UNIVERSITÀ DEGLI STUDI DI SALERNO
Facoltà di informatica

Progetto: Database per e-commerce di videogame

Insegnamento: Basi di Dati

Studenti:
Giuseppe Caiazzo
Alessio Rizzolo

Docente:
Dott.ssa Genoveffa Tortora
Dott. Michele Risi

Anno Accademico 2017/2018

SOMMARIO

1 RACCOLTA DELLE SPECIFICHE DELLA REALTÀ DI INTERESSE

1.1 DESCRIZIONE SINTETICA DELLA REALTÀ DI INTERESSE

Si vuole modellare un sistema per la gestione di un **e-commerce di videogame** e dei clienti iscritti al sito dedito alla vendita. Ciascun cliente (identificato da ID, nome, cognome, età e dal numero di ordini effettuati) può effettuare uno o più ordini all'interno del sito, ma può anche decidere di non effettuarne.
Ogni ordine è identificato dal numero dell'ordine, da uno stato (che può essere: presa in carico, sospeso, in corso, concluso) e da un costo totale (dato composto dal costo unitario di tutti i prodotti e dal costo della spedizione, quando è presente). Un ordine deve avere: almeno un prodotto ma può averne N, una sola spedizione ma può anche non averne (momentaneamente) e un solo cliente di riferimento. Tra l'ordine e il prodotto vi è inoltre l'attributo quantità, che rappresenta per l'appunto la quantità di prodotti dello stesso tipo inseriti nell'ordine.
Un prodotto è formato dal codice prodotto (che lo identifica), da un nome, da una descrizione, da un costo unitario e da uno o più colori. Il prodotto viene poi suddiviso in videogame (di cui si vogliono conoscere uno o più generi a cui può appartenere, una piattaforma e un formato), in periferiche (di cui si vuole conoscere il tipo) e in console (della quale si vuole conoscere la versione).
Una spedizione, che fa riferimento ad un solo ordine, è identificata da un numero di spedizione, da un indirizzo e da un corriere. Si deve tener presente che tra la spedizione e l'ordine vi è un costo di spedizione, che dipende da ambedue le entità.
In fine, un cliente può effettuare una o più recensioni. Le recensioni sono caratterizzate da un titolo, da una descrizione e da un punteggio (un valore numerico tra 1 e 5). Una recensione è identificata univocamente dall'ID del cliente che ha scritto quella determinata recensione e dal codice del prodotto alla quale la recensione si riferisce.

1.2 DESCRIZIONE DETTAGLIATA DELLA REALTÀ DI INTERESSE

La realtà di interesse presa in considerazione è **un e-commerce online**, ovvero un negozio specializzato nella vendita di prodotti online; nello specifico si tratta di un e-commerce specializzato in prodotti a tema videoludico, come videogame, console e simili. Il sito vuole realizzare una base di dati per poter gestire la vendita dei propri prodotti e per tenere sotto controllo le movimentazioni dei vari clienti, che possono iscriversi al sito stesso.
Per quanto riguarda i clienti si vuole tener traccia dei loro dati anagrafici più importanti (nome, cognome e età). Per identificarli in modo univoco si introduce un nuovo attributo ID, un numero generato randomicamente che ci permette di distinguere ognuno dei clienti iscritti. Il numero di clienti attualmente iscritti al sito è pari a 100.
Per i prodotti, invece, si hanno i seguenti attributi: nome, descrizione, costo unitario e colore. Il costo unitario può variare nel tempo, in base ad eventuali offerte o a sovrapprezzi. Il colore è un attributo multivalore perché ogni prodotto può avere più di un colore. Per identificare un prodotto in maniera univoca si introduce l'attributo codice_prodotto, un valore alfanumerico che identifica ogni prodotto inserito nell'e-commerce. Il numero di prodotti attualmente disponibili sul sito è pari a 600.
Tra il prodotto ed il cliente vi è un rapporto molto importante, proprio perché si è interessati a gestire il rapporto tra queste due entità. Ogni cliente, infatti, può acquistare più prodotti, sia dello stesso tipo che di altre tipologie. Per gestire questa relazione tra le due entità abbiamo deciso di inserire l'entità "Ordine", in questo modo possiamo tener traccia anche degli ordini effettuati dai clienti. Ordine, inoltre, contiene le informazioni relative allo stato dell'ordine (ovvero se lo stesso è stato preso in carico, se è stato inviato, se è concluso o se è stato invece sospeso). Oltre ciò, consideriamo un attributo ridondante che può essere utile all'e-commerce: il costo totale di un ordine, all'interno dell'entità ordine. Questo attributo viene calcolato considerando tutti i prodotti che sono stati inseriti nell'ordine stesso. Al momento sono stati effettuati 200 ordini sul sito.
Tra ordine e prodotto vi è un intermediario che, in gergo, viene chiamato "carrello". Nel nostro modello ER sarà una relazione con un attributo quantità, per tenere traccia di quanti prodotti dello stesso tipo vengono aggiunti all'ordine. Quindi, l'attributo ridondante costo totale sarà calcolato come segue: per ogni prodotto inserito nel carrello si moltiplica il suo prezzo unitario per la quantità, alla fine tutti i risultati vengono sommati tra di loro per poter ottenere il costo totale relativo all'ordine stesso.
Dato che nell'e-commerce sono disponibili delle categorie ben precise di prodotto, abbiamo deciso di introdurre una specializzazione del prodotto, anche perché ognuna di queste categorie presenta degli attributi univoci:

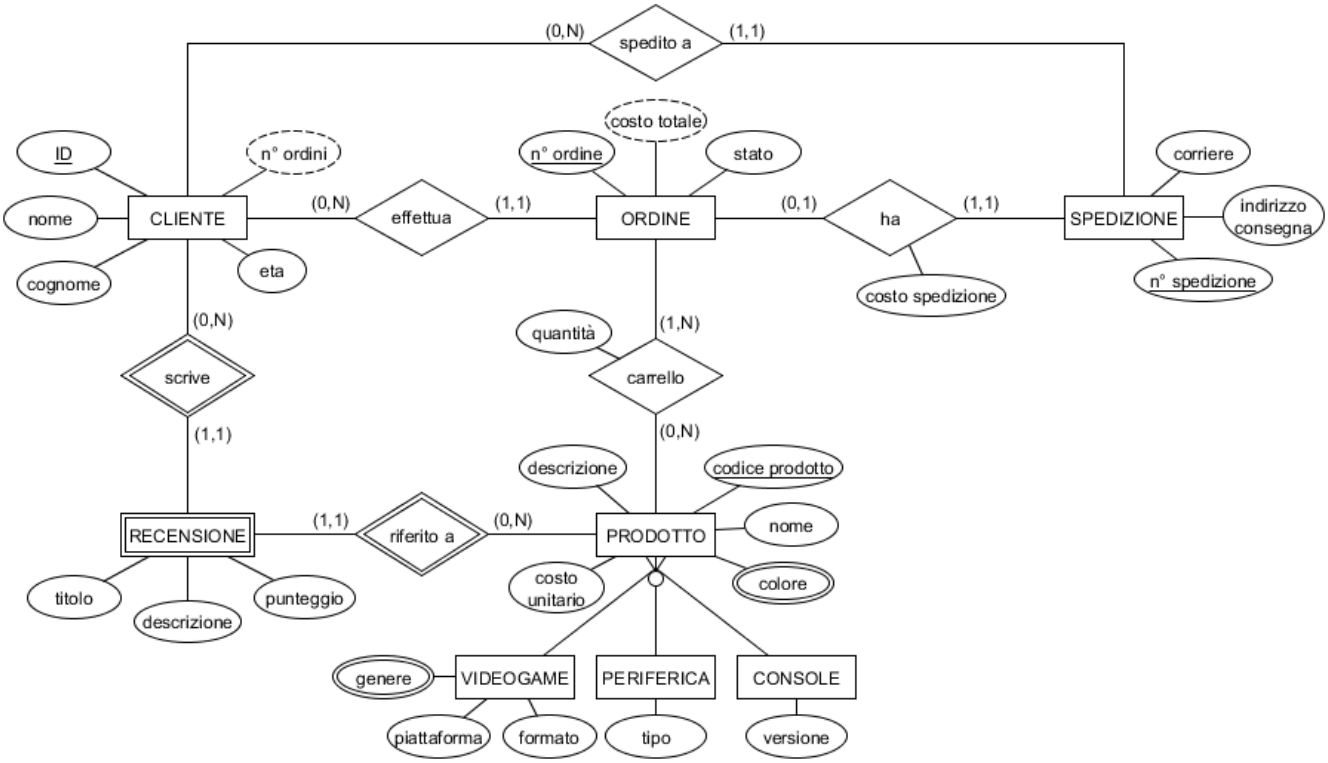
- La prima specializzazione sono i Videogame, ovvero tutti i videogiochi in vendita sul sito. Per ogni videogame si vuole tener traccia del genere al quale appartiene (avventura, sparatutto, corsa, ecc.), su quale piattaforma può essere giocato (ovvero con quale console è compatibile) e il formato con il quale viene distribuito (ogni videogame può essere o in formato fisico o in formato digitale). Sono disponibili 400 videogame diversi.
- La seconda specializzazione sono le periferiche, ovvero tutti gli hardware che consentono di giocare con una delle console, ogni periferica ha uno scopo differente. Le periferiche hanno un attributo tipo, che serve per distinguere nel dettaglio la tipologia di periferica (joystick, volante). Sono disponibili 150 periferiche diverse.
- La terza ed ultima specializzazione sono le console, ovvero gli hardware che consentono di poter giocare. Per ogni hardware si vuole conoscere la sua versione, dato che per ognuna di esse sono disponibili più versioni. Questa versione dipende da console a console, ad esempio può esserci una console disponibile in versione "lite" e in versione "pro", mentre può esserci una console che si evolve di anno in anno, in questo caso si ha una versione di tipo numerico. Sono disponibili 50 console diverse.

