

Society6

introduzione	5
Descrizione del DataBase	5
Descrizione delle Entità	6
Descrizione delle Relazioni (normalizzato)	7
Progettazione Concettuale	8
Livello Fisico	13
Creazione del DataBase	13
Tabella 1: utenti	14
Tabella 2: post	15
Tabella 3: piace_post	16
Tabella 4: segue	17
Tabella 5: art	18
Tabella 6: piace_art	19
Tabella 6: tag	19
Tabella 7: categorieArt	20
Tabella 8: artInCat	20
Tabella 9: commenti	21
Tabella 10: collection	22
Tabella 11: art_collection	22
Tabella 12: categorieOggetto	23
Tabella 13: oggetto	23
Tabella 14: colore	24
Tabella 15: oggetto_colore	24
Tabella 16: prodotto	25
Tabella 17: colore_abilitato	25
Tabella 18: taglia	26
Tabella 19: taglia_oggetto	26
Tabella 20: carrello	27
Tabella 21: compravendite	28
Riepilogo Tabelle e Attributi	29
Trigger	31
Trigger 1 trig_segue:	31
Trigger 3 trig_compravendite:	32
Trigger 4 trig_aggiungiProd :	33
Trigger 5 trig_aggiungiColore :Questo trigger si occupa di abilitare tutti i colori su un prodotto quanto quest'ultimo viene inserito	34
Trigger 6 trig_aggiungiColore :	34

Trigger 7 trig_artDel:	35
Trigger 8 trig_prodottoDel:	35
Trigger 9 trig_postDel:	36
Inserimento dei Dati	36
Metotdo di inserimento	37
Inserimenti tabella utenti:	37
Inserimento tabella art:	38
Inserimento tabella piace_art:	39
Inserimento tabella post:	39
Inserimento tabella piace_post:	40
Inserimento tabella tag:	41
Inserimento tabella segue:	42
Inserimento tabella collection	42
Inserimento tabella artInCat	43
Inserimento tabella categorieArt	43
Inserimento tabella collection	44
Inserimento tabella art_collection	44
Inserimento tabella categorieOggetto	45
Inserimento tabella oggetto	45
Inserimento tabella taglia	46
Inserimento tabella colore	46
Inserimento tabella oggetto_colore	46
Inserimento tabella carrello	47
Inserimento tabella compravendite	48
Inserimento tabella taglia	48
Numero di Record per Tabella	49
Query	50
Query 1:	50
Query 2:	51
Query 3:	52
Query 4:	53
Query 5:	54
Query 7:	56
Query 8:	56
Query 8:	57
Query 9:	57
Query 10:	59
Query 11:	60
Query 12:	61
Query 13:	62
Query 14:	63

Query 15:	64
Query 15:	65
Query 16:	66
Query 17:	66
Query 18:	66
Query 19:	67
Query 20:	67
Query 21:	68
Query 22:	69
Interrogazione Algebra Relazionale	71
Algebra Relazionale	71
Query Algebra Relazionale	71
Query1:	71
Query2:	71
MongoDB	73
Creazione DataBase Collection	73
Query MongoDB	74
Query1:	75
Query2:	75

introduzione

Il database presentato è stato progettato prendendo ispirazione dal reale database utilizzato da Society6 una azienda Americana che mette a disposizione uno store dove vende prodotti come magliette quadri asciugamani ..ecc con disegni di artisti registrati sulla piattaforma.

Lo scopo è quello di tenere tenere traccia dei disegni dell'artista e su quali oggetti è stato abilitato.

Si pone quindi particolare attenzione sulle possibilità dell'utente (artista) all'interno della piattaforma, come seguire altri utenti, comprare prodotti con art di altri utenti su determinati prodotti mettere mi piace ai disegni ...ecc

Descrizione del DataBase

Il database è stato progettato in modo tale da gestire:

- I follow di ciascuno utente
- le art di ciascuno utente
- I mi piace di ciascuno utente sulle art o sui post
- Le categorie degli oggetti e delle art
- La possibilità dell'utente di decidere su quali oggetti e con quali colori vendere a propria art
- la possibilità dell'utente creare collection con dentro art da lui scelte

Descrizione delle Entità

Utente: può essere sia un artista che pubblica (disegni) sia un compratore è colui che dà vita alla piattaforma seguendo altri utenti mettendo like creando collection pubblicando art e post

Art: identifica il disegno pubblicato dall'utente

Tag: sono collegati alle art e rappresentano le chiavi di ricerca delle art all'interno di Society6 sono inseriti dall'utente quando pubblica un'art

Categoria Art: rappresenta un insieme di art che appartengono a una Categoria è l'utente a scegliere in quali categorie inserire la propria art

Collection: rappresenta un insieme di art selezionate da un utente (l'utente può inserire anche Art non di sua proprietà)

Post: identifica un testo che l'utente pubblica

Oggetto: identifica la maglietta, il quadro... o qualsiasi altro oggetto che potrà essere venduto

Colore:identifica il colore dell'oggetto

Taglia: identifica la taglia dell'oggetto (grande ,picolo,L,M, XL)

Categoria Oggetto: identifica la categoria dell'oggetto (un singolo oggetto si può trovare in una sola categoria)

Prodotto: Identifica il connubio tra art e oggetto andando a comporre il Prodotto che sara poi effettivamente messo in vendita è l'utente che decide quale oggetto abilitare per la vendita

Descrizione delle Relazioni (normalizzato)

Commenta (art): ogni utente può commentare le art questa relazione si occupa di tenere traccia dei commenti

Commenta (post) : ogni utente può commentare i post questa relazione si occupadi tenerne traccia

Piace (art): ogni utente può mettere mi piace alle art quando lo fa viene tenuto traccia su questa relazione

Piace (Post): ogni utente può mettere mi piace ai post quando lo fa viene tenuto traccia su questa relazione

SiTrova (categoria): ogni utente può selezionare in quali categorie mettere la propria art in questa relazione tiene traccia di questo

Contiene (Collection): ogni utente può inserire le art in delle collection questa relazione si occupa di tenerne traccia

Possiede (Colore): questa relazione tiene traccia dei possibili colori disponibili rispetto a uno oggetto

Possiede (Taglia): questa relazione tiene traccia delle possibili taglie disponibili rispetto a uno oggetto e del prezzo dell'oggetto con una determinata taglia

Carrello : questa relazione si occupa di tenere traccia del carrello degli utenti che inseriscono il prodotto con un colore e una taglia nel proprio carrello

Compravendite: questa relazione si occupa di tenere di chi ha comprato, di chi ha venduto rispetto al prodotto con taglia e un colore

Colore Abilitato(prodotto): questa relazione si occupa di tenere traccia delle preferenze dell'utente proprietario(che ha fatto l'art e che la ha abilitatata su quell'oggetto) del prodotto di rendere abilitati determinati colori

