarà un codice univoco formato da 2 lettere che indicano l'espansione con la quale è uscita e un numero di 3 cifre che indica la posizione nella lista di tale espansione, la quale avrà un anno di pubblicazione e un nome. Alcune copie avranno un grado ovvero uno o più "voti" sullo stato fisico e di conservazione che viene assegnato da dei giudici terzi specializzati in questo lavoro di valutazione in modo da fornire al compratore informazioni sullo stato di tale copia. Sapendo che le carte contengono un'illustrazione si vuole tenere traccia del disegnatore il quale, fattore che può dare valore ad una carta. Gli utenti saranno in grado di ordinare e farsi spedire delle carte anche con la possibilità di scegliere tra varie società di trasporto merci le quali potranno differire per prezzo, tempo medio di spedizione e in alcuni casi "condizioni di consegna".

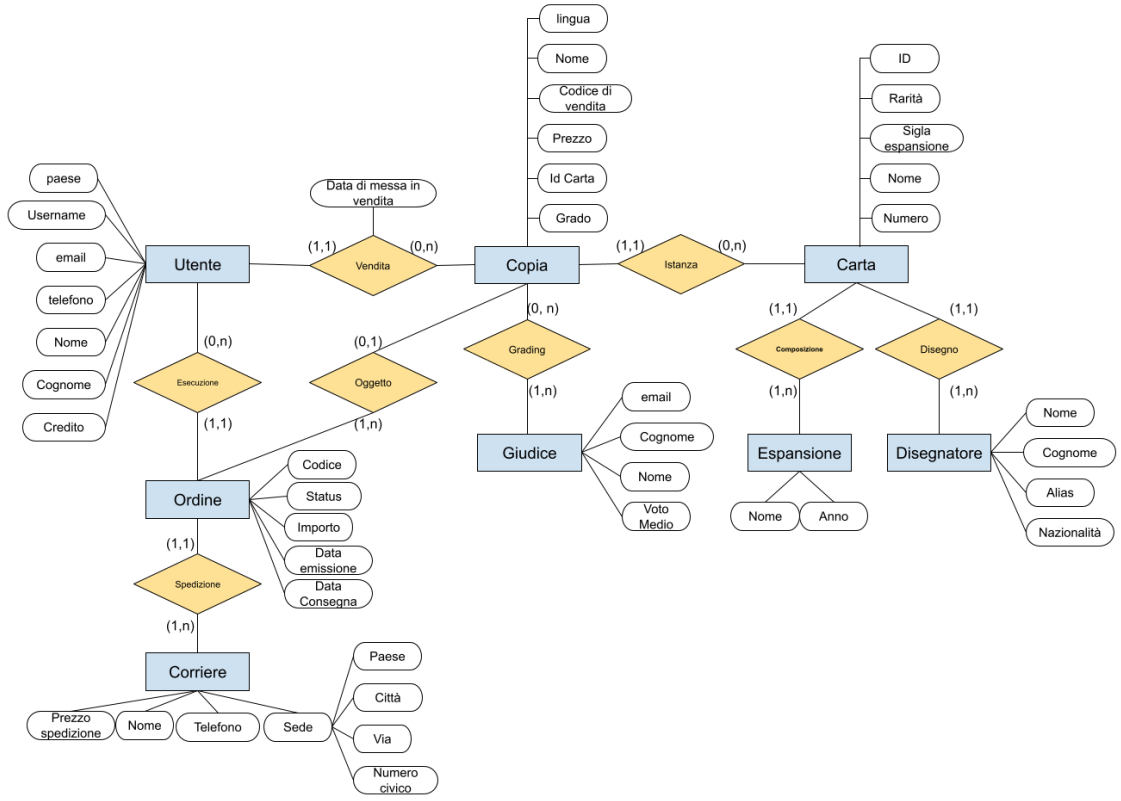
Requisiti della base di dati

Si vuole tenere aggiornata la base di dati per tracciare gli ordini degli utenti, i prodotti che vengono messi in vendita, da quali utenti e riguardo al grading, chi ha messo quale voto in modo da poter offrire al cliente abbastanza informazioni per aiutarlo nella scelta dei propri acquisti e permettere anche una concorrenza tra diversi giudici. Delle copie in vendita si vuole tenere le informazioni su prezzo, data di messa in vendita, utente, grado (facoltativo), lingua e di quale carta sono la copia in modo poi da avere informazioni su indici come il prezzo medio al quale le copie di una determinata carta vengono vendute. Per semplicità e scarsa probabilità si costruisce si ipotizza che non ci siano artisti con lo stesso nome d'arte o alias così come le società di trasporto. Inoltre si vuole che gli utenti che hanno ricevuto i prodotti ordinati ricevano una notifica via telefono una sola volta e che la verifica su quali ordini sono stati consegnati venga fatta circa due volte al giorno.

Glossario dei termini

Termine	Sinonimi	Definizione	Collegamento
Utente	collezionista, giocatore	La persona che usufruirà del servizio per ordinare carte singole e bustine e vendere carte singole	Ordini, Copie
Copia	Comunemente chiamata “carta”	La stampa fisica dell’entità “Carta”.	Carta, Utente
Carta		La carta indipendente dalla lingua di stampa e appartenente ad un’ espansione	Copia,Espansione
Disegnatore	Illustratore, Artista	L’artista responsabile dell’illustrazione presente sulla carta	Carta
Espansione		Insieme di carte uscite nello stesso anno	Carta
Società di trasporto	Azienda di trasporto	Società specializzata nel trasporto e consegna merci che si prenderà a carico un’ordine	Ordine