Per quanto riguarda l'ordine, il sito mette a disposizione un monitoraggio della sua spedizione, per questo motivo abbiamo introdotto una nuova entità chiamata, per l'appunto, Spedizione, che si collega all'entità ordine in quanto ogni ordine ha una propria spedizione (ma può anche non averne perché un ordine appena effettuato non verrà immediatamente spedito). L'entità spedizione è composta dal corriere che effettua la spedizione, dall'indirizzo di consegna (che può variare da cliente a cliente) e dal numero della spedizione, un codice alfanumerico che identifica univocamente ogni spedizione. Tra spedizione e ordine vi è anche un costo di spedizione, che può dipendere sia dal costo totale (se vi sono eventuali promozioni/sconti) sia dall'indirizzo di consegna stesso. Dato che si è aggiunto un nuovo costo, si decide di sommarlo al costo totale nell'entità ordine, per rendere più efficiente l'attributo stesso.
Un'ultima caratteristica dell'e-commerce riguarda le recensioni, ogni cliente iscritto al sito può eseguire una recensione, sia che abbia acquistato un prodotto o meno. Una recensione è formata da un titolo, da una breve descrizione e da una votazione (un numero da 1 a 5). Dato che una recensione non ha una chiave che la identifica in maniera univoca si decide di renderla entità debole, prendendo le chiavi di Cliente e di Prodotto si può identificare in modo univoco ogni recensione (ogni Cliente può scrivere una sola recensione per lo stesso prodotto). Al momento ci sono 1.200 recensioni sul sito, questo perché anche chi non ha acquistato quel determinato prodotto può scrivere una recensione per lo stesso.

1.3 GLOSSARIO

Termine	*Descrizione*	*Sinonimi*	*Collegamenti*
Cliente	Un cliente iscritto all'e-commerce	Consumatore, acquirente	Recensione, Ordine
Recensione	Una descrizione scritta da un cliente, relativa ad un prodotto	Giudizio	Cliente, Prodotto
Prodotto	Un prodotto in vendita sull'e-commerce	Merci, articolo	Ordine, Recensione, Videogame, Periferiche, Console
Ordine	Un ordine per poter acquistare uno o più prodotti	Acquisto	Cliente, Spedizione, Prodotto
Spedizione	Una spedizione per poter inviare l'ordine ad un cliente	Invio prodotti	Ordine
Videogame	Un videogame che può appartenere ad una o a più categorie	Videogioco, gioco	Prodotto
Periferiche	Una periferica per poter giocare con la console	Joystick, controller	Prodotto
Console	Una console per poter giocare ai videogame	Calcolatore	Prodotto

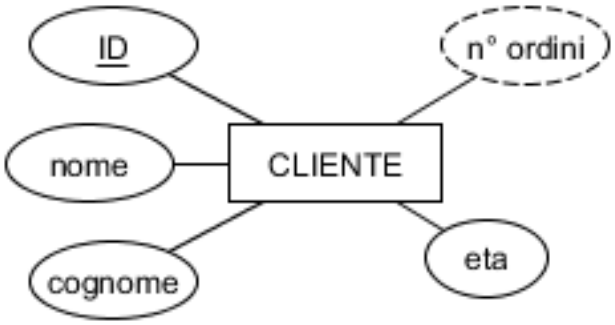
2 COSTRUZIONE DEL MODELLO EER



2.1 DESCRIZIONE DELLE ENTITÀ

Entità CLIENTE

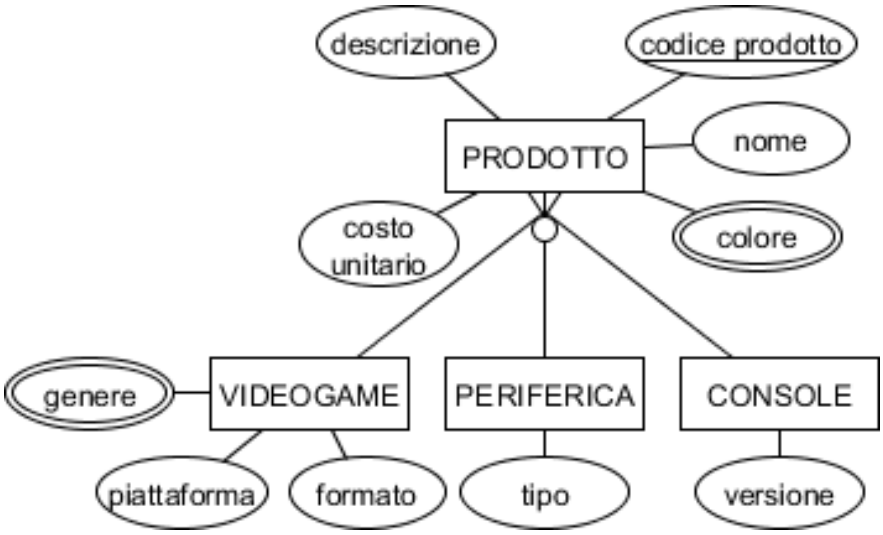
Rappresenta una persona fisica che si registra al sito di e-commerce.



Attributo chiave ID: un codice numerico a 5 cifre che identifica univocamente il cliente iscritto al sito.
Attributo nome*:* una stringa contenente il nome del cliente.
Attributo cognome*:* una stringa contenente cognome del cliente.
Attributo età*:* l'età del cliente che si iscrive all'e-commerce.
Attributo ridondante n° ordini*:* un intero che rappresenta il numero di ordini effettuati dal cliente all'interno dell'e-commerce.

Entità PRODOTTO

Gli articoli in vendita sul sito di e-commerce. I prodotti sono suddivisi in tre categorie: videogame, periferiche e console. Dato che le tre categorie hanno degli attributi in comune si decide di inserire una specializzazione di prodotto, mantenendo distinte le varie categorie dato che hanno degli attributi a parte.



Attributo chiave codice_prodotto: un codice alfanumerico che identifica univocamente il prodotto.
Attributo nome: una stringa contenente il nome del prodotto.
Attributo costo unitario: un float che indica il costo di un singolo prodotto, può variare nel tempo.
Attributo descrizione: una stringa con una breve descrizione del prodotto.
Attributo multivalore colore: un attributo stringa che descrive un colore per tutte le tipologie di prodotti, può assumere più valori. Nel caso dei videogame descrive un solo colore chiave della console associata (verde nel caso dell'xbox, azzurro nel caso della playstation, ecc.), nel caso di periferica e console si riferisce semplicemente al colore dell'hardware di riferimento.
Attributo multivalore genere: un attributo stringa dell'entità videogame; rappresenta una tipologia di videogame (Es: sparatutto, sportivo, simulazione, …).
Attributo piattaforma: un attributo stringa dell'entità videogame che indica su quale piattaforma è possibile scaricare il videogame in questione (nel caso di giochi digitali) o su quale console è possibile giocarci (nel caso di giochi fisici). Si deve tener presente che non è un attributo multivalore in quanto, per uno stesso gioco disponibile su più piattaforme, vengono inseriti tanti prodotti quante sono le piattaforme disponibili.
Attributo formato: un attributo stringa di videogame che può assumere due valori: fisico o digitale.
Attributo tipo: un attributo stringa di periferica che descrive nel dettaglio il tipo di periferica (ES: joystick, volante, …).
Attributo versione: un attributo stringa dell'entità console che rappresenta la versione della console presa in considerazione (ES: 1, 2, 3, pro, lite …).

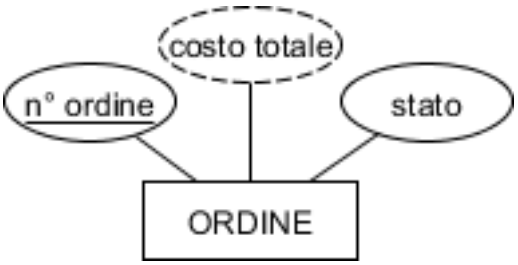
Entità SPEDIZIONE

Contiene i dati relativi alle spedizioni di un ordine effettuato da uno dei clienti iscritto al sito di e-commerce.

Attributo chiave n° spedizione: un codice alfanumerico di max 20 caratteri che identifica la spedizione.
Attributo corriere: una stringa con il nome del corriere che effettuerà la spedizione.
Attributo indirizzo di consegna: un alfanumerico con l'indirizzo al quale il corriere dovrà recapitare l'ordine del cliente (composto dall'indirizzo e dal numero civico).

Entità ORDINE

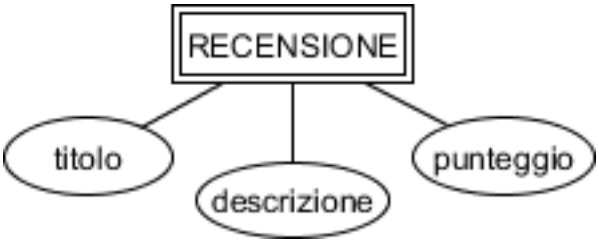
Contiene i dati relativi ad un determinato ordine effettuato dal cliente all'interno dell'e-commerce.



Attributo chiave n° ordine: è un codice alfanumerico che identifica univocamente un ordine effettuato dal cliente.
Attributo stato: una stringa che identifica se l'ordine è stato preso in carico, sospeso, in corso o concluso.
Attributo ridondante costo totale: un float che viene calcolato a partire dal costo unitario dei prodotti acquistati, dal costo di spedizione (se presente) e dalla quantità di ogni singolo prodotto.

Entità RECENSIONE

Una recensione è una valutazione che può essere eseguita da ogni cliente iscritto al sito di e-commerce. La recensione è caratterizzata univocamente dal nome del cliente e dal nome del prodotto sul quale il cliente scrive la recensione; per questo motivo l'entità recensione è *debole*.



Attributo titolo: una stringa con il titolo che il cliente può assegnare alla propria recensione.
Attributo descrizione: una stringa con una descrizione generale sul prodotto in questione.
Attributo punteggio: un valore numerico compreso tra 1 e 5 che rappresenta il grado di soddisfazione per quel determinato prodotto.

2.2 DESCRIZIONE DELLE RELAZIONI

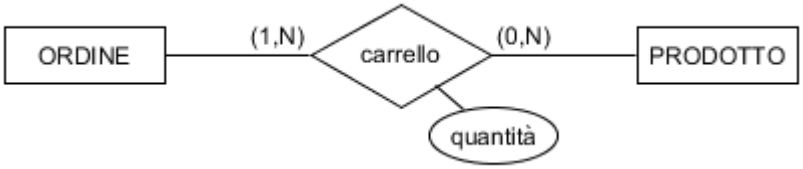
Relazione *effettua*

La relazione "*effettua*" intercorre tra l'entità CLIENTE e l'entità ORDINE.

La relazione serve per permettere ad un cliente registrato di effettuare un ordine all'interno del sito di e-commerce. Un cliente può decidere di non effettuare ordini o può effettuarne N; mentre un ordine fa riferimento ad un solo cliente.