Progettazione Concettuale

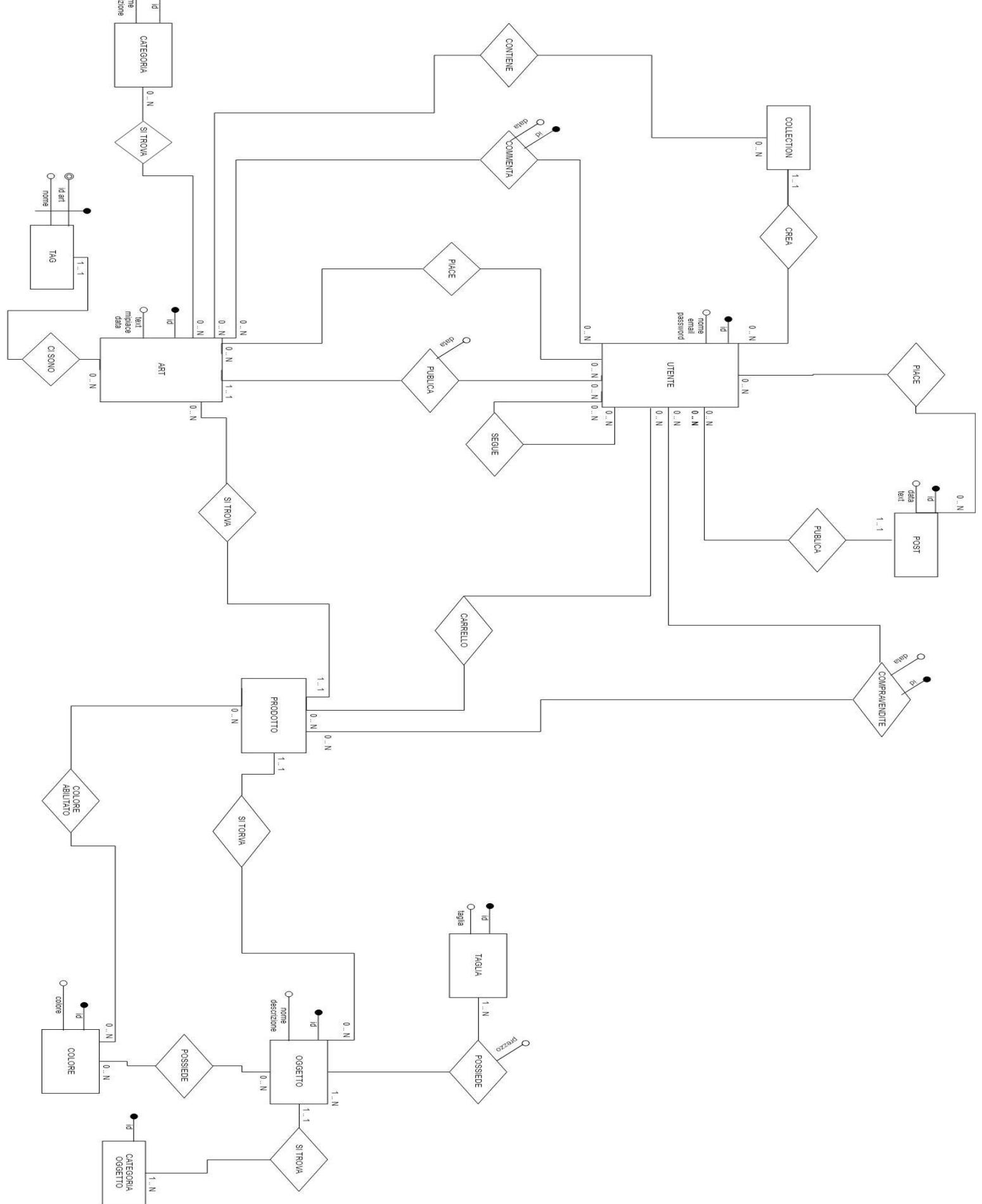
Il modello entità/relazione (E/R), è uno strumento per analizzare le caratteristiche di una realtà in modo indipendente dagli eventi che in essa accadono, cioè per costruire un modello concettuale dei dati indipendenti dalle applicazioni. Il risultato di questo lavoro è la definizione di una rappresentazione grafica, detta Schema E/R, che mette in evidenza gli aspetti fondamentali del modello concettuale, con i dati caratterizzati e le associazioni tra essi. Il modello descrive lo schema concettuale di un problema o di una gestione aziendale e non si occupa dell'efficienza delle operazioni di manipolazione e ritrovamento dei dati sugli archivi fisici. Gli elementi di un modello Entity/Relationship sono:

Entità: è un oggetto (concreto o astratto) che ha un significato anche quando viene considerato in modo isolato ed è di interesse per la realtà che si vuole modellare.

Relazione: è un legame che stabilisce un'interazione tra le entità.

Attributi: le proprietà delle entità e delle associazioni sono descritte attraverso gli attributi.

Diagramma E/R



Progettazione Logica

L'obiettivo della progettazione logica è quello di costruire uno schema logico in grado di descrivere, in maniera corretta ed efficiente, tutte le informazioni contenute nello schema E/R prodotto nella fase di progettazione concettuale.

La semplificazione dello schema si rende necessaria perché non tutti i costrutti del modello E/R hanno una traduzione naturale nei modelli logici, ad esempio le generalizzazioni.

Utilizzeremo anche la normalizzazione che ci permetterà la ristrutturazione dello schema in modo tale da definire lo schema fisico che sarà la base per la costruzione del nostro database.

Normalizzazione

Una relazione è un legame che stabilisce un'interazione tra le entità.

La cardinalità di un verso della relazione è la caratteristica che indica quante istanze dell'entità di arrivo si associano all'istanza dell'entità di partenza.

La cardinalità può essere a uno oppure a molti e pertanto le relazioni tra due entità si classificano in:

- 1 : 1

Ad ogni elemento del primo insieme corrisponde un unico elemento del secondo insieme e viceversa.

- 1 : N

Ad un elemento del primo insieme possono corrispondere più elementi del secondo, mentre ad ogni elemento del secondo insieme deve corrispondere un unico elemento del primo.

- N : M

Ad ogni elemento del primo insieme possono corrispondere più elementi del secondo insieme e viceversa.

Le Tre Forme Normali

- Prima Forma Normale: Una relazione è in prima forma normale (1FN) quando rispetta i requisiti fondamentali del modello relazionale, cioè:
 - Tutte le righe della tabella contengono lo stesso numero di colonne;
 - Gli attributi rappresentano informazioni elementari;
 - I valori che compaiono in una colonna sono dello stesso tipo, cioè appartengono allo stesso dominio;
 - Ogni riga è diversa da tutte le altre, cioè non ci possono essere due righe con gli stessi valori nelle colonne;
 - L'ordine con il quale le righe compaiono nella tabella è irrilevante.
- Seconda Forma Normale: Una relazione è in seconda forma normale (2FN) quando è in prima forma normale e tutti i suoi attributi non-chiave dipendono dall'intera chiave. Cioè non possiede attributi che dipendono soltanto da una parte della chiave.
- Terza Forma Normale: Una relazione è in terza forma normale (3FN) quando è in seconda forma normale e tutti gli attributi non-chiave dipendono direttamente dalla chiave. Cioè non possiede attributi non-chiave che dipendono da altri attributi non-chiave.