Giudice		Persona esterna al circolo di vendite con il compito di ricevere copie dagli utenti e assegnare un voto in una scala da uno a 10 sullo stato di conservazione della carta	Copie
Ordine	Acquisto	Informazioni relative all’acquisto fatto da un utente	Copie

Diagramma Entity-Relationship



Dizionario dei dati (entità)

Entità	Attributi	Identificatore
Utente	Username, Nome, Cognome, Email, Telefono, Paese, Credito	Username
Copia	Grado medio, Codice di vendita, Data inizio vendita, Prezzo, Stato, Lingua, Stato	Codice di vendita
Carta	Rarità, Numero di collezione, Nome Sigla espansione, Id, Illustratore	Id
Espansione	Nome, Anno, Sigla	Nome, Anno
Illustratore	Nome, Cognome, Alias, Paese	Alias
Società di trasporto	Nome, Telefono, Sede (Paese, CAP, Via, Numero) Prezzo spedizione	Nome
Giudice	Nome, Cognome, email, Matricola, Paese	Matricola
Ordine	Codice id, Importo totale, Utente, Stato, Data emissione, Data consegna	Codice

Dizionario dei dati (relazioni)

Relazione	Descrizione	Componenti	Attributi
Vendita	L'utente mette in vendita una copia	Utente, Copia	Data di messa in vendita
Istanza	La copia in vendita è istanza di una carta indipendente	Copia, Carta	
Composizione	Un'espansione è un insieme di carte	Carta, Espansione	
Disegno	L'illustrazione di una carta è fatta da un artista	Carta, Artista	
Grading	Una copia può essere valutata da uno o più giudici	Copia, Giudice	
Oggetto	Una copia è oggetto di un ordine	Copia, Ordine	
Esecuzione	L'ordine è eseguito da un utente	Ordine, Utente	
Spedizione	La spedizione dei prodotti ordinati è a carico di una società di trasporto	Ordine, Azienda	

Vincoli non esprimibili graficamente

- Il grado della carta va da 1 a 10
- Una copia il cui stato è "venduta" non può essere oggetto di nuovi ordini
- Nel caso una carta sia stata valutata da più giudici si terrà conto della media dei voti che ha ricevuto
- La data di consegna di un ordine non può essere precedente alla data dell'emissione dello stesso e nessuna di queste due può essere precedente alla data di messa in vendita

Tabella dei volumi

Suppongo che sulla piattaforma ci siano circa 5000 utenti attivi, che solo la metà di essi abbia messo in vendita una copia e che quelli che lo hanno fatto stiano cercando di venderne 3 in media. Suppongo che ci siano circa una ventina di espansioni totali (più o meno una all'anno) e che un'espansione sia formata da 150 carte. Considerando che spedire una copia ad un giudice e richiedere il suo servizio non è gratuito ed è a carico del proprietario della copia, si stima che la metà delle carte presenti non siano state valutate da un giudice ma che gli utenti decidano di far valutare solamente le copie di una certa rarità. I giudici registrati nel database saranno circa una quindicina. Gli artisti saranno almeno uno per espansione.