Relazione *carrello*

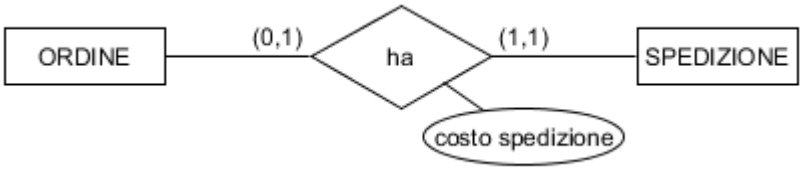
La relazione "*carello*" lega l'entità ORDINE all'entità PRODOTTO.



La relazione serve per tenere traccia di tutti i prodotti all'interno di un ordine. Un ordine deve avere almeno un prodotto ma può averne anche N; un prodotto può non far riferimento ad un ordine o può appartenere ad N ordini. La relazione, inoltre, ha un attributo *quantità* che indica la quantità di prodotti aggiunta (per ogni singolo prodotto inserito all'interno del carrello), questo andrà a diminuire il numero di accessi (senza questo dato, se il cliente voleva acquistare due prodotti uguali avrebbe dovuto inserire due volte lo stesso prodotto).

Relazione *ha*

La relazione "*ha*" lega l'entità ORDINE all'entità SPEDIZIONE.



La relazione consente di spedire un determinato ordine (ad un indirizzo assegnato nell'entità spedizione). Un ordine può riferirsi ad un'unica spedizione e una spedizione può riferirsi ad un unico ordine. L'ordine, inoltre, può anche non avere una spedizione (verrà aggiunta in un secondo momento). La relazione, inoltre, ha un attributo *costo spedizione*, che può variare in base a dove verrà spedito quel determinato ordine o al costo totale dell'ordine stesso.

Relazione *scrive*

La relazione "*scrive"* lega l'entità CLIENTE all'entità debole RECENSIONE.

La relazione permette ad un cliente di scrivere una recensione. Un cliente può non scrivere una recensione o può scriverne N, mentre una recensione fa riferimento ad un unico cliente. Essendo l'entità recensione un'entità debole, la relazione ci consente di prendere come chiave esterna la chiave di cliente.

Relazione *riferito a*

La relazione *"riferito a"* lega l'entità RECENSIONE all'entità PRODOTTO.

Una recensione può far riferimento ad un unico prodotto, mentre un prodotto può non avere recensioni o può averne N. Essendo l'entità recensione un'entità debole, la relazione ci consente di prendere come chiave esterna la chiave di prodotto.

3 ELENCO DELLE OPERAZIONI E CARICO APPLICATIVO

3.1 ELENCO DELLE OPERAZIONI

Operazione 1: aggiungi un nuovo cliente.
Un nuovo cliente si iscrive al sito di e-commerce, inserendo i propri dati anagrafici.

Operazione 2: aggiungi un nuovo videogame.
Viene inserito un nuovo videogame all'interno del sito di e-commerce, specificando solo un genere a cui appartiene.

Operazione 3: aggiungi una nuova recensione relativa ad un prodotto.
Un cliente scrive una nuova recensione che fa riferimento ad un determinato prodotto dando anche un voto compreso tra uno e cinque.

Operazione 4: aggiunge un nuovo ordine relativo ad un cliente.
Un cliente (già iscritto al sito) effettua un nuovo ordine.

Operazione 5: stampa il numero totale di ordini effettuati da tutti i clienti.
Vogliamo conoscere quanti ordini sono stati effettuati dai clienti iscritti all'e-commerce (per sapere quanti ordini vengono effettuati al giorno).

Operazione 6: stampa il costo totale di un ordine effettuato da un cliente.
Si vuole stampare il costo totale di un ordine.

Operazione 7: stampa tutte le spedizioni eseguite sull'e-commerce.
Vengono stampate tutte le spedizioni che sono state eseguite dal sito.

Operazione 8: stampa il numero di videogame che sono presenti nel database, raggruppandoli in base al loro genere.
Vengono stampati i nomi dei generi presenti nel DB con il numero di copie presenti per quel determinato genere.

Operazione 9: per ogni cliente che ha effettuato almeno un ordine, si vuole stampare il loro numero di ordini effettuati e, in più, il costo totale medio di tutti i loro ordini.
Vengono stampati ID cliente, numero ordini e costo totale medio.

3.2 TAVOLE DELLE OPERAZIONI E DEI VOLUMI

Tavola delle operazioni		
Operazione	Tipo	Frequenza
Op1	I	10/g
Op2	I	7/g
Op3	I	20/g
Op4	I	8/g
Op5	I	1/g
Op6	I	8/g
Op7	B	12/a
Op8	I	8/g
Op9	B	12/a

Tavola dei volumi		
Concetto	Tipo	Volume
CLIENTE	E	100
ORDINE	E	200
SPEDIZIONE	E	200
PRODOTTO	E	600
VIDEOGAME	E	400
PERIFERICA	E	150
CONSOLE	E	50
RECENSIONE	E	1.200
effettua	R	200
carrello	R	400
ha	R	200
scrive	R	1.200
riferito a	R	1.200

4 PROGETTAZIONE LOGICA

4.1 RISTRUTTURAZIONE

4.1.1 Eliminazione delle generalizzazioni

Nel modello EER è presente una sola specializzazione nell'entità PRODOTTO. Dato che tutte e tre le entità figlie presentano degli attributi non ci conviene incorporare le figlie nel padre altrimenti si avrebbero troppi attributi di tipo null. Per questo motivo decidiamo di mantenere le figlie, trasformando la specializzazione in tre relazioni con le tre entità.

4.1.2 Partizionamento/accorpamento di entità e associazioni

Eliminazione di attributi multivalore

Gli attributi multivalore presenti nel modello EER sono due:

- Nell'entità PRODOTTO trasformiamo l'attributo multivalore colore in un'entità debole, aggiungendo un nuovo attributo "nome_colore" (che avrà le stesse caratteristiche dell'attributo multivalore). Un prodotto deve avere almeno un colore ma può averne anche N, un colore si riferisce ad un unico prodotto.
- Nell'entità VIDEOGAME trasformiamo l'attributo multivalore genere in un'entità, aggiungendo un nuovo attributo chiave "nome_genere" (che avrà le stesse caratteristiche dell'attributo multivalore, ma in questo caso l'attributo identifica univocamente la sua entità di riferimento). Un videogame deve avere almeno un genere ma può averne più di uno, un genere può non riferirsi a nessun videogame ma può riferirsi a più di uno.
- Abbiamo deciso di non trasformare l'attributo GENERE in un'entità debole in quanto, all'interno dell'e-commerce, può essere comodo visualizzare un elenco di tutti i videogame appartenenti ad un determinato genere.

4.1.3 Analisi delle ridondanze

Nel modello EER sono presenti due attributi ridondanti:

- L'attributo n°ordini nell'entità CLIENTE;
- L'attributo costo_totale nell'entità ORDINE.

Vogliamo studiare tutte le operazioni che coinvolgono questi dati ridondanti per verificare se conviene mantenerli o meno. Costruiamo le tavole degli accessi considerando i seguenti accessi: 1L = 1, 1S = 2.

Dato ridondante: *n° ordini*

Vogliamo studiare il dato ridondante *n° ordini* dell'entità CLIENTE per capire se ci conviene mantenere il dato o se dobbiamo eliminarlo. Le operazioni che coinvolgono il dato ridondante sono: Op4, Op5 e Op9. Op9 non viene considerata dato che è Brench (viene eseguita 12 volte l'anno).

Op4: aggiunge un nuovo ordine relativo ad un cliente.

Tavola degli accessi Op4								
Con dato ridondante				Senza dato ridondante				
ORDINE	E	1	S	ORDINE	E	1	S
carrello	R	2	S	carrello	R	2	S
effettua	R	1	S	effettua	R	1	S
CLIENTE	E	1	L				
CLIENTE	E	1	S				
Accessi: 1L + 5S = 11 accessi 11*8/g (frequenza Op4) = **88**				**Accessi:** 0L + 4S = 8 accessi 8*8/g (frequenza Op4) = **64**				

Op5: stampa il numero totale di ordini effettuati da tutti i clienti iscritti all'E-commerce.

Tavola degli accessi Op5							
Con dato ridondante				Senza dato ridondante			
CLIENTE	E	100	L	CLIENTE	E	100	L
				effettua	R	200	L
Accessi: 100L + 0S = 100 accessi 100*1/g (frequenza Op5) = **100**				**Accessi:** 300L + 0S = 300 accessi 300*1/g (frequenza Op5) = **300**			

Memoria occupata:

Considerando 2 byte per memorizzare il dato "*n°ordini*", in totale si hanno 2*100 = 200 byte.

Accessi:

Totale accessi con dato ridondante: 188.
Totale accessi senza dato ridondante: 364.

In definitiva, tenendo conto sia della memoria occupata che del numero di accessi con e senza dato ridondante, conviene **mantenere il dato *n° ordini***.

Dato ridondante: *costo totale*

Vogliamo studiare il dato ridondante *costo totale* dell'entità ORDINE per capire se ci conviene mantenere il dato o se dobbiamo eliminarlo. Le operazioni che coinvolgono il dato ridondante sono: Op4, Op6 e Op9. Op9 non viene considerata dato che è Brench (viene eseguita 12 volte l'anno).

Op4: aggiunge un nuovo ordine relativo ad un cliente.

Tavola degli accessi Op4							
Con dato ridondante				Senza dato ridondante			
ORDINE	E	1	S	ORDINE	E	1	S
carrello	R	2	S	carrello	R	2	S
PRODOTTO	E	2	L	effettua	R	1	S
effettua	R	1	S	ha	R	1	S
ha	R	1	S				
Accessi: 2L + 5S = 12 accessi 12*8/g (frequenza Op4) = **96**				**Accessi:** 0L + 5S = 10 accessi 8*8/g (frequenza Op4) = **80**			

Op6: stampa il costo totale di un ordine relativo ad un cliente.

Tavola degli accessi Op6							
Con dato ridondante				Senza dato ridondante			
ORDINE	E	1	L	ORDINE	E	1	L
				carrello	R	2	L
				PRODOTTO	E	2	L
				ha	R	1	L
Accessi: 1L + 0S = 1 accessi 1*8/g (frequenza Op6) = **8**				**Accessi:** 6L + 0S = 6 accessi 6*8/g (frequenza Op6) = **48**			

Memoria occupata:

Considerando 2 byte per memorizzare il dato "*costo totale*", in totale si hanno 2*200 = 400 byte.