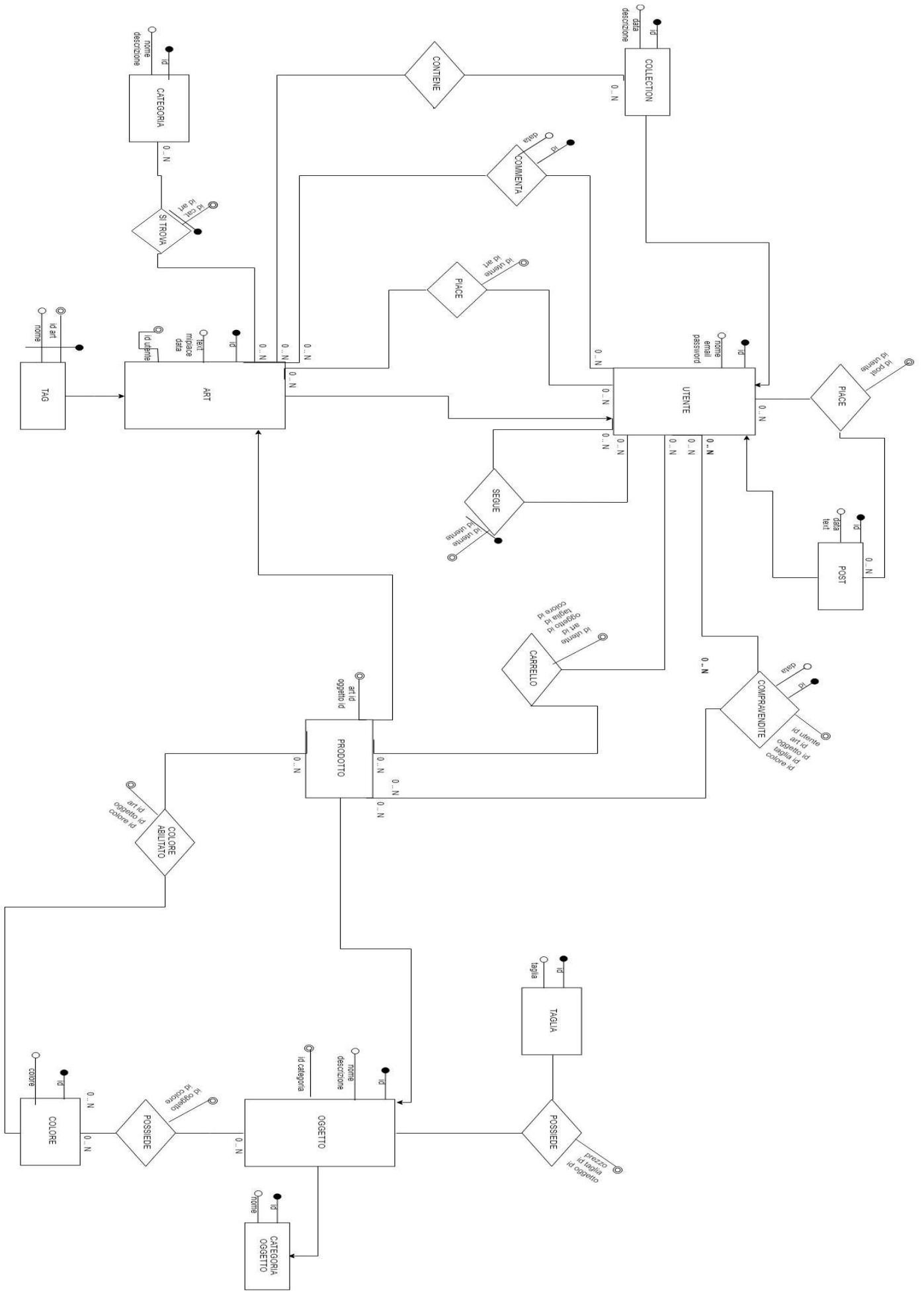
Queste forme normali, insieme ai processi effettuati in precedenza rappresentano il significato della normalizzazione.

Si deve osservare che la normalizzazione diminuisce la ridondanza dei dati e la possibilità di inconsistenza, ma rende più complesse le operazioni di ritrovamento dei dati.

La normalizzazione è importante nel modello di un database perché l'integrità e la consistenza dei dati sono prioritarie rispetto alla velocità di ritrovamento dei dati, che rimane comunque un fattore essenziale.

Diagramma E/R normalizzato

nel diagramma E/R normalizzato sono state aggiunte alcune foreign key che per motivi di leggibilità dello schema non sono state collegate con le rispettive relazione mi riferisco alle foreign key dell'entità COMPRAVENDITE e dell'entità CARRELLO.



Progettazione Fisica

Livello Fisico

Il livello fisico rappresenta l'effettiva installazione degli archivi elettronici, esso indica l'ubicazione dei dati nelle memorie di massa.

Il livello fisico è quindi l'implementazione del livello logico sui supporti per la registrazione fisica dei dati.

In questo capitolo svilupperemo il database tramite MySQL usato come Relationale DataBase Management System (RDMS).

Creazione del DataBase

Adesso quindi creiamo la nostra base di dati con il nome che di society6

Utilizziamo quindi i seguenti comandi.

```
1  create database society6;
2  use society6
3  |
```

Adesso abbiamo il pieno controllo del database e possiamo eseguire tutte le operazioni volute, come la creazione delle tabelle.

Creazione Tabelle

Adesso andremo a creare la struttura del nostro database, implementando le tabelle attraverso i comandi che consentono di farlo.

Quando si crea una nuova tabella è possibile specificare quale motore di archiviazione usare aggiungendo l'opzione ENGINE al comando CREATE TABLE.

Se questa opzione viene omessa, allora viene utilizzato il metodo di archiviazione di default che per MySQL 5.7 è InnoDB

Tabella 1: utenti

```
4  create table utenti (
5      id bigint auto_increment NOT NULL,
6      nome varchar(30) NOT NULL UNIQUE,
7      email varchar(50) NOT NULL UNIQUE,
8      password varchar(100) not null,
9      data TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
10     primary key(id)
11 );
```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
nome	varchar(30)	NO	UNI	NULL	
email	varchar(50)	NO	UNI	NULL	
password	varchar(100)	NO		NULL	
data	timestamp	NO		CURRENT_TIMESTAMP	

5 rows in set (0.01 sec)

Nella tabella Utenti ci sono 5 campi di cui l'id è primary key ossia identifica il record specifico quindi non esistono due utenti con lo stesso id inoltre è autoincrement il che significa che ad ogni inserimento verrà aumentato;

nome ed email sono UNIQUE il che significa che non esistono due utenti che hanno lo stesso nome o con la stessa email ma possono esistere benissimo due email con la stessa password, il campo data è di tipo timestamp e identifica la data di registrazione dell'utente

Tabella 2: post

```
14  create table post(
15      id bigint auto_increment NOT NULL,
16      testo varchar(500) not null,
17      titolo varchar(50) not null,
18      utenteID bigint not null,
19      data TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
20      primary key(id),
21      foreign key (utenteID) references utenti(id)
22  );
```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
testo	varchar(500)	NO		NULL	
titolo	varchar(50)	NO		NULL	
utenteID	bigint(20)	NO	MUL	NULL	
data	timestamp	NO		CURRENT_TIMESTAMP	

5 rows in set (0.00 sec)

Nella tabella post ci sono 5 campi di cui uno primary key (id) inoltre c'è una foreign key utenteID che punta alla primary key (id) della tabella utenti in modo tale da associare ogni post ad un singolo utente