Concetto	Tipo	Volume
Utente	E	≈ 5 000
Copia	E	≈ (5 000/2)*3 = 7500
Carta	E	≈ 150 * 20 = 3000
Espansione	E	≈ 20
Giudice	E	≈ 15
Artista	E	≈ 30
Ordine	E	≈ 5 000/2 = 2500
Istanza	R	≈ (3000 -100) * 15 = 4350
Composizione	R	≈ 150
Oggetto	R	≈ 3
Esecuzione	R	≈ 200
Grading	R	≈ 4000
Vendita	R	≈ 7500

Operazioni di interesse

Operazione	Tipo	Frequenza
Ricerca di una carta	Interattiva	500 / giorno
Ricerca “con filtri”	Interattiva	200 / giorno
Acquisto di una carta	Interattiva	50 / settimana
Messa in vendita	Interattiva	50 / settimana
Verifica stato ordini e notifica gli utenti che hanno ricevuto	Batch	2 / giorno
Aggiornamento grado copie	Batch	Aggiornamento tabella “Gradazioni”
Reperimento informazioni sull’ordine da parte dell’utente	Interattiva	2 / giorno x ordine non ancora completato
Visualizza tutte le carte di un’espansione	Interattiva	1 /settimana

Analisi delle ridondanze

Ridondanze

L’entità Ordine contiene un attributo “importo” che è semplicemente calcolabile sommando tutti i prezzi degli articoli nello stesso ordine e il prezzo della spedizione.
Per l’operazione “Reperimento informazioni sull’ordine da parte dell’utente” senza la ridondanza avrei all’incirca 2 accessi in lettura al giorno per ogni ordine presente nel database non ancora completato e per ognuno di essi dovrei fare un numero di accessi (lettura) alla tabella “copie” pari al numero di quelle che sono state ordinate, inoltre dovrei fare un accesso (lettura) anche per reperire l’informazione sul prezzo della spedizione. Mantenendo la ridondanza questi accessi verrebbero fatti una sola volta al momento dell’ordine. Risulta conveniente mantenere la ridondanza.
L’attributo “Id” delle carte è facilmente calcolabile combinando “Numero” e “Sigla espansione”. Anche questo verrebbe fatto solo la prima volta che la carta viene aggiunta al database.
Il grado di una carta è facilmente calcolabile in quanto si riferisce al valore medio di tutti i voti che ha ricevuto quella determinata copia e viene ricalcolato una volta ogni che una carta viene valutata e vista la scarsa frequenza di aggiornamento della tabella

gradazione e l'utilità di questa informazione per il cliente si decide di mantenerlo nel database.

Cicli

Si può notare che l'unico ciclo presente nello schema entità-relazione è quello che fa "Utente" → "Copia" → "Ordine" → "Utente". L'esistenza di questo ciclo però, è il risultato del fatto che, come è normale ipotizzare, l'utente che venderà la copia sarà diverso dall'utente che eseguirà l'ordine, quindi le relazioni del ciclo vengono usate in contesti diversi.

Generalizzazioni

Non sono attualmente presenti generalizzazione quindi non è necessaria nessuna operazione di accorpamento/ sostituzione.