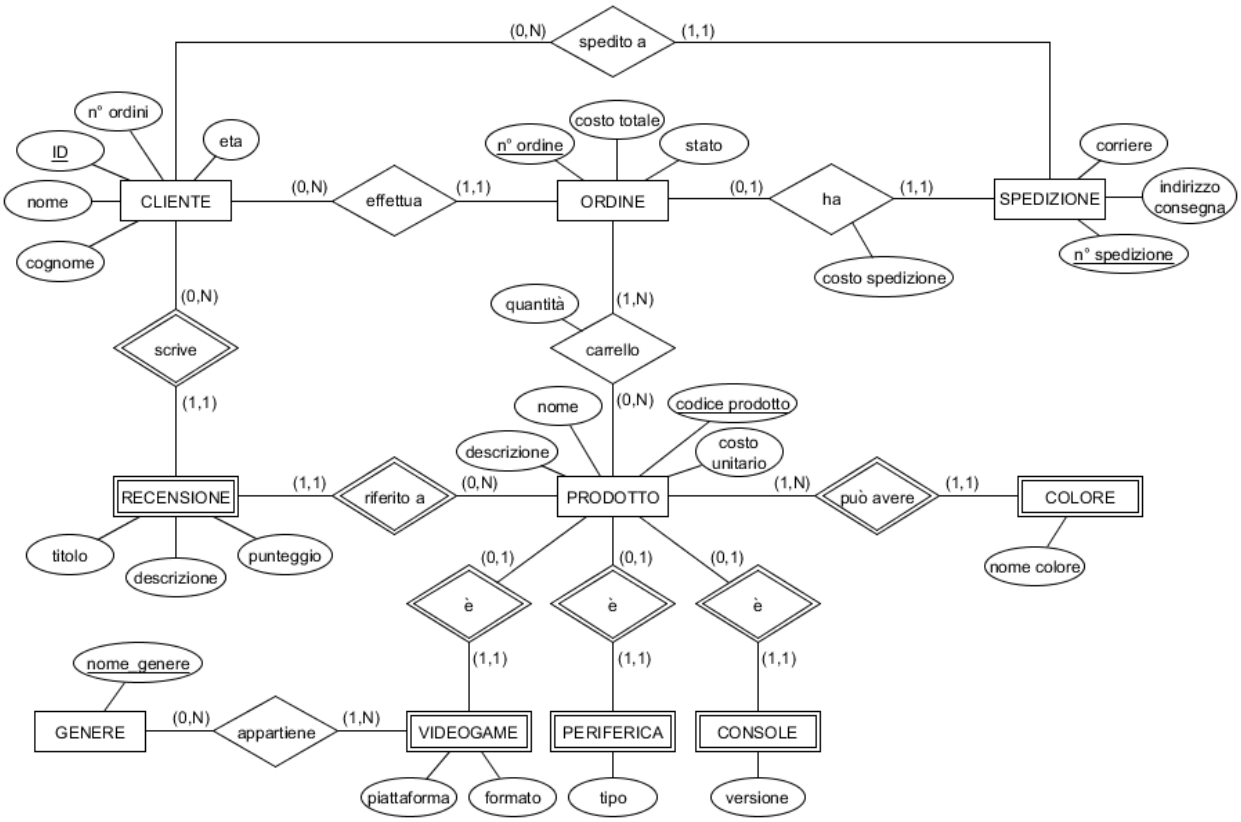
Accessi:

Totale accessi con dato ridondante: 104.
Totale accessi senza dato ridondante: 128.

In definitiva, tenendo conto sia della memoria occupata che del numero di accessi con e senza dato ridondante, è più conveniente **mantenere il dato *costo totale*.**

4.2 SCHEMA RISTRUTTURATO

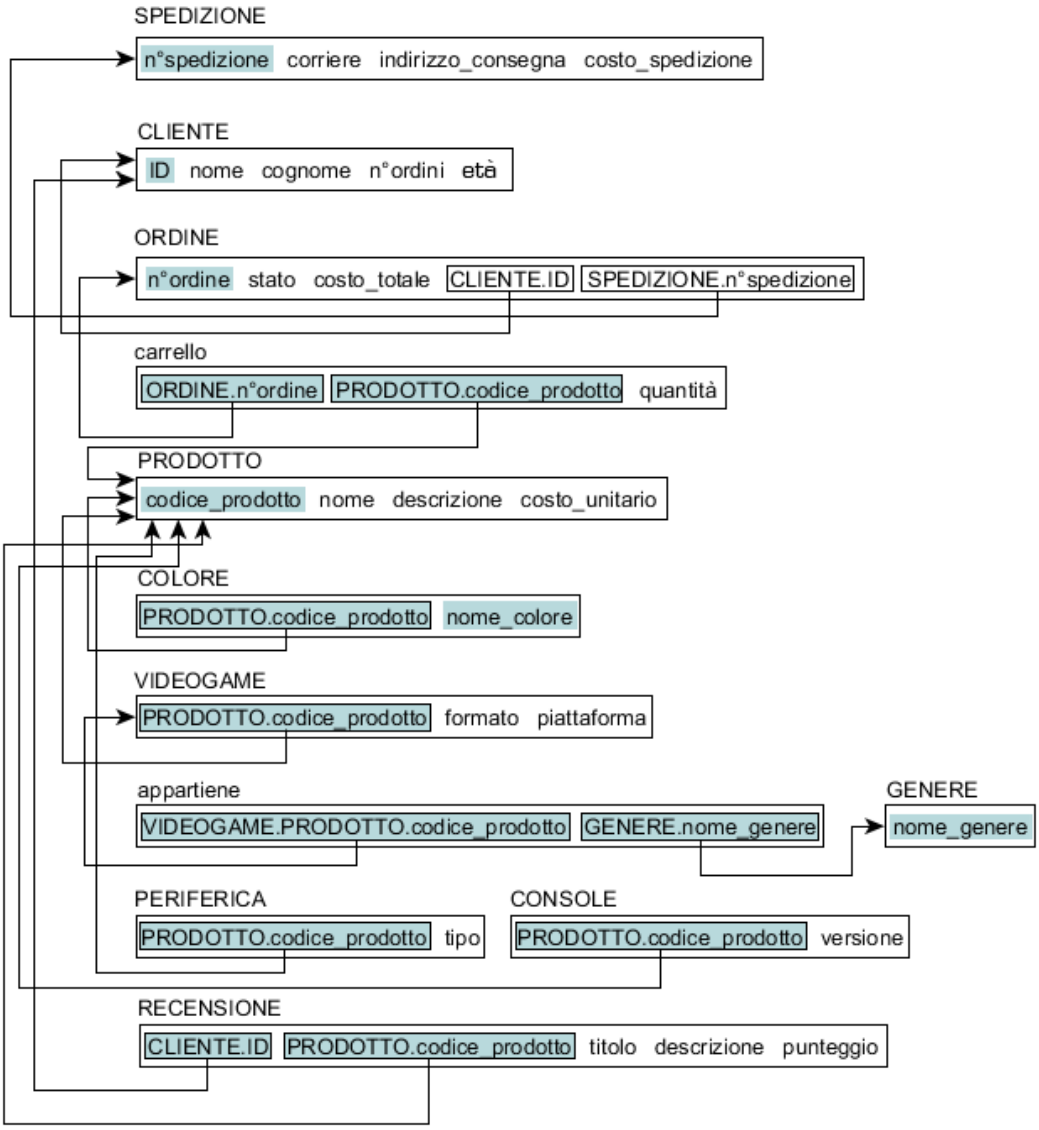
Il modello EER ristrutturato (tenendo conto della documentazione riportata in precedenza) è il seguente:



4.3 TRADUZIONE VERSO IL RELAZIONALE

CLIENTE (ID, nome, cognome, n°ordini, età)
ORDINE (n°ordine, stato, costo_totale, CLIENTE.ID↑, SPEDIZIONE.n°spedizione*↑)
SPEDIZIONE (n°spedizione, corriere, indirizzo_consegna, costo_spedizione)
carrello (ORDINE. n°ordine↑, PRODOTTO.codice_prodotto↑, quantità)
PRODOTTO (codice_prodotto, nome, descrizione, costo_unitario)
COLORE (PRODOTTO.codice_prodotto↑, nome_colore)
VIDEOGAME (PRODOTTO.codice prodotto↑, formato, piattaforma)
appartiene (VIDEOGAME.PRODOTTO.codice_prodotto↑, GENERE.nome genere↑)
GENERE (nome genere)
PERIFERICA (PRODOTTO.codice prodotto↑, tipo)
CONSOLE (PRODOTTO.codice prodotto↑, versione)
RECENSIONE (CLIENTE.ID↑, PRODOTTO.codice_prodotto↑, titolo, descrizione, punteggio)

4.4 Modello relazionale



4.5 Normalizzazione del modello relazionale

Vogliamo normalizzare, finché è possibile, lo schema relazionale che abbiamo riportato.

Verifichiamo se lo schema è in prima forma normale:

- Non devono esserci attributi non atomici.

Data la definizione stessa di relazione possiamo dire che il modello relazionale è già in 1NF.

Verifichiamo se lo schema è in seconda forma normale:

- Lo schema deve essere già in 1NF (e lo abbiamo verificato);
- Tutti i campi non chiave devono dipendere dall'intera chiave primaria (e non solo da una parte di essa)

Possiamo facilmente verificare che il modello relazionale rispetta entrambi i punti e quindi è già in 2NF.

Verifichiamo se lo schema è in terza forma normale:

- Lo schema deve essere già in 2NF (e lo abbiamo verificato);
- Tutti gli attributi non chiave dipendono direttamente dalla chiave.

Possiamo facilmente verificare che il modello relazionale rispetta entrambi i punti e quindi è già in 3NF.