Tabella 3: piace_post

```
24  create table piace_post(
25      utenteID bigint not null,
26      postID bigint not null,
27      foreign key (utenteID) references utenti(id),
28      foreign key (postID) references post(id),
29      primary key(postID,utenteID)
30  );|
```

Field	Type	Null	Key	Default	Extra
utenteID	bigint(20)	NO	PRI	NULL	
postID	bigint(20)	NO	PRI	NULL	

2 rows in set (0.00 sec)

Questa tabella rappresenta una relazione molti a molti (Piace (post))
nello specifico c'è una primary key composta da due utenteID e postID il che significa che un
utente può mettere mi piace solo una volta ad un post inoltre utenteID e postID sono delle
foreign key che puntano rispettivamente a alla tabella utenti e post

Tabella 4: segue

```
33  --utente1 segue utente2
34  create table segue(
35      utente1 bigint not null,
36      utente2 bigint not null,
37      foreign key (utente1) references utenti(id),
38      foreign key (utente2) references utenti(id),
39      primary key(utente1,utente2)
40  --not working  CONSTRAINT ck_segue CHECK (utente1 != utente2)
41  );
```

Field	Type	Null	Key	Default	Extra
utente1	bigint(20)	NO	PRI	NULL	
utente2	bigint(20)	NO	PRI	NULL	

2 rows in set (0.00 sec)

Questa tabella rappresenta la relazione segue, come la tabella precedente ha una primary key composta e due foreign key con la differenza che essendo la relazione sulla stessa entità, le 2 foreign key puntano alla stessa tabella (utenti) ; questa relazione si legge: utente1 segue utente2 , il constraint mostrato viene interpretato ma non eseguito da mysql quindi il constraint viene implementato attraverso l'uso di un trigger che controlla che l'utente non segua se stesso

Tabella 5: art

```
43  create table art(
44      id bigint auto_increment NOT NULL,
45      titolo varchar(50) not null,
46      testo varchar(500) not null,
47      utenteID bigint not null,
48      data TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
49      primary key(id),
50      foreign key (utenteID) references utenti(id)
51  );
```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
titolo	varchar(50)	NO	MUL	NULL	
testo	varchar(500)	NO		NULL	
utenteID	bigint(20)	NO	MUL	NULL	
data	timestamp	NO		CURRENT_TIMESTAMP	

5 rows in set (0.00 sec)

In questa tabella ci sono tutte le art (disegni) degli utenti la primary key è l'id ed è auto_increment inoltre ci possono essere art con lo stesso titolo per questo titolo non è unique per identificare l'utente che ha pubblicato l'art c'è la foreign key utenteID che punta alla primary key della tabella utenti

Tabella 6: piace_art

```
53  create table piace_art(
54    |   utenteID bigint not null,
55    |   artID bigint not null,
56    |   foreign key (utenteID) references utenti(id),
57    |   foreign key (artID) references art(id),
58    |   primary key(artID,utenteID)
59 );
```

Field	Type	Null	Key	Default	Extra
utenteID	bigint(20)	NO	PRI	NULL	
artID	bigint(20)	NO	PRI	NULL	

2 rows in set (0.00 sec)

questa tabella è simile a piace_post ma è riferita alle art

Tabella 6: tag

```
61  create table tag(
62    |   artID bigint not null,
63    |   nome varchar(10) not null,
64    |   foreign key (artID) references art(id),
65    |   primary key(artID,nome)
66 );
```

Field	Type	Null	Key	Default	Extra
artID	bigint(20)	NO	PRI	NULL	
nome	varchar(10)	NO	PRI	NULL	

2 rows in set (0.01 sec)

Questa tabella rappresenta i tag delle art ogni tag è riferito a un'art grazie alla foreign key artID che insieme all'attributo nome compone la primary key della tabella

Tabella 7: categorieArt

```
68  create table categorieArt(
69    id int auto_increment,
70    nome varchar(20) not null,
71    descrizione varchar(200) not null,
72    primary key(id)
73 );
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nome	varchar(20)	NO		NULL	
descrizione	varchar(1000)	NO		NULL	

3 rows in set (0.00 sec)

Questa tabella rappresenta l'entità Categoria Art

Tabella 8: artInCat

```
75  create table artInCat(
76    categoriaID int not null,
77    artID bigint,
78    primary key(categoriaID,artID),
79    foreign key (artID) references art(id),
80    foreign key (categoriaID) references categorieArt(id)
81 );
```

Field	Type	Null	Key	Default	Extra
categoriaID	int(11)	NO	PRI	NULL	
artID	bigint(20)	NO	PRI	NULL	

2 rows in set (0.00 sec)

Questa tabella rappresenta la relazione SiTrova (categoria) ossia quale art si trova in quale categoria si noti che essendo la relazione molti a molti un'art può stare in più categorie

Tabella 9: commenti

```
83  create table commenti(
84      id int auto_increment,
85      commento varchar(500) not null,
86      data TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
87      utenteID bigint not null,
88      artID bigint not null,
89      foreign key (artID) references art(id),
90      foreign key (utenteID) references utenti(id),
91      primary key(id)
92 );
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
commento	varchar(500)	NO		NULL	
data	timestamp	NO		CURRENT_TIMESTAMP	
utenteID	bigint(20)	NO	MUL	NULL	
artID	bigint(20)	NO	MUL	NULL	

5 rows in set (0.00 sec)

Questa tabella rappresenta la relazione commenta, in questo caso le due foreign key utenteID e artID non sono primary key (a differenza di piace_art) in quanto un utente può commentare più volte la stessa art di conseguenza ho aggiunto una primary key id auto_increment

Tabella 10: collection

```

94  create table collection(
95      id bigint auto_increment,
96      titolo varchar(30) not null,
97      utenteID bigint not null,
98      descrizione varchar(20) not null,
99      data TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
100     foreign key (utenteID) references utenti(id),
101     primary key(id),
102     unique(utenteID,titolo)
103 );

```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
titolo	varchar(30)	NO	MUL	NULL	
utenteID	bigint(20)	NO	MUL	NULL	
descrizione	varchar(500)	NO		NULL	
data	timestamp	NO		CURRENT_TIMESTAMP	

Questa tabella è simile a categorieArt con la differenza che una collection viene creata da uno utente quindi vi è una foreign key alla tabella utenti inoltre un utente non può creare una collection con stesso titolo di un'altra che aveva già creato ma utenti diversi possono avere collection con lo stesso nome questo grazie a unique tra utenteID e titolo in

Tabella 11: art_collection

```

106  create table art_collection(
107      collectionID bigint not null,
108      artID bigint not null,
109      foreign key (artID) references art(id),
110      foreign key (collectionID) references collection(id),
111      primary key(collectionID,artID)
112 );

```

Field	Type	Null	Key	Default	Extra
collectionID	bigint(20)	NO	PRI	NULL	
artID	bigint(20)	NO	PRI	NULL	

Questa tabella rappresenta la relazione Contiene (Collection), rappresenta la presenza di un'art in una collection è simile alla tabella artInCat

Tabella 12: categorieOggetto

```
116  create table categorieOggetto(
117      id int auto_increment,
118      nome varchar(20) not null,
119      descrizione varchar(200) not null,
120      primary key(id)
121 );
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nome	varchar(20)	NO		NULL	
descrizione	varchar(200)	NO		NULL	

Questa tabella rappresenta l'entità Categoria Oggetto ha un id un nome e una descrizione

Tabella 13: oggetto