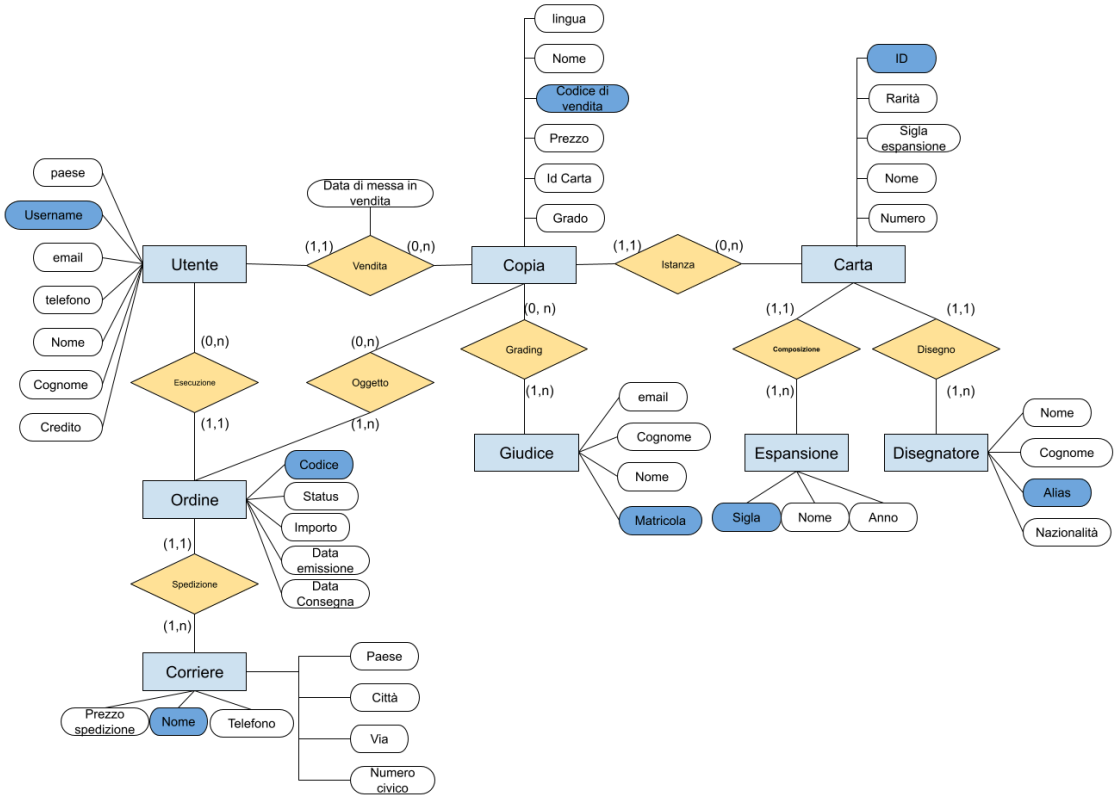
Partizionamento di entità

Si osserva che nessuna entità presenta degli attributi che si riferiscono a concetti diversi o che necessitano un accesso separato, perciò non risulta necessario alcun tipo di partizionamento di entità.