5 BASE DI DATI SU MYSQL

5.1 LISTATO ISTRUZIONI PER CREARE LA BASE DI DATI IN MYSQL

```
#
# Structure for table "carrello"
#

DROP TABLE IF EXISTS `carrello`;
CREATE TABLE `carrello` (
  `ordine_nordine` varchar(20) NOT NULL,
  `prodotto_codiceProdotto2` varchar(10) NOT NULL,
  `quantità` int(99) NOT NULL,
  PRIMARY KEY (`ordine_nordine`,`prodotto_codiceProdotto2`),
  KEY `prodotto_codiceProdotto2_idx` (`prodotto_codiceProdotto2`),
  CONSTRAINT `ordine_nordine` FOREIGN KEY (`ordine_nordine`) REFERENCES
`ordine` (`nordine`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `prodotto_codiceProdotto2` FOREIGN KEY
(`prodotto_codiceProdotto2`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "cliente"
#

DROP TABLE IF EXISTS `cliente`;
CREATE TABLE `cliente` (
  `ID` int(5) unsigned zerofill NOT NULL,
  `nome` char(24) NOT NULL,
  `cognome` char(24) NOT NULL,
  `nordini` int(10) unsigned DEFAULT NULL,
  `eta` int(3) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `ID_UNIQUE` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "genere"
#

DROP TABLE IF EXISTS `genere`;
CREATE TABLE `genere` (
  `nome_genere` char(45) NOT NULL,
  PRIMARY KEY (`nome_genere`),
  UNIQUE KEY `nome_genere_UNIQUE` (`nome_genere`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "prodotto"
#

DROP TABLE IF EXISTS `prodotto`;
CREATE TABLE `prodotto` (
  `codice_prodotto` varchar(10) NOT NULL,
  `nome` mediumtext NOT NULL,
  `descrizione` mediumtext NOT NULL,
```

```
  `costo_unario` float unsigned NOT NULL,
  PRIMARY KEY (`codice_prodotto`),
  UNIQUE KEY `codice_prodotto_UNIQUE` (`codice_prodotto`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "periferica"
#

DROP TABLE IF EXISTS `periferica`;
CREATE TABLE `periferica` (
  `prodotto1_codice_prodotto` varchar(10) NOT NULL,
  `tipo` char(35) NOT NULL,
  PRIMARY KEY (`prodotto1_codice_prodotto`),
  UNIQUE KEY `prodotto1_codice_prodotto_UNIQUE`
(`prodotto1_codice_prodotto`),
  CONSTRAINT `prodotto1_codice_prodotto` FOREIGN KEY
(`prodotto1_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "appartiene"
#

DROP TABLE IF EXISTS `appartiene`;
CREATE TABLE `appartiene` (
  `videogame_prodotto_codice_prodotto` varchar(10) NOT NULL,
  `genere_nome_genere` char(45) NOT NULL,
  PRIMARY KEY (`videogame_prodotto_codice_prodotto`,`genere_nome_genere`),
       UNIQUE KEY  PRIMARY KEY
(`videogame_prodotto_codice_prodotto`,`genere_nome_genere`),
  KEY `genere_nome_genere` (`genere_nome_genere`),
  CONSTRAINT `genere_nome_genere` FOREIGN KEY (`genere_nome_genere`)
REFERENCES `genere` (`nome_genere`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `videogame_prodotto_codice_prodotto` FOREIGN KEY
(`videogame_prodotto_codice_prodotto`) REFERENCES `prodotto`
(`codice_prodotto`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "console"
#

DROP TABLE IF EXISTS `console`;
CREATE TABLE `console` (
  `prodotto2_codice_prodotto` varchar(10) NOT NULL,
  `versione` varchar(15) NOT NULL,
  PRIMARY KEY (`prodotto2_codice_prodotto`),
  UNIQUE KEY `prodotto2_codice_prodotto_UNIQUE`
(`prodotto2_codice_prodotto`),
  CONSTRAINT `prodotto2_codice_prodotto` FOREIGN KEY
(`prodotto2_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "colore"
#
```

```
DROP TABLE IF EXISTS `colore`;
CREATE TABLE `colore` (
  `prodotto3_codice_prodotto` varchar(10) NOT NULL,
  `nome_colore` char(30) NOT NULL,
  PRIMARY KEY (`prodotto3_codice_prodotto`,`nome_colore`),
       UNIQUE KEY (`prodotto3_codice_prodotto`,`nome_colore`),
  CONSTRAINT `prodotto3_codice_prodotto` FOREIGN KEY
(`prodotto3_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "recensione"
#

DROP TABLE IF EXISTS `recensione`;
CREATE TABLE `recensione` (
  `cliente_id` int(5) unsigned zerofill NOT NULL,
  `prodotto_codice_prodotto` varchar(10) NOT NULL,
  `titolo` tinytext NOT NULL,
  `descrizione` mediumtext NOT NULL,
  `punteggio` int(1) NOT NULL,
  PRIMARY KEY (`cliente_id`,`prodotto_codice_prodotto`),
       UNIQUE KEY PRIMARY KEY (`cliente_id`,`prodotto_codice_prodotto`),
  KEY `prodotto_codice_prodotto_idx` (`prodotto_codice_prodotto`),
  KEY `cliente_idx` (`cliente_id`),
  CONSTRAINT `cliente` FOREIGN KEY (`cliente_id`) REFERENCES `cliente`
(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `prodotto_codice_prodotto` FOREIGN KEY
(`prodotto_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


#
# Structure for table "spedizione"
#

DROP TABLE IF EXISTS `spedizione`;
CREATE TABLE `spedizione` (
  `nspedizione` varchar(20) NOT NULL,
  `corriere` char(15) NOT NULL,
  `indirizzo_consegna` varchar(45) NOT NULL,
  `costo_spedizione` float unsigned NOT NULL,
  `cliente1_ID` int(5) unsigned NOT NULL,
  PRIMARY KEY (`nspedizione`),
  UNIQUE KEY `nspedizione_UNIQUE` (`nspedizione`),
  KEY `CLIENTE.ID_idx` (`cliente1_ID`),
  CONSTRAINT `cliente1_ID` FOREIGN KEY (`cliente1_ID`) REFERENCES `cliente`
(`ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "ordine"
#

DROP TABLE IF EXISTS `ordine`;
CREATE TABLE `ordine` (
```

```
  `nordine` varchar(20) NOT NULL,
  `stato` char(15) NOT NULL,
  `costo_totale` float unsigned NOT NULL,
  `cliente_id` int(5) unsigned zerofill NOT NULL,
  `spedizione_nspedizione` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`nordine`),
  UNIQUE KEY `nordine_UNIQUE` (`nordine`),
  KEY `cliente.id_idx` (`cliente_id`),
  KEY `spedizione_nspedizione_idx` (`spedizione_nspedizione`),
  CONSTRAINT `cliente_id` FOREIGN KEY (`cliente_id`) REFERENCES `cliente`
(`ID`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `spedizione_nspedizione` FOREIGN KEY (`spedizione_nspedizione`)
REFERENCES `spedizione` (`nspedizione`) ON DELETE NO ACTION ON UPDATE NO
ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Structure for table "videogame"
#

DROP TABLE IF EXISTS `videogame`;
CREATE TABLE `videogame` (
  `prodotto_codiceprodotto` varchar(10) NOT NULL,
  `formato` char(20) NOT NULL,
  `piattaforma` char(50) NOT NULL,
  PRIMARY KEY (`prodotto_codiceprodotto`),
  CONSTRAINT `prodotto _codiceprodotto` FOREIGN KEY
(`prodotto_codiceprodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

5.2 SCRIPT PER POPOLARE DA ZERO LA BASE DI DATI

```
#
# Structure for table "carrello"
#

DROP TABLE IF EXISTS `carrello`;
CREATE TABLE `carrello` (
  `ordine_nordine` varchar(20) NOT NULL,
  `prodotto_codiceProdotto2` varchar(10) NOT NULL,
  `quantità` int(99) NOT NULL,
  PRIMARY KEY (`ordine_nordine`,`prodotto_codiceProdotto2`),
  KEY `prodotto_codiceProdotto2_idx` (`prodotto_codiceProdotto2`),
  CONSTRAINT `ordine_nordine` FOREIGN KEY (`ordine_nordine`) REFERENCES
`ordine` (`nordine`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `prodotto_codiceProdotto2` FOREIGN KEY
(`prodotto_codiceProdotto2`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "carrello"
#

INSERT INTO `carrello` VALUES
('DFE394','H90YJH',2),('DHHDD99','HDON34',1),('FER35Y','ZFIEF43',3),('FKEJ45
5','JEGR9T',5),('GKL44H','ITUH9P',1);

#
# Structure for table "cliente"
#

DROP TABLE IF EXISTS `cliente`;
CREATE TABLE `cliente` (
  `ID` int(5) unsigned zerofill NOT NULL,
  `nome` char(24) NOT NULL,
  `cognome` char(24) NOT NULL,
  `nordini` int(10) unsigned DEFAULT NULL,
  `eta` int(3) NOT NULL,
  PRIMARY KEY (`ID`),
  UNIQUE KEY `ID_UNIQUE` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "cliente"
#

INSERT INTO `cliente` VALUES
(00001,'Mario','Rossi',2,15),(12356,'Giuseppe','Caiazzo',0,17),(23754,'Giuse
ppe','Pappalardo',0,32),(34295,'Alessio','Rizzolo',0,20),(45693,'Gian','Scal
a',0,11),(46824,'Nina','Peluso',0,24),(54985,'Dave','Smith',1,25),(67690,'Gi
ovanni','Caiazzo',1,50),(74938,'Michele','Genovesi',1,30),(78308,'Simone','A
uriemma',0,30),(85327,'Giuseppe','Sordano',0,67),(88348,'Ildegardo','Esposit
o',1,56);

#
# Structure for table "genere"
#
```

```
DROP TABLE IF EXISTS `genere`;
CREATE TABLE `genere` (
  `nome_genere` char(45) NOT NULL,
  PRIMARY KEY (`nome_genere`),
  UNIQUE KEY `nome_genere_UNIQUE` (`nome_genere`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "genere"
#

INSERT INTO `genere` VALUES
('Avventura'),('Azione'),('GDR'),('MMO'),('MMOPRG'),('Picchiaduro'),('Plattf
orm'),('Simulazione'),('Sparatutto');

#
# Structure for table "prodotto"
#

DROP TABLE IF EXISTS `prodotto`;
CREATE TABLE `prodotto` (
  `codice_prodotto` varchar(10) NOT NULL,
  `nome` mediumtext NOT NULL,
  `descrizione` mediumtext NOT NULL,
  `costo_unario` float unsigned NOT NULL,
  PRIMARY KEY (`codice_prodotto`),
  UNIQUE KEY `codice_prodotto_UNIQUE` (`codice_prodotto`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "prodotto"
#