```
123  create table oggetto(
124      id int auto_increment,
125      nome varchar(20) not null,
126      descrizione varchar(200) not null,
127      categoriaID int not null,
128      foreign key (categoriaID) references categorieOggetto(id),
129      primary key (id)
130 );
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
nome	varchar(20)	NO		NULL	
descrizione	varchar(200)	NO		NULL	
categoriaID	int(11)	NO	MUL	NULL	

Questa tabella contiene tutti gli oggetti si noti come ogni oggetto può appartenere una singola categoria a differenza delle art questo viene implementato attraverso la foreign key categoriaID che punta alla tabella categorieOggetto

Tabella 14: colore

```
135  create table colore(
136    id int auto_increment,
137    colore varchar(20) not null,
138    primary key(id)
139 );
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
colore	varchar(20)	NO		NULL	

Si noti che in questa tabella ci sarà un record unico che sarà poi abilitato sugli oggetti che non hanno un colore

Tabella 15: oggetto_colore

```
141  create table oggetto_colore(
142    coloreID int not null,
143    oggettoID int not null,
144    foreign key (oggettoID) references oggetto(id),
145    foreign key (coloreID) references colore(id),
146    primary key(coloreID,oggettoID)
147 );
```

Field	Type	Null	Key	Default	Extra
coloreID	int(11)	NO	PRI	NULL	
oggettoID	int(11)	NO	PRI	NULL	

questa tabella si riferisce alla relazione Possiede (Colore) e tiene traccia dei colori disponibili degli oggetti

Tabella 16: prodotto

```
149  create table prodotto(
150      oggettoID int not null,
151      artID bigint not null,
152      foreign key (oggettoID) references oggetto(id),
153      foreign key (artID) references art(id),
154      primary key(oggettoID,artID)
155 );
```

Field	Type	Null	Key	Default	Extra
oggettoID	int(11)	NO	PRI	NULL	
artID	bigint(20)	NO	PRI	NULL	

Questa tabella va a formare il prodotto venduto è l'utente a decidere quale oggetto abilitare su quale art

Tabella 17: colore_abilitato

```
157  create table colore_abilitato(
158      oggettoID int not null,
159      artID bigint not null,
160      coloreID int not null,
161      foreign key (oggettoID) references oggetto(id),
162      foreign key (artID) references art(id),
163      foreign key (coloreID) references colore(id),
164      primary key(oggettoID,artID,coloreID)
165 );
```

Field	Type	Null	Key	Default	Extra
oggettoID	int(11)	NO	PRI	NULL	
artID	bigint(20)	NO	PRI	NULL	
coloreID	int(11)	NO	PRI	NULL	

Questa tabella si riferisce alla relazione Colore Abilitato(prodotto) e tiene traccia delle preferenze dell'utente rispetto a quali colori rendere disponibili alla vendita qua è stata usata una chiave primaria composta da tre attributi che sono anche foreign key

Tabella 18: taglia

```
169  create table taglia(
170      id int auto_increment,
171      taglia varchar(20) not null,
172      primary key(id)
173 );
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
taglia	varchar(20)	NO		NULL	

Questa tabella tiene traccia di tutte le taglie

Tabella 19: taglia_oggetto

```
178  create table taglia_oggetto(
179      tagliaID int not null,
180      oggettoID int not null,
181      prezzo int not null,
182      primary key(oggettoID,tagliaID),
183      foreign key(tagliaID) references taglia(id),
184      foreign key (oggettoID) references oggetto(id)
185 );
```

Field	Type	Null	Key	Default	Extra
tagliaID	int(11)	NO	PRI	NULL	
oggettoID	int(11)	NO	PRI	NULL	
prezzo	int(11)	NO		NULL	

Questa tabella rappresenta la relazione Possiede (Taglia)

Tabella 20: carrello

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
utenteID	bigint(20)	NO	MUL	NULL	
artID	bigint(20)	NO	MUL	NULL	
oggettoID	int(11)	NO	MUL	NULL	
tagliaID	int(11)	NO		NULL	
coloreID	int(11)	NO	MUL	NULL	

Questa tabella è molto interessante rappresenta il carrello dei vari utenti quindi ci sarà il prodotto composto come una foreign key composta (artID,oggettoID) che punta a prodotto inoltre ci sono altre 2 foreign key composte come tagliaID , oggettoID che puntano a taglia_oggetto e coloreID,oggettoID ,artID che rispettivamente a taglia oggetto e a colore_abilitato garantendo così che un utente possa inserire un prodotto esistente con un colore che è stato abilitato e una taglia disponibile su quello oggetto

Tabella 21: compravendite

```

219  create table compravendite(
220      id bigint auto_increment not null,
221      compratore bigint not null,
222      venditore bigint not null,
223      artID bigint not null,
224      oggettoID int not null,
225      tagliaID int not null,
226      coloreID int not null,
227      primary key(id)
228  );

```

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	NULL	auto_increment
compratore	bigint(20)	NO		NULL	
venditore	bigint(20)	NO		NULL	
artID	bigint(20)	NO		NULL	
oggettoID	int(11)	NO		NULL	
tagliaID	int(11)	NO		NULL	
coloreID	int(11)	NO		NULL	

Questa tabella rappresenta le vendite (e le compere) all'interno di society6 come struttura è uguale alla tabella carrello, con l'aggiunta del venditore ma c'è un'altra differenza fondamentale, in questa tabella non ci sono foreign key, questa scelta è stata fatta per un motivo: nel caso un utente si cancelli o che viene cancellata un'art le vendite o le compere di quell'utente/art devono rimanere in questa tabella per ragioni di contabilità si noti che l'attributo prezzo non è stato aggiunto in quanto si assume rimanga sempre uguale e non cambi.

quindi in questa tabella si vuole consistenza dei dati nell'inserimento ma non nella cancellazione cosa che con le foreign key non è possibile quindi è stata implementata attraverso l'uso dei trigger

Riepilogo Tabelle e Attributi

```
select TABLE_NAME,COLUMN_NAME,COLUMN_TYPE,COLUMN_KEY,EXTRA from
information_schema.columns where table_schema = 'society6'
```

TABLE_NAME	COLUMN_NAME	COLUMN_TYPE	COLUMN_KEY	EXTRA
art	id	bigint(20)	PRI	auto_increment
art	titolo	varchar(50)		
art	testo	varchar(500)		
art	utenteID	bigint(20)	MUL	
art	data	timestamp		
artInCat	categoriaID	int(11)	PRI	
artInCat	artID	bigint(20)	PRI	
art_collection	collectionID	bigint(20)	PRI	
art_collection	artID	bigint(20)	PRI	
carrello	id	bigint(20)	PRI	auto_increment
carrello	utenteID	bigint(20)	MUL	
carrello	artID	bigint(20)	MUL	
carrello	oggettoID	int(11)	MUL	
carrello	tagliaID	int(11)		
carrello	coloreID	int(11)	MUL	
categorieArt	id	int(11)	PRI	auto_increment
categorieArt	nome	varchar(20)		
categorieArt	descrizione	varchar(1000)		
categorieOggetto	id	int(11)	PRI	auto_increment
categorieOggetto	nome	varchar(20)		
categorieOggetto	descrizione	varchar(200)		
collection	id	bigint(20)	PRI	auto_increment
collection	titolo	varchar(30)		
collection	utenteID	bigint(20)	MUL	
collection	descrizione	varchar(500)		
collection	data	timestamp		
colore	id	int(11)	PRI	auto_increment
colore	colore	varchar(20)		
colore_abilitato	oggettoID	int(11)	PRI	
colore_abilitato	artID	bigint(20)	PRI	
colore_abilitato	coloreID	int(11)	PRI	
commenti	id	int(11)	PRI	auto_increment
commenti	commento	varchar(500)		
commenti	data	timestamp		
commenti	utenteID	bigint(20)	MUL	
commenti	artID	bigint(20)	MUL	
compravendite	id	bigint(20)	PRI	auto_increment
compravendite	compratore	bigint(20)		
compravendite	venditore	bigint(20)		
compravendite	artID	bigint(20)		
compravendite	oggettoID	int(11)		

TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_TYPE	COLUMN_KEY	EXTRA
compravendite	tagliaID	int	int(11)		
compravendite	coloreID	int	int(11)		
logs	id	bigint	bigint(20)	PRI	auto_increment
logs	aint	int	int(11)		
logs	bint	int	int(11)		
logs	cbig	int	int(11)		
oggetto	id	int	int(11)	PRI	auto_increment
oggetto	nome	varchar	varchar(20)		
oggetto	descrizione	varchar	varchar(200)		
oggetto	categorialD	int	int(11)	MUL	
oggetto_colore	coloreID	int	int(11)	PRI	
oggetto_colore	oggettoID	int	int(11)	PRI	
piace_art	utenteID	bigint	bigint(20)	PRI	
piace_art	artiD	bigint	bigint(20)	PRI	
piace_post	utenteID	bigint	bigint(20)	PRI	
piace_post	postID	bigint	bigint(20)	PRI	
post	id	bigint	bigint(20)	PRI	auto_increment
post	testo	varchar	varchar(500)		
post	titolo	varchar	varchar(50)		
post	utenteID	bigint	bigint(20)	MUL	
post	data	timestamp	timestamp		
prodotto	oggettoID	int	int(11)	PRI	
prodotto	artiD	bigint	bigint(20)	PRI	
segue	utente1	bigint	bigint(20)	PRI	
segue	utente2	bigint	bigint(20)	PRI	
tag	artiD	bigint	bigint(20)	PRI	
tag	nome	varchar	varchar(10)	PRI	
taglia	id	int	int(11)	PRI	auto_increment
taglia	taglia	varchar	varchar(20)		
taglia_oggetto	tagliaID	int	int(11)	PRI	
taglia_oggetto	oggettoID	int	int(11)	PRI	
taglia_oggetto	prezzo	int	int(11)		
utenti	id	bigint	bigint(20)	PRI	auto_increment
utenti	nome	varchar	varchar(30)	UNI	
utenti	email	varchar	varchar(50)	UNI	
utenti	password	varchar	varchar(100)		
utenti	data	timestamp	timestamp		

Trigger

A partire dalla versione 5.0.2, MySQL, ha introdotto i Trigger, ovvero un meccanismo attraverso il quale è possibile automatizzare tali operazioni al verificarsi di eventi riguardanti i dati, quali INSERT, UPDATE o DELETE.

Quando definiamo un Trigger in MySQL dobbiamo stabilire se questo debba innescarsi prima o dopo un certo evento.

Potremo quindi definire dei Trigger per queste diverse combinazioni di tempo/evento:

- BEFORE INSERT
- BEFORE UPDATE
- BEFORE DELETE
- AFTER INSERT
- AFTER UPDATE
- AFTER DELETE

Ciascun Trigger deve necessariamente essere associato ad una tabella.

Trigger 1 trig_segue:

```
3  DELIMITER $$  
4  create trigger trig_segue before insert on segue  
5  FOR EACH ROW  
6  BEGIN  
7      IF( NEW.utente1 = NEW.utente2 ) THEN  
8          SIGNAL SQLSTATE '45000'  
9          SET MESSAGE_TEXT = 'self following not allowed';  
10     END IF;  
11 END$$  
12 DELIMITER;
```

questo trigger si occupa di impedire che un utente si segua da solo quindi controlla che i dei attributi utente1 e utente2 siano uguali e nel caso segnala l'errore

Trigger 3 trig_compravendite:

```
15  DELIMITER //
16  create trigger trig_compravendite before insert on compravendite
17  FOR EACH ROW
18  BEGIN
19  if not exists(select 1 from utenti where NEW.compratore =utenti.id) THEN
20      SIGNAL SQLSTATE '45000'
21      SET MESSAGE_TEXT = 'errore';
22  END IF;
23
24  if not exists(select 1 from art where NEW.artID = art.id and NEW.venditore = art.utenteID) THEN
25      SIGNAL SQLSTATE '45000'
26      SET MESSAGE_TEXT = 'errore';
27  END IF;
28
29  if not exists(select 1 from prodotto where NEW.artID = prodotto.artID and NEW.oggettoID = prodotto.oggettoID ) THEN
30      SIGNAL SQLSTATE '45000'
31      SET MESSAGE_TEXT = 'prodotto non disponibile';
32  END IF;
33
34
35  if not exists(select 1 from taglia_oggetto where NEW.tagliaID = taglia_oggetto.tagliaID and NEW.oggettoID = taglia_oggetto.oggettoID ) THEN
36      SIGNAL SQLSTATE '45000'
37      SET MESSAGE_TEXT = 'taglia non disponibile';
38  END IF;
39
40  if not exists(select 1 from colore_abilitato where NEW.coloreID = colore_abilitato.coloreID and NEW.oggettoID = colore_abilitato.oggettoID ) THEN
41      SIGNAL SQLSTATE '45000'
42      SET MESSAGE_TEXT = 'colore non disponibile';
43  END IF;
44  END//
```

Questo trigger sulla tabella compravendite si occupa di mantenere la coerenza dei dati in quanto non essendoci foreign key nella tabella compravendite la coerenza nell'inserimento viene garantita attraverso questo trigger che controlla il caso in cui un utente prova a comprare un determinato prodotto con un colore o una taglia non disponibile questo trigger non lo permette

Trigger 4 trig_aggiungiProd :

```
48  DELIMITER $$|  
49  create trigger trig_aggiungiProd after insert on art  
50  FOR EACH ROW  
51  BEGIN  
52      DECLARE done INT DEFAULT FALSE;  
53      DECLARE oggID int;  
54      DECLARE oggCursor CURSOR FOR SELECT id FROM oggetto;  
55      DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  
56      OPEN oggCursor;  
57      mLoop: LOOP  
58          FETCH oggCursor INTO oggID;  
59          IF done THEN  
60              LEAVE mLoop;  
61          END IF;  
62          insert into prodotto(oggettoID,artID) values (oggID,NEW.id);  
63      END LOOP;  
64      CLOSE oggCursor;  
65  END $$  
66  DELIMITER;
```

Questo trigger si occupa di abilitare tutti gli oggetti su quando viene inserita una nuova art inserendo su prodotto l'art e tutti gli oggetti