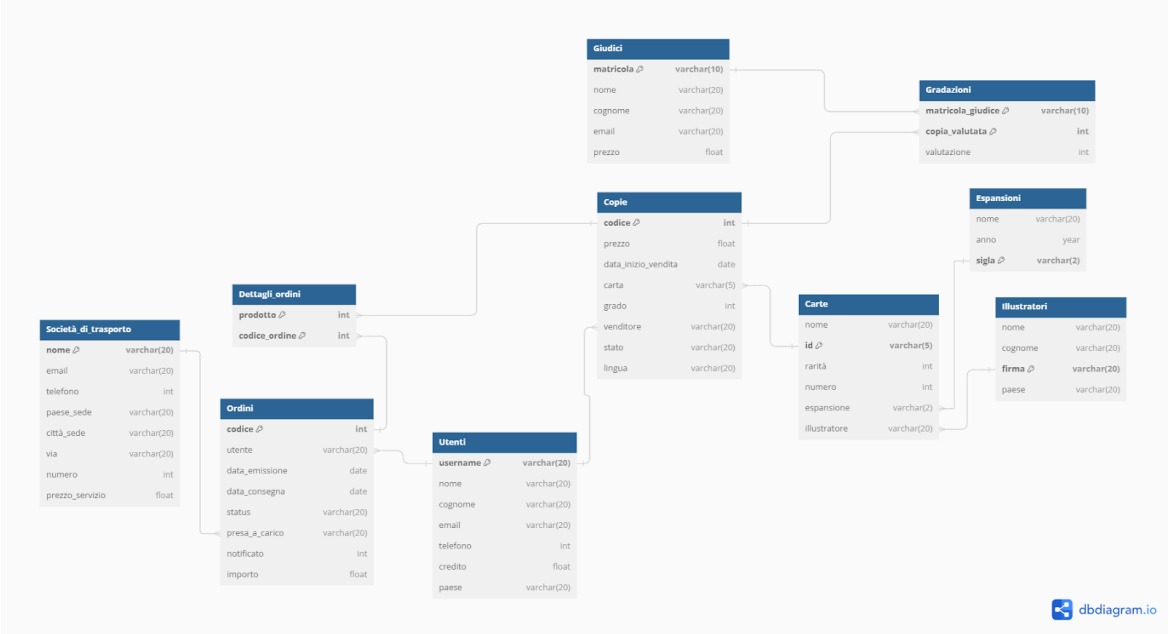
Scelta degli identificatori

- Utente: come identificatore univoco si sceglie di utilizzare lo username imponendo, alla registrazione di un nuovo utente, di non utilizzarne uno già esistente.
- Copia: per identificare le copie in vendita/vendute dal servizio si utilizza un codice che viene generato al momento della messa in vendita.
- Carta: le carte vengono identificate in base a un identificativo formato da 2 lettere che daranno l'informazione sull'espansione e un numero di 3 cifre unico per ogni carta di un'espansione
- Espansione : l'espansione sarà identificata dalla sigla composta da due lettere la quale può comporre 676 possibili valori.
- Ordine: anche all'ordine si assegna un codice nel momento in cui viene emesso e lo si usa come identificativo
- Artista : si sceglie l'alias come identificativo in quanto il volume è abbastanza ridotto e si ipotizza che non ce ne siano di uguali.
- Società di trasporto: si sceglie il nome.
- Giudice: si sceglie per semplicità di scegliere la matricola.

Diagramma Entity-Relationship Ristrutturato



Schema logico



Normalizzazione:

Si può notare come tutte le colonne siano atomiche quindi non contengono valori multipli e per la presenza di primary key viene garantita anche l'unicità delle unità, di conseguenza la prima forma normale è garantita per tutte le tabelle. Noto che nella tabella "Copie" il parametro "nome" non dipende dalla primary-key ma dall'attributo "carta" ovvero dipende dal valore "nome" nella tabella "Carte" quindi viene rimosso per garantire la seconda forma normale. Ogni altra colonna della base di dati dipende strettamente dalla primary-key, le tabelle che potevano venire scomposte lo sono state (ex. Valutazioni e Giudici) quindi viene soddisfatta anche la seconda forma normale. Gli attributi inoltre sono dipendenti interamente dalla primary key quindi anche le condizioni della terza forma normale sono soddisfatte per tutte le tabelle tranne che per "Ordini" e "Carte" che contengono campi calcolati, ma entrambi i campi vengono calcolati solo la prima volta che vengono inseriti e ciò risparmia un grande numero di accessi al database, quindi si decide di mantenerli.

Particolarità nella creazione del database

L'unica particolarità presente nella creazione del database è stata messa in modo da prevenire la perdita di informazioni sugli ordini già eseguiti nel caso di cancellazione di un utente. Si ritiene corretto inserire nel campo "utente" dell'ordine in questione il valore nullo:

```
ALTER TABLE `Ordini` ADD FOREIGN KEY (`utente`) REFERENCES `Utenti` (`username`) ON DELETE SET NULL;
```