INSERT INTO `prodotto` VALUES ('5674h','Spyro','Gioco di
Spyro',19.9),('GH5875','League of Legends','Gioco per
Pc',10),('H69864','Metin 2','Gioco Metin 2 per Pc',4.99),('H90YJH','Volante
da corsa \"Driving Force\"','Volante per tutti i tipi di console.
',250),('HDISK32','The Sims 4','Gioco di
simulazione',24.99),('HDON34','JoJo','Ciaone',100.99),('HFDU8','The Legend
of Zelda','Gioco della saga di Zelda per
Nintendo',35),('ITUH9P','Microfono','Microfono per
multiplayer',15.99),('JEGR9T','Cuffie Wireless','Cuffie per qualsiasi
piattaforma',39.99),('JRH8G8','Crash Bandicoot N. Sane Trilogy','Copia del
gioco Crash Bandicoot N. Sane Trilogy',19.99),('JRK4JK','Nintendo Classic
Mini','Console di gioco Nintendo Classic Mini di colore grigio. Dimensioni:
30X20',79.9),('KFEJ4T','God Of War 4','Gioco God of War per XBOX (digital
delivery)',49.99),('KJL569','PlayStation 4','PlayStation 4 da
1TB',235.98),('ZFIEF43','XBOX One','XBOX One da 1TB',229.99);

#
# Structure for table "periferica"
#

DROP TABLE IF EXISTS `periferica`;
CREATE TABLE `periferica` (
  `prodotto1_codice_prodotto` varchar(10) NOT NULL,
  `tipo` char(35) NOT NULL,
  PRIMARY KEY (`prodotto1_codice_prodotto`),
```

```
  UNIQUE KEY `prodotto1_codice_prodotto_UNIQUE`
(`prodotto1_codice_prodotto`),
  CONSTRAINT `prodotto1_codice_prodotto` FOREIGN KEY
(`prodotto1_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "periferica"
#

INSERT INTO `periferica` VALUES ('H90YJH','controller'),('ITUH9P','input
audio'),('JEGR9T','output audio');

#
# Structure for table "appartiene"
#

DROP TABLE IF EXISTS `appartiene`;
CREATE TABLE `appartiene` (
  `videogame_prodotto_codice_prodotto` varchar(10) NOT NULL,
  `genere_nome_genere` char(45) NOT NULL,
  PRIMARY KEY (`videogame_prodotto_codice_prodotto`,`genere_nome_genere`),
       UNIQUE KEY  PRIMARY KEY
(`videogame_prodotto_codice_prodotto`,`genere_nome_genere`),
  KEY `genere_nome_genere` (`genere_nome_genere`),
  CONSTRAINT `genere_nome_genere` FOREIGN KEY (`genere_nome_genere`)
REFERENCES `genere` (`nome_genere`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `videogame_prodotto_codice_prodotto` FOREIGN KEY
(`videogame_prodotto_codice_prodotto`) REFERENCES `prodotto`
(`codice_prodotto`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "appartiene"
#

INSERT INTO `appartiene` VALUES
('5674h','Avventura'),('5674h','Azione'),('GH5875','GDR'),('GH5875','MMO'),(
'H69864','Azione'),('H69864','MMOPRG'),('HDISK32','Simulazione'),('HDON34','
Avventura'),('HDON34','Picchiaduro'),('HFDU8','Avventura'),('HFDU8','Azione'
),('JRH8G8','Avventura'),('JRH8G8','Plattform'),('KFEJ4T','Azione'),('KFEJ4T
','Picchiaduro');

#
# Structure for table "console"
#

DROP TABLE IF EXISTS `console`;
CREATE TABLE `console` (
  `prodotto2_codice_prodotto` varchar(10) NOT NULL,
  `versione` varchar(15) NOT NULL,
  PRIMARY KEY (`prodotto2_codice_prodotto`),
  UNIQUE KEY `prodotto2_codice_prodotto_UNIQUE`
(`prodotto2_codice_prodotto`),
  CONSTRAINT `prodotto2_codice_prodotto` FOREIGN KEY
(`prodotto2_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Data for table "console"
#

INSERT INTO `console` VALUES ('JRK4JK','Classic
Mini'),('KJL569','4'),('ZFIEF43','One');

#
# Structure for table "colore"
#

DROP TABLE IF EXISTS `colore`;
CREATE TABLE `colore` (
  `prodotto3_codice_prodotto` varchar(10) NOT NULL,
  `nome_colore` char(30) NOT NULL,
  PRIMARY KEY (`prodotto3_codice_prodotto`,`nome_colore`),
       UNIQUE KEY (`prodotto3_codice_prodotto`,`nome_colore`),
  CONSTRAINT `prodotto3_codice_prodotto` FOREIGN KEY
(`prodotto3_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "colore"
#

INSERT INTO `colore` VALUES
('5674h','Azzuro'),('GH5875','Verde'),('H69864','Rosso'),('H90YJH','Blu'),('
HDISK32','Verde'),('HDON34','Viola'),('HFDU8','Verde'),('ITUH9P','Verde'),('
JEGR9T','Arancione'),('JRH8G8','Blu'),('JRK4JK','Bianco'),('KFEJ4T','Rosso')
,('KJL569','Blu'),('ZFIEF43','Bianco');

#
# Structure for table "recensione"
#

DROP TABLE IF EXISTS `recensione`;
CREATE TABLE `recensione` (
  `cliente_id` int(5) unsigned zerofill NOT NULL,
  `prodotto_codice_prodotto` varchar(10) NOT NULL,
  `titolo` tinytext NOT NULL,
  `descrizione` mediumtext NOT NULL,
  `punteggio` int(1) NOT NULL,
  PRIMARY KEY (`cliente_id`,`prodotto_codice_prodotto`),
       UNIQUE KEY PRIMARY KEY (`cliente_id`,`prodotto_codice_prodotto`),
  KEY `prodotto_codice_prodotto_idx` (`prodotto_codice_prodotto`),
  KEY `cliente_idx` (`cliente_id`),
  CONSTRAINT `cliente` FOREIGN KEY (`cliente_id`) REFERENCES `cliente`
(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `prodotto_codice_prodotto` FOREIGN KEY
(`prodotto_codice_prodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "recensione"
#
```

```
INSERT INTO `recensione` VALUES (00001,'HFDU8','Bello','Un bel
gioco',5),(54985,'JRH8G8','Male','Gioco rilasciato troppo presto e
ottimizzato male.Presenta numerosi bug, attenderò una
patch',2),(67690,'ITUH9P','Pessimo','Prodotto difettoso. Dopo due giorni di
utilizzo ha smesso di funzionare',1),(74938,'JEGR9T','Si trova di
meglio','Cuffie discrete. A quel prezzo si trova di
meglio',3),(88348,'JRK4JK','Quanti ricordi!','Console che mi ha fatto
ricordare i bei vecchi tempi di quan\'ero più piccolo. Qualità ottima',5);

#
# Structure for table "spedizione"
#

DROP TABLE IF EXISTS `spedizione`;
CREATE TABLE `spedizione` (
  `nspedizione` varchar(20) NOT NULL,
  `corriere` char(15) NOT NULL,
  `indirizzo_consegna` varchar(45) NOT NULL,
  `costo_spedizione` float unsigned NOT NULL,
  `cliente1_ID` int(5) unsigned NOT NULL,
  PRIMARY KEY (`nspedizione`),
  UNIQUE KEY `nspedizione_UNIQUE` (`nspedizione`),
  KEY `CLIENTE.ID_idx` (`cliente1_ID`),
  CONSTRAINT `cliente1_ID` FOREIGN KEY (`cliente1_ID`) REFERENCES `cliente`
(`ID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "spedizione"
#

INSERT INTO `spedizione` VALUES ('DEFEI39','TNT','Abbey
Road',0,54985),('DWU782','GLS','Via Alan
Turing',1.5,67690),('FERHF48','DHL','Via Tazio
Nuvolari',15.99,1),('KTO4I03','BARTOLINI','Via
Bartali',9.99,74938),('UREI30G','FEDEX','Via Togliatti',5.99,88348);

#
# Structure for table "ordine"
#

DROP TABLE IF EXISTS `ordine`;
CREATE TABLE `ordine` (
  `nordine` varchar(20) NOT NULL,
  `stato` char(15) NOT NULL,
  `costo_totale` float unsigned NOT NULL,
  `cliente_id` int(5) unsigned zerofill NOT NULL,
  `spedizione_nspedizione` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`nordine`),
  UNIQUE KEY `nordine_UNIQUE` (`nordine`),
  KEY `cliente.id_idx` (`cliente_id`),
  KEY `spedizione_nspedizione_idx` (`spedizione_nspedizione`),
  CONSTRAINT `cliente_id` FOREIGN KEY (`cliente_id`) REFERENCES `cliente`
(`ID`) ON DELETE NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT `spedizione_nspedizione` FOREIGN KEY (`spedizione_nspedizione`)
REFERENCES `spedizione` (`nspedizione`) ON DELETE NO ACTION ON UPDATE NO
ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
#
# Data for table "ordine"
#

INSERT INTO `ordine` VALUES ('DFE394','IN
CORSO',41.48,67690,'DWU782'),('DHHDD99','IN
CORSO',4.99,00001,NULL),('DWF323','SOSPESO',175,54985,'DEFEI39'),('FER35Y','
IN CORSO',275.98,88348,'UREI30G'),('FKEJ455','PRESA IN
CARICO',79.95,00001,'FERHF48'),('GKL44H','CONCLUSO',29.98,74938,'KTO4I03');

#
# Structure for table "videogame"
#

DROP TABLE IF EXISTS `videogame`;
CREATE TABLE `videogame` (
  `prodotto_codiceprodotto` varchar(10) NOT NULL,
  `formato` char(20) NOT NULL,
  `piattaforma` char(50) NOT NULL,
  PRIMARY KEY (`prodotto_codiceprodotto`),
  CONSTRAINT `prodotto _codiceprodotto` FOREIGN KEY
(`prodotto_codiceprodotto`) REFERENCES `prodotto` (`codice_prodotto`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

#
# Data for table "videogame"
#

INSERT INTO `videogame` VALUES
('5674h','Fisico','XBoX'),('GH5875','Digitale','PC'),('H69864','Digitale','P
C'),('HDISK32','Digitale','PC'),('HDON34','Fisico','PlayStation
4'),('HFDU8','Fisico','Nintendo'),('JRH8G8','Fisico','PlayStation
4'),('KFEJ4T','Digitale','XBOX');
```

6 APPLICAZIONE JAVA

```
package connessione;

import java.sql.*;
import java.sql.Date;
import java.util.*;
import java.io.*;
import entita.*;

@SuppressWarnings("unused")
public class ConnessioneMYsql {
	public static void main(String[] args) throws SQLException {
		Connection connessione; //connessione al DB