Trigger 5 trig_aggiungiColore :

```
71  DELIMITER $$  
72  create trigger trig_aggiungiColore after insert on prodotto  
73  FOR EACH ROW  
74  BEGIN  
75      DECLARE done INT DEFAULT FALSE;  
76      DECLARE col int;  
77      DECLARE cursor1 CURSOR FOR select coloreID from oggetto_colore where oggettoID = NEW.oggettoID;  
78      DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;  
79      OPEN cursor1;  
80      mLoop: LOOP  
81          FETCH cursor1 INTO col;  
82          IF done THEN  
83              LEAVE mLoop;  
84          END IF;  
85          insert into colore_abilitato(oggettoID,artID,coloreID) values (NEW.oggettoID,NEW.artID,col);  
86      END LOOP;  
87      CLOSE cursor1;  
88  END$$  
89  DELIMITER;
```

Questo trigger si occupa di abilitare tutti i colori su un prodotto quanto quest'ultimo viene inserito

Trigger 6 trig_aggiungiColore :

```
99  DELIMITER $$  
100 create trigger trig_du BEFORE DELETE ON utenti  
101 FOR EACH ROW  
102 BEGIN  
103     delete from art where utenteID=OLD.id;  
104     delete from collection where utenteID=OLD.id;  
105     delete from post where utenteID=OLD.id;  
106     delete from piace_post where utenteID=OLD.id;  
107     delete from piace_art where utenteID=OLD.id;  
108     delete from commenti where utenteID=OLD.id;  
109     delete from segue where utente1=OLD.id;  
110     delete from segue where utente2=OLD.id;  
111 END $$  
112 DELIMITER;
```

quando si elimina un utente vengono cancellate tutte le sue art,post,mipiace,commenti,tutti i segue

Trigger 7 trig_artDel:

```
116  DELIMITER $$  
117  create trigger trig_artDel BEFORE DELETE ON art  
118  FOR EACH ROW  
119  BEGIN  
120      delete from artInCat where artID=OLD.id;  
121      delete from tag where artID=OLD.id;  
122      delete from piace_art where artID=OLD.id;  
123      delete from commenti where artID=OLD.id;  
124      delete from art_collection where artID=OLD.id;  
125      delete from prodotto where artID=OLD.id;  
126  END$$
```

quando viene eliminata un'art vengono eliminati anche tutti i commenti relativi all'art i mi piace,i prodotti ecc ecc..

Trigger 8 trig_prodottoDel:

```
129  DELIMITER $$  
130  create trigger trig_prodottoDel BEFORE DELETE ON prodotto  
131  FOR EACH ROW  
132  BEGIN  
133      delete from colore_abilitato where artID=OLD.artID and oggettoID= OLD.oggettoID;  
134      delete from carrello where artID=OLD.artID and oggettoID = OLD.oggettoID;  
135  END $$  
136
```

Quando viene eliminato un prodotto viene tolto dal carrello degli utenti ma non dalle compravendite e vengono cancellati i riferimenti al colore di questo prodotto

Trigger 9 trig_postDel:

```
138  DELIMITER $$  
139  create trigger trig_postDel BEFORE DELETE ON post  
140  FOR EACH ROW  
141  BEGIN  
142  |   delete from piace_post where postID = OLD.id;  
143  END $$  
144  DELIMITER;
```

Quando viene cancellato un post vengono cancellati tutti i mi piace di questo post

Inserimento dei Dati

L'inserimento dei dati all'interno del database avviene tramite codice da terminale.
Si utilizza il seguente comando:

```
3  INSERT INTO tabella (campo1,campo2) values [values1,values2]
```

Inoltre mysql offre il bulk insert che consiste di inserire più record con una singolo inserimento.

```
3  INSERT INTO tabella (campo1,campo2) values (values1,values2),(values3,values3),.....,(valuesN,valuesM);
```

Il bulk insert è stato molto utile nella fase dell'inserimento dei dati in quanto sono riuscito a inserire anche più di 100.000 record attraverso una singola query ottimizzando di molto il processo di inserimento

Metodo di inserimento

Gli inserimenti sono stati realizzati attraverso script in python collegandosi al database in questo modo

```
1 import MySQLdb
2 import csv
3 import string
4 import random
5 db = MySQLdb.connect(host="localhost",      # your host, usually localhost
6                       user="root",          # your username
7                       passwd="toor",        # your password
8                       db="society6")       # name of the data base
9
```

Inserimenti tabella utenti:

```
12 cur = db.cursor()
13 gestori = ['@gmail.it', '@yahoo.it', '@libero.it', '@webmail.com', '@gestore.it']
14 file = open('nomi.csv', 'r')
15 lines=file.readlines()
16 for k in range(20):
17     for x in lines:
18         nome=x[:-2]+str(k)
19         email=x[:-2]+str(k)+random.choice(gestori)
20         password = ''.join([random.choice(string.ascii_letters + string.digits) for n in random xrange(32)])
21         stri = "INSERT INTO utenti(nome,email,password) VALUES('{}','{}','{}')".format(nome,email,password)
22         cur.execute(stri)
23         db.commit()
24
25 db.close()
```

Il nome dell'utente è stato preso da un file con dentro una lista nomi e poi concateno un numero al nome l'email è formata dal nome concatenato il restante dell'email scelto casualmente, la password viene generata casualmente

Inserimento tabella art:

```
19 i= 1
20 stri = "INSERT into art(titolo,testo,utenteID) values "
21 while (i < 200001):
22     for k in range(1,random.randint(1,5)):
23         art='art'+''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(6))
24         stri=stri+("'{0}', '{1}', {2} )".format(art,art+" Lorem ipsum dolor sit amet, consectetur adipis
25         i=i+1
26 stri=stri[:-1]+';'
27 print('inserimento')
28 cur.execute(stri)
29 print('commit')
30 db.commit()
31 print(i)
32 db.close()
```

il nome dell'art viene generato dalla stringa art più una parola random si può vedere come andiamo a inserire minimo da minimo 1 art a massimo 5 art per utente in una singola query

Inserimento tabella piace_art:

```
12 cur = db.cursor()
13
14 stri="insert into piace_art(utenteID,artID) values "
15 i=30000
16 while i< 40000:
17     k=random.randint(0,100) #200740
18     list=random.sample(range(1,10740), k)
19     random.choice
20     for x in list:
21         stri= stri+"('{}','{}'),".format(i,x)
22     i=i+1
23 stri=stri[:-1]+';'
24 print(i)
25 cur.execute(stri)
26 print(i)
27 db.commit()
28 db.close()
```

Si può vedere come un utente mette minimo 0 massimo 100 mi piace alle art random tra (1,10740)
sono stati inseriti i mi piace di 10000 utenti ad ogni esecuzione dello script
si noti come viene eseguita una singola query per esecuzione

Inserimento tabella post:

```
19 i= 80000
20 stri = "INSERT into post(titolo,testo,utenteID) values "
21 while (i < 20000+20000+20000+20000+20000):
22
23
24     for k in range(1,random.randint(1,5)):
25         art='art'+''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(6))
26         stri=stri+"('{}','{}',{},{})".format(art,art+" Lorem ipsum dolor sit amet, consectetur adipis
27     i=i+1
28 stri=stri[:-1]+';'
29 print('inserimento')
30 cur.execute(stri)
31 print('commit')
32 db.commit()
33 print(i)
34 db.close()
```