Operazioni

1. Ricerca in base al nome di una carta: funzione che prende in input il nome della carta ricercata e mostra le offerte ancora disponibili

```
-- RICERCA IN BASE AL NOME
DELIMITER §§
CREATE PROCEDURE ricerca_carta_per_nome(IN
nome_carta varchar(20))
BEGIN
    SELECT * FROM copie INNER JOIN carte on copie.carta = carte.id WHERE copie.nome = nome_carta AND copie.stato = 'Disponibile';
END §§
DELIMITER ;
```

2. Ricerca "filtrata" di una carta: Considerando che le possibili combinazioni di filtri sarebbero moltissime vengono mostrati alcuni esempi.

```
-- Ricerca di tutte le copie in vendita di una particolare con rarità decrescente
DELIMITER §§
CREATE PROCEDURE ricerca_carta(IN codice_carta varchar(5))
BEGIN
    select carte.nome, copie.prezzo, copie.lingua, carte.rarità as rarità from copie
    left join carte on copie.carta = carte.id where copie.stato != 'Venduta' order by rarità desc ;
END §§
DELIMITER ;

-- Ricercare tutte le copie in vendita di carte illustrate da artisti di un determinato paese
DELIMITER §§
CREATE PROCEDURE ricerca_carta_per_artista(IN paese varchar(20))
BEGIN
    SELECT * FROM copie
    LEFT JOIN carte on copie.carta = carte.id
    LEFT JOIN illustratori on carte.illustratore = illustratori.firma WHERE illustratori.paese = paese;

END §§
DELIMITER ;

-- Mostra un'intera espansione
DELIMITER §§
CREATE PROCEDURE mostra_espansione(IN
nome_espansione varchar(20))
BEGIN
    SELECT * FROM carte INNER JOIN espansione on carte.espansione = espansioni.sigla WHERE espansioni.nome = nome_espansione;
END §§
DELIMITER ;
```

3. Aggiornamento sul grado di una carta: Per questa operazione si decide di utilizzare un trigger che ricalcola il valore di grado nella tabella “Copie” ogni volta che viene inserita una gradazione nella tabella “Gradazioni”.

```
-- Aggiornare il grado di una carta
DELIMITER §§
CREATE TRIGGER aggiorna_grado
AFTER INSERT ON gradazioni
FOR EACH ROW
BEGIN
    DECLARE voto_totale INT;
    DECLARE voto_count INT;
    DECLARE voto_medio float;

    SELECT SUM(valutazione), COUNT(*) INTO voto_totale, voto_count FROM gradazioni
    WHERE copia_valutata = NEW.copia_valutata;

    -- Aggiorna il voto medio nella tabella Carte
    UPDATE Copie
    SET grado = voto_totale / voto_count
    WHERE codice = NEW.copia_valutata;
END §§
DELIMITER ;
```

4. Prezzo medio di una carta: Un utente potrebbe voler reperire l’informazione sul prezzo medio a cui una determinata carta viene venduta solitamente. La funzione prenderà in input il codice identificativo della carta e restituirà il valore desiderato.

```
-- Calcolare il prezzo medio a cui è stata venduta una determinata carta
DELIMITER §§
CREATE PROCEDURE prezzo_medio_carta(IN codice_carta varchar(5), OUT prezzo_medio float)
BEGIN
    DECLARE temp float;
    SET temp = (SELECT AVG(prezzo) FROM copie WHERE carta = codice_carta and stato = 'Venduta');
    SET prezzo_medio = temp;

END §§
DELIMITER ;
```

5. Mettere in vendita la copia di una carta: Funzione per mettere in vendita una carta che prenderà in input tutti i parametri della carta venduta e assegnerà un codice.

```
-- Mettere una copia in vendita
DELIMITER §§
CREATE PROCEDURE crea_vendita(IN
username varchar(20), IN nome varchar(20), IN prezzo float, IN carta varchar(5), IN lingua varchar(20))
BEGIN

INSERT INTO Copie (prezzo, data_inizio_vendita, carta, venditore, lingua) VALUES
(prezzo, curdate(), carta, username, lingua);

END §§
DELIMITER ;
```