		/*CONNESSIONE*/
		//verifichiamo se la libreria è presente
		try {
			Class.forName("com.mysql.jdbc.Driver"); //verificare che sia presente la libreria che permette la connessione, caricamento del driver
		}
		catch (ClassNotFoundException e){ // se il driver non è caricato con successo
			System.out.println("ClassNotFoundException: ");
			System.err.println(e.getMessage());
		} //fine try-catch

		//Creo la connessione al database
		//3306 è la porta, da controllare
		connessione = DriverManager.getConnection("jdbc:mysql://localhost:3306/progetto0?user=root&password=alessio97");

		int scelta=1;
		Scanner input = new Scanner (System.in); //oggetto per l'input

		System.out.println("1: inserisci un nuovo cliente.");
		System.out.println("2: inserisci un nuovo videogame.");
		System.out.println("3: inserisci una nuova recensione relativa ad un prodotto.");
		System.out.println("4: inserisci un nuovo ordine relativo ad un cliente.");
		System.out.println("5: stampa il numero totale di ordini effettuati da tutti i clienti.");
		System.out.println("6: stampa il costo totale di un ordine effettuato da un cliente.");
		System.out.println("7: stampa tutte le spedizioni eseguite sull'e-commerce.");
		System.out.println("8: stampa il numero di videogiochi per ogni genere.");
		System.out.println("9: stampa il costo totale medio per i clienti che hanno effettuato almeno un ordine.");


		System.out.println("Si scelga il numero dell'operazione da eseguire: ");
```

```
		scelta = input.nextInt();

		switch (scelta) {
			case 1:
				op1(connessione);
				break;
			case 2:
				op2(connessione);
				break;
			case 3:
				op3(connessione);
				break;
			case 4:
				op4(connessione);
				break;
			case 5:
				op5(connessione);
				break;
			case 6:
				op6(connessione);
				break;
			case 7:
				op7(connessione);
				break;
			case 8:
				op8(connessione);
				break;
			case 9:
				op9(connessione);
				break;
			default:
				System.out.println("Errore nell'op scelta.\n");
				break;
		} //fine switch


		input.close(); //chiusura scanner
		connessione.close(); //chiusura connessione
}
```

```
	private static void op1(Connection connessione) throws SQLException {
		//OP1: inserisci un nuovo cliente

		Scanner input = new Scanner(System.in); //oggetto per l'input
		Scanner input_int = new Scanner(System.in); //oggetto per l'input
		Statement state = null; //per costruire le query
		String sql; //stringa contenete il codice

		System.out.println("\nInserisci ID cliente (numerico MAX 5 caratteri): ");
		int ID = input_int.nextInt();
		System.out.println("Inserisci nome cliente: ");
		String nome = input.nextLine();
		System.out.println("Inserisci cognome cliente: ");
		String cognome = input.nextLine();
		System.out.println("Inserisci età cliente: ");
		int eta = input_int.nextInt();

		Cliente cl = new Cliente(ID, nome, cognome, 0, eta);

		sql = "INSERT INTO cliente(ID, nome, cognome, nordini, eta) VALUES('" + cl.getID() + "','" +
cl.getNome() + "','" + cl.getCognome() + "','0','" + cl.getEta() + "')";


		try {
			state = connessione.createStatement();
			int n = state.executeUpdate(sql);
			if (n == 1) {
				System.out.println("\n**Nuovo cliente inserito**");
			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

		input.close();
		input_int.close();
	}
```

```
private static void op2(Connection connessione) throws SQLException {
	//OP2: inserisci un nuovo videogame

	Scanner input = new Scanner (System.in); //oggetto per l'input
	Scanner input_float = new Scanner (System.in); //oggetto per l'input
	Statement state = null; //per costruire le query
	ResultSet rs; //i risultati delle query (quindi dei record)
	String sql1;
	String sql2;

	System.out.println("\nInserisci codice prodotto: ");
	String codice_prodotto = input.nextLine();
	System.out.println("Inserisci nome prodotto: ");
	String nome_prodotto = input.nextLine();
	System.out.println("Inserisci descrizione_prodotto: ");
	String descrizione_prodotto = input.nextLine();
	System.out.println("Inserisci costo unario prodotto: ");
	float costo_unario = input_float.nextFloat();

	Prodotto pr = new Prodotto(codice_prodotto, nome_prodotto, descrizione_prodotto,
costo_unario);

	System.out.println("Inserisci formato prodotto (Fisico o Digitale): ");
	String formato_videogame = input.nextLine();
	System.out.println("Inserisci piattaforma prodotto: ");
	String piattaforma_videogame = input.nextLine();

	Videogame vd = new Videogame(pr.getCodice_prodotto(), formato_videogame,
piattaforma_videogame);

	sql1 = "INSERT INTO prodotto(codice_prodotto, nome, descrizione, costo_unario) VALUES('"
+ pr.getCodice_prodotto() + "','" +
			pr.getNome() + "','" + pr.getDescrizione() + "','" + pr.getCosto_unario() + "')";

	sql2 = "INSERT INTO videogame(prodotto_codiceprodotto, formato, piattaforma) VALUES('"
			+ vd.getProdottoCodice_prodotto() + "','" + vd.getFormato() + "','" +
vd.getPiattaforma() + "')";

	try {
		state = connessione.createStatement();
		int n = state.executeUpdate(sql1);
		if (n == 1) {
			System.out.println("\n**Nuovo prodotto inserito**");
		}
		n = state.executeUpdate(sql2);
		if (n == 1) {
			System.out.println("\n**Nuovo videogioco associato**");
		}
```

```
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

		sql1 = "SELECT * FROM genere";

		System.out.println("Inserisci il genere del prodotto (già esistente): ");
		String nome_genere = input.nextLine();
		Genere gn = null;
		try {
			state = connessione.createStatement();
			rs = state.executeQuery(sql1);

			while (rs.next() == true) {
				if(rs.getString("nome_genere").equals(nome_genere)) {
					gn = new Genere(rs.getString("nome_genere"));
				}
			}

		} catch (SQLException e) {
		System.out.println("errore: "+ e.getMessage());
	  }

		Appartiene ap = new Appartiene(vd.getProdottoCodice_prodotto(), gn.getNome_genere());

		sql1 = "INSERT INTO appartiene(videogame_prodotto_codice_prodotto,
genere_nome_genere) VALUES('" + vd.getProdottoCodice_prodotto() + "','" +
				gn.getNome_genere() + "')";

		try {
			state = connessione.createStatement();
			int n = state.executeUpdate(sql1);
			if (n == 1) {
				System.out.println("\n**Associazione tra genere e videogame creata**");
			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

		input.close();
		input_float.close();
	}
```

```
	private static void op3(Connection connessione) throws SQLException {
		//OP3: inserisci una nuova recensione relativa ad un prodotto

		Scanner input = new Scanner (System.in); //oggetto per l'input
		Scanner input_int = new Scanner (System.in); //oggetto per l'input
		Statement state = null; //per costruire le query
		String sql; //stringa contenete il codice

		System.out.println("\nInserisci l'ID del cliente (già esistente): ");
		int ID = input_int.nextInt();
		System.out.println("Inserisci il codice del prodotto (già esistente): ");
		String codice_prodotto = input.nextLine();
		System.out.println("Inserisci un titolo per la recensione: ");
		String titolo = input.nextLine();
		System.out.println("Inserisci una descrizione per la recensione: ");
		String descrizione = input.nextLine();
		System.out.println("Inserisci un punteggio (da 1 a 5) per la recensione: ");
		int punteggio = input_int.nextInt();
		for(;;) {
			if(punteggio <= 0 || punteggio >= 6) {
				System.out.println("Punteggio non valido, inserire di nuovo:");
				punteggio = input_int.nextInt();
			}
			else break;
		}

		Recensione rc = new Recensione(ID, codice_prodotto, titolo, descrizione, punteggio);

		sql = "INSERT INTO recensione(cliente_id, prodotto_codice_prodotto, titolo, descrizione,
punteggio) VALUES('" +
				rc.getClienteID() + "','" + rc.getProdottoCodice_prodotto() + "','" +
rc.getTitolo() + "','" +
				rc.getDescrizione() + "','" + rc.getPunteggio() + "')";


		try {
			state = connessione.createStatement();
			int n = state.executeUpdate(sql);
			if (n == 1) {
				System.out.println("\n**Nuovo prodotto inserito**");
			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

		input.close();
		input_int.close();
	}
```

```
	private static void op4(Connection connessione) throws SQLException {
		//OP4: inserisci un nuovo ordine relativo ad un cliente
		//cliente id, spedizione a null, carrello

		Scanner input = new Scanner (System.in); //oggetto per l'input
		Scanner input_num = new Scanner (System.in);
		Scanner input_2 = new Scanner(System.in);
		Statement state = null; //per costruire le query
		ResultSet rs; //i risultati delle query (quindi dei record)
		String sql_or; //stringa contenete il codice sql dell'ordine
		String sql_car; //stringa contenete il codice sql del carrello
		String sql;
		float costo_tot = 0;

		System.out.println("\nInserisci il codice dell'ordine: ");
		String n_ordine = input.nextLine();
		System.out.println("Inserisci il codice del cliente (già esistente): ");
		int clienteID = input_num.nextInt();

		Ordine or = new Ordine(n_ordine, "IN CORSO", 0, clienteID, null);

		sql_or = "INSERT INTO ordine(nordine, stato, costo_totale, cliente_id) VALUES('" +
				or.getN_ordine() + "','" + or.getStato() + "','" + or.getCosto_totale() + "','" +
or.getClienteID() + "')";

		try {
			state = connessione.createStatement();
			int n = state.executeUpdate(sql_or);
			if (n == 1) {
				System.out.println("**Nuovo ordine inserito**");
			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

		int scelta = 1;
		while(scelta==1) {
			System.out.println("\nInserisci il codice del prodotto: ");
			String codice_prodotto = input_2.nextLine();
			System.out.println("\nInserisci la quantità di questo prodotto: ");
			int quantità = input_num.nextInt();

			Carrello cr = new Carrello(or.getN_ordine(), codice_prodotto, quantità);

			sql_car = "INSERT INTO carrello(ordine_nordine, prodotto_codiceProdotto2,
quantità) VALUES('" +
				cr.getOrdineNordine() + "','" + cr.getProdottoCodice_prodotto() +
"','" + cr.getQuantità() + "')";
```