Inserimento tabella piace_post:

```
12 cur = db.cursor()
13
14 stri="insert into piace_post(utenteID,postID) values "
15 i=40000
16 while i< 50000:
17     k=random.randint(0,100) #200740
18     list=random.sample(range(1,10000), k)
19     for x in list:
20         stri= stri+"('{}','{}'),".format(i,x)
21     i=i+1
22 stri=stri[:-1]+';'
23
24 #print(stri)
25 cur.execute(stri)
26 print(i)
27 db.commit()
28 db.close()
```

Inserimento tabella tag:

```
20  def randlist(k):
21      list =[]
22      for i in range(k):
23          list.append(''.join(random.choice(string.ascii_uppercase ) for _ in range(6)))
24      return list
25
26  |
27  cur = db.cursor()
28  i= 1
29  stri = "INSERT into tag(artID,nome) values "
30  #188055
31  #100000
32  #50000
33  while i< 50000:
34      k=random.randint(0,20)
35      list= randlist(k)
36      for x in list:
37          stri= stri+"('{}','{}'),".format(i,x)
38          i=i+1
39  stri=stri[:-1]+';'
40  cur.execute(stri)
41  db.commit()
```

Inserimento tabella segue:

```
19 cur = db.cursor()
20 i=205
21 x=3
22 while (i < 103262):
23     f=open('file.txt','w+')
24     stri= "INSERT INTO segue(utente1,utente2) VALUES"
25     while(x <103262):
26         stri = stri +"({},{})".format(x,i)
27         x=x+1
28     x=3
29     f.write(' dd '+str(i)+' dd ')
30     f.close()
31     stri=stri[:-1]+';'
32     i=i+1
33     cur.execute(stri)
34     db.commit()
35 db.close()
```

in questa tabella ci sono 104933395 record sono stati inseriti 103262 alla volta ed stato lasciato lo script in esecuzione per una notte la tabella occupa circa 7 gb sul disco

Inserimento tabella collection

```
12 cur = db.cursor()
13
14 des ="Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
15 i=1
16 stri="insert into collection(titolo,utenteID,descrizione) values "
17 count=i
18 while i<20000:
19     for x in range(1,random.randint(1,10)):
20         stri= stri+"('{}','{}','{}')".format('collection numero'+str(count),i,des)
21         count=count+1
22     i=i+5
23
24 stri=stri[:-1]+';'
25 cur.execute(stri)
26 db.commit()
27 db.close()
```

Inserimento tabella artInCat

```
12 cur = db.cursor()
13
14 des ="Lorem ipsum dolor sit amet, consectetur adipiscing
15
16 stri="insert into artInCat(artID,categoriaID) values "
17 for i in range(1,188054):
18     k=random.randint(0,10)
19     list=random.sample(range(1,27), k)
20     for x in list:
21         stri=stri+"('{}','{}'),".format(i,x)
22 stri=stri[:-1]+';'
23 cur.execute(stri)
24 db.commit()
25 db.close()
```

Inserimento tabella categorieArt

```
12 cur = db.cursor()
13
14 des ="Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
15
16 for i in range(1,27):
17     stri="insert into categorieArt(nome,descrizione) values ('{}','{}');".format('categoria '+str(i),des)
18     print(stri)
19     cur.execute(stri)
20     db.commit()
21 db.close()
22
```

Inserimento tabella collection

```
12 cur = db.cursor()
13 des ="Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
14 i=1
15 stri="insert into collection(titolo,utenteID,descrizione) values "
16 count=i
17 while i<20000:
18     for x in range(1,random.randint(1,10)):
19         stri= stri+"('{}','{}','{}')".format('collection numero'+str(count),i,des)
20         count=count+1
21     i=i+5
22 stri=stri[:-1]+';'
23 cur.execute(stri)
24 db.commit()
25 db.close()
```

Inserimento tabella art_collection

```
10 cur = db.cursor()
11 des ="Lorem ipsum dolor sit amet, consectetur adipiscing elit,
12 i=1
13 stri="insert into art_collection(artID,collectionID) values "
14 for i in range(1,18038):
15     k=random.randint(0,50)
16     list=random.sample(range(1,188055), k)
17     for x in list:
18         stri= stri+"('{}','{}')".format(x,i)
19 stri=stri[:-1]+';'
20 cur.execute(stri)
21 db.commit()
22 db.close()
```

Inserimento tabella categorieOggetto

```
10 |
11 INSERT into categorieOggetto(nome,descrizione) VALUES ('vestiti','abbigliamento vestiario magliette ecc..')
12 INSERT into categorieOggetto(nome,descrizione) VALUES ('tecnologia','cover per iphon telefoni ecc..')
13 INSERT into categorieOggetto(nome,descrizione) VALUES ('quadri','quadri portraoit ecc ecc ');
14 INSERT into categorieOggetto(nome,descrizione) VALUES ('bagno','bagno tappetini ecc ecc');
15 INSERT into categorieOggetto(nome,descrizione) VALUES ('mobili','mobili per la casa');
16 INSERT into categorieOggetto(nome,descrizione) VALUES ('tazze','tazze bichieri ecc');
17 INSERT into categorieOggetto(nome,descrizione) VALUES ('borse',' zaini borse ecc ecc');
18 INSERT into categorieOggetto(nome,descrizione) VALUES ('cuscini e lenzuola','bagno tappetini ecc ecc');
19 INSERT into categorieOggetto(nome,descrizione) VALUES ('stikers','stikers pellicole protezione ecc ecc');
20 INSERT into categorieOggetto(nome,descrizione) VALUES ('decorazioni','tende ecc');
```

Inserimento tabella oggetto

```
24
25 INSERT into oggetto(nome,descrizione,categoriaID) values ('cover iphone','cover per iphone',1);
26 INSERT into oggetto(nome,descrizione,categoriaID) values ('cover laptop','cover per laptop',1);
27
28 INSERT into oggetto(nome,descrizione,categoriaID) values ('maglietta uomo','maglietta uomo larga',2);
29 INSERT into oggetto(nome,descrizione,categoriaID) values ('leggins','leggins per donna',2);
30
31 INSERT into oggetto(nome,descrizione,categoriaID) values ('wall art','arte da muro',3);
32 INSERT into oggetto(nome,descrizione,categoriaID) values ('canvas print','canvas',3);
33
34 INSERT into oggetto(nome,descrizione,categoriaID) values ('tappetino','tappetino da bagno',4);
35 INSERT into oggetto(nome,descrizione,categoriaID) values ('tenda da doccia','cover per laptop',4);
36
37 INSERT into oggetto(nome,descrizione,categoriaID) values ('tavolo','tavolo con art',5);
38 INSERT into oggetto(nome,descrizione,categoriaID) values ('sgabello','sgabello con art',5);
39
40 INSERT into oggetto(nome,descrizione,categoriaID) values ('coffe mug','tazza da caffè',6);
41 INSERT into oggetto(nome,descrizione,categoriaID) values ('borraccia','borraccia per liquidi',6);
42
43 INSERT into oggetto(nome,descrizione,categoriaID) values ('zaino','zaino classico',7);
44 INSERT into oggetto(nome,descrizione,categoriaID) values ('borsa','borsa da donna',7);
45
46 INSERT into oggetto(nome,descrizione,categoriaID) values ('piumone','piumone da