6. Eseguire un ordine: Funzione per ordinare una o più carte. La funzione prende in input una stringa formattata nel seguente modo: “codice_prodotto_1;codice_prodotto_2;codice_prodotto_3;...”. Tale stringa verrà composta e fornita a livello di applicazione quindi si ipotizza sia formattata in nel modo richiesto.
   Prende in input anche la società di trasporto scelta dall’utente per la spedizione (per calcolare l’importo totale) e l’username dell’utente che sta effettuando l’ordine. Il tutto verrà fatto in una transazione perché si vuole garantire l’atomicità della procedura e evitare la concorrenza.

```
DELIMITER §§
CREATE PROCEDURE EseguiOrdini(IN codici VARCHAR(1000), IN username varchar(20), IN trasporto varchar(20), OUT importo_totale float)
BEGIN
    DECLARE codice_prod VARCHAR(20);
    DECLARE pos INT DEFAULT 1;
    DECLARE len INT;
    DECLARE my_codice_ordine INT;
    DECLARE totale_ordine float;

    START TRANSACTION;
    INSERT INTO ordini (utente, data_emissione, data_consegna, status, presa_a_carico) VALUES
    (username, curdate(), null, 'emesso', trasporto);
    SET my_codice_ordine = (SELECT codice FROM ordini ORDER BY codice DESC LIMIT 1);

   WHILE (pos <= LENGTH(codici)) DO
      SET len = LOCATE(';', codici, pos);
      -- LOCATE() ritorna la posiz. della prima occorrenza di una stringa in una stringa (le posizioni partono da 1)
      -- se la sottostringa non è presente ritorna 0
      IF len = 0 THEN
          SET len = LENGTH(codici) + 1;
      END IF;
      SET codice_prod = SUBSTRING(codici, pos, len - pos);
      -- SUBSTRING(stringa, inizio, fine) ritorna la substring corrisponde

      -- Eseguire l'operazione sulle tabelle per ciascun codice
      IF codice_prod != '' THEN
          -- Aggiornare lo stato dell'ordine
          UPDATE Copie
          SET stato = 'venduto'
          WHERE Copie.codice = codice_prod;

          INSERT INTO dettagli_ordini (prodotto, codice_ordine) VALUES
          (codice_prod, my_codice_ordine);
      END IF;
      SET pos = len + 1;
   END WHILE;
   SET totale_ordine =
   (SELECT SUM(copie.prezzo) FROM dettagli_ordini INNER JOIN copie on prodotto = copie.codice WHERE codice_ordine = my_codice_ordine)
   + (SELECT prezzo FROM società_di_trasporti WHERE nome = trasporto);

    SET importo_totale = totale_ordine;
    UPDATE ordini SET importo = totale_ordini WHERE codice = my_codice_ordine;
   -- Completare la transazione
   COMMIT;
END §§
DELIMITER ;
```

7. Notifica per utenti: Funzione che restituisce in output una stringa formattata nel seguente modo:
“numero_utente1;numero_utente2;numero_utente3;...” in modo da inviare una notifica di completamento a tutti gli utenti che avevano un ordine in sospeso e non avevano ancora ricevuto notifica.

```
-- Raccogliere le informazioni su utenti a cui è arrivato l'ordine per inviare la notifica
DELIMITER §§
CREATE PROCEDURE notifica_utenti(OUT lista_numeri varchar(4000))
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE numero_tel VARCHAR(20);
    DECLARE temp VARCHAR(4000) DEFAULT '';

    DECLARE cursore CURSOR FOR
    SELECT telefono FROM Ordini join Utenti on Ordini.utente = Utenti.username WHERE Ordini.status = 'completato' AND Ordini.notificato = 0;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    OPEN cursore;

    WHILE (done = 0) DO
        FETCH cursore INTO numero_tel;
        IF done = 0 THEN
            SET temp = CONCAT(numero_tel,';',temp);
        END IF;
    END WHILE;
    CLOSE cursore;
    UPDATE Ordini JOIN Utenti ON Ordini.utente = Utenti.username SET notificato = 1 WHERE Ordini.status = 'completato' AND Ordini.notificato = 0;
    SET lista_numeri = temp;
END §§
DELIMITER ;
```