```
                    try {
                            state = connessione.createStatement();
                            int n = state.executeUpdate(sql_car);
                            if (n == 1) {
                                    System.out.println("**Nuovo carrello inserito**");
                            }
                    } catch (SQLException e) {
                            System.out.println("Errore:" + e.getMessage());
                    }

                    sql = "SELECT costo_unario, codice_prodotto FROM carrello,prodotto,ordine WHERE
ordine_nordine = nordine AND prodotto_codiceProdotto2 = codice_prodotto";

                    try {
                  state = connessione.createStatement();
                  rs = state.executeQuery(sql);

                  while (rs.next() == true) {
                     if(rs.getString("codice_prodotto").equals(cr.getProdottoCodice_prodotto())     &&
or.getN_ordine() == cr.getOrdineNordine()) {
                              costo_tot = costo_tot + (rs.getFloat("costo_unario") * cr.getQuantità());
                              //System.out.println(rs.getFloat("costo_unario"));
                     }
                  }

                } catch (SQLException e) {
                        System.out.println("errore: "+ e.getMessage());
                }


                        System.out.println("\nInserire altri prodotti nel carrello? Sì = 1 | No = 0");
                        scelta = input.nextInt();
                }


                or.setCosto_totale(costo_tot);



                sql_or = "UPDATE ordine SET costo_totale='" + or.getCosto_totale() + "'WHERE nordine='" +
or.getN_ordine() + "'";

                try {
                        state = connessione.createStatement();
                        int n = state.executeUpdate(sql_or);
                        if (n == 1) {
                                System.out.println("**Ordine aggiornato con il costo totale**");
```

```
                }
        } catch (SQLException e) {
                System.out.println("Errore:" + e.getMessage());
        }

        input.close();
        input_num.close();
        input_2.close();
}
```

```
private static void op5(Connection connessione) throws SQLException {
        //OP5: stampa il numero totale di ordini effettuati da tutti i clienti

        Statement state = null; //per costruire le query
        String sql = "SELECT * FROM cliente"; //stringa contenete il codice
        ResultSet rs; //i risultati delle query (quindi dei record)

        try {
                state = connessione.createStatement();
                rs = state.executeQuery(sql);
                System.out.println("ID     ORDINI EFFETTUATI");
                while(rs.next() == true) {
                        System.out.println(rs.getString("ID") + "\t" + rs.getString("nordini"));
                }
        } catch (SQLException e) {
                System.out.println("Errore:" + e.getMessage());
        }
}
```

```
	private static void op6(Connection connessione) throws SQLException {
		//OP6: stampa il costo totale di un ordine effettuato da un cliente

		Scanner input = new Scanner (System.in); //oggetto per l'input
		Statement state = null; //per costruire le query
		String sql; //stringa contenete il codice
		ResultSet rs; //i risultati delle query (quindi dei record)


		System.out.println("\nInserisci n°ordine (già esistente): ");
		String n_ordine = input.nextLine();


		sql = "SELECT * FROM ordine WHERE nordine='" + n_ordine + "'";

		try {
			state = connessione.createStatement();
			rs = state.executeQuery(sql);
			while(rs.next() == true) {
				System.out.println("NORDINE  COSTO TOTALE");
				System.out.println(rs.getString("nordine") + "\t" +
rs.getString("costo_totale"));
			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

		input.close();
	}
```

```
	private static void op7(Connection connessione) throws SQLException {
		//OP7: stampa tutte le spedizioni eseguite sull'e-commerce


		Statement state = null; //per costruire le query
		String sql; //stringa contenete il codice
		ResultSet rs; //i risultati delle query (quindi dei record)


		sql = "SELECT * FROM spedizione";

		try {
			state = connessione.createStatement();
			rs = state.executeQuery(sql);
			while(rs.next() == true) {
				System.out.println("\nNSPEDIZIONE    ID CLIENTE");
				System.out.println(rs.getString("nspedizione")    +    "                    "    +
rs.getString("cliente1_ID"));
			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

	}
```

```
	private static void op8(Connection connessione) throws SQLException {
		//OP8: stampa il numero di videogiochi per ogni genere.


		Statement state = null; //per costruire le query
		String sql; //stringa contenete il codice
		ResultSet rs; //i risultati delle query (quindi dei record)

		sql = "SELECT genere_nome_genere, COUNT(*) as NumVideogame FROM appartiene GROUP BY genere_nome_genere";

		try {
			state = connessione.createStatement();
			rs = state.executeQuery(sql);
			System.out.println("\nGENERE		NUMVIDEOGAME");
			while(rs.next() == true) {
				if(rs.getString("genere_nome_genere").length() >= 7) {
					System.out.println(rs.getString("genere_nome_genere") + "\t" + rs.getString("NumVideogame"));
				} else {
					System.out.println(rs.getString("genere_nome_genere") + "\t\t" + rs.getString("NumVideogame"));
				}

			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

	}
```

```
	private static void op9(Connection connessione)  throws SQLException {
		//OP9: stampa il costo totale medio per i clienti che hanno effettuato almeno un ordine


		Statement state = null; //per costruire le query
		String sql; //stringa contenete il codice
		ResultSet rs; //i risultati delle query (quindi dei record)

		sql = "SELECT cliente_id, avg(costo_totale) AS costo_totale_medio, nordini FROM ordine, cliente WHERE cliente_id = ID GROUP BY cliente_id";

		try {
			state = connessione.createStatement();
			rs = state.executeQuery(sql);
			System.out.println("\nID	COSTO MEDIO	NORDINI");
			while(rs.next() == true) {
				System.out.println(rs.getInt("cliente_id") + "\t" + rs.getFloat("costo_totale_medio") + "\t\t" + rs.getInt("nordini"));
			}
		} catch (SQLException e) {
			System.out.println("Errore:" + e.getMessage());
		}

	}

}
```

7 TEST OPERAZIONI APPLICAZIONE JAVA

Qui di seguito sono riportati dei test sull'applicazione Java che implementa le op. descritte al punto 3.

7.1 OPERAZIONE 1

Operazione 1: aggiungi un nuovo cliente.
Un nuovo cliente si iscrive al sito di e-commerce, inserendo i propri dati anagrafici.

```
Inserisci ID cliente (numerico MAX 5 caratteri):
46824
Inserisci nome cliente:
Nina
Inserisci cognome cliente:
Peluso
Inserisci età cliente:
24

**Nuovo cliente inserito**
```

7.2 OPERAZIONE 2

Operazione 2: aggiungi un nuovo videogame.
Viene inserito un nuovo videogame all'interno del sito di e-commerce, specificando solo un genere a cui appartiene.

```
Inserisci codice prodotto:
fjwoif2
Inserisci nome prodotto:
Tomb Raider
Inserisci descrizione_prodotto:
Un gioco di avventura boh
Inserisci costo unario prodotto:
29,99
Inserisci formato prodotto (Fisico o Digitale):
Fisico
Inserisci piattaforma prodotto:
PS4

**Nuovo prodotto inserito**

**Nuovo videgioco associato**
Inserisci il genere del prodotto (già esistente):
Avventura

**Associazione tra genere e videogame creata**
```

7.3 Operazione 3

Operazione 3: aggiungi una nuova recensione relativa ad un prodotto.
Un cliente scrive una nuova recensione che fa riferimento ad un determinato prodotto dando anche un voto compreso tra uno e cinque.

```
Inserisci l'ID del cliente:
1
Inserisci il codice del prodotto:
HFDU8
Inserisci un titolo per la recensione:
Bello
Inserisci una descrizione per la recensione:
Un bel gioco
Inserisci un punteggio (da 1 a 5) per la recensione:
5

**Nuovo prodotto inserito**
```

7.4 Operazione 4

Operazione 4: aggiunge un nuovo ordine relativo ad un cliente.
Un cliente (già iscritto al sito) effettua un nuovo ordine.

```
Inserisci il codice dell'ordine:
YDHKSDM23
Inserisci il codice del cliente (già esistente):
1
**Nuovo ordine inserito**

Inserisci il codice del prodotto:
HDISK32

Inserisci la quantità di questo prodotto:
1
**Nuovo carrello inserito**

Inserire altri prodotti nel carrello? Sì = 1 | No = 0
1

Inserisci il codice del prodotto:
KJL569

Inserisci la quantità di questo prodotto:
1
**Nuovo carrello inserito**

Inserire altri prodotti nel carrello? Sì = 1 | No = 0
0
**Ordine aggiornato con il costo totale**
```

7.5 Operazione 5

Operazione 5: stampa il numero totale di ordini effettuati da tutti i clienti.
Vogliamo conoscere quanti ordini sono stati effettuati dai clienti iscritti all'e-commerce (per sapere quanti ordini vengono effettuati al giorno).

```
ID      ORDINI EFFETTUATI
00001   4
12356   0
20234   0
20467   0
23754   0
31357   0
34238   0
34295   0
39742   0
42946   0
45690   0
45693   0
45908   0
46824   0
54895   0
54985   5
56395   0
67240   0
67690   3
74938   1
85274   0
85327   0
88348   2
96736   0
```

7.6 Operazione 6

Operazione 6: stampa il costo totale di un ordine effettuato da un cliente.
Si vuole stampare il costo totale di un ordine.

```
Inserisci n°ordine (già esistente):
FER35Y
NORDINE  COSTO TOTALE
FER35Y  275.98
```

7.7 Operazione 7

Operazione 7: stampa tutte le spedizioni eseguite sull'e-commerce.
Vengono stampate tutte le spedizioni che sono state eseguite dal sito.

```
NSPEDIZIONE    ID CLIENTE
DEFEI39        54985

NSPEDIZIONE    ID CLIENTE
DWU782        67690

NSPEDIZIONE    ID CLIENTE
FERHF48        1

NSPEDIZIONE    ID CLIENTE
KTO4I03        74938

NSPEDIZIONE    ID CLIENTE
UREI30G        88348
```

7.8 Operazione 8

Operazione 8: stampa il numero di videogame che sono presenti nel database, raggruppandoli in base al loro genere.

Vengono stampati i nomi dei generi presenti nel DB con il numero di videogiochi presenti per quel determinato genere.

GENERE	NUMVIDEOGAME
Avventura	4
Azione	4
GDR	1
MMO	1
MMOPRG	1
Picchiaduro	2
Plattform	1
Simulazione	1

7.9 Operazione 9

Operazione 9: per ogni cliente che ha effettuato almeno un ordine, si vuole stampare il loro numero di ordini effettuati e, in più, il costo totale medio di tutti i loro ordini.

Vengono stampati ID cliente, numero ordini e costo totale medio.

ID	COSTO MEDIO	NORDINI
1	42.469997	2
54985	175.0	1
67690	41.48	1
74938	29.98	1
88348	275.98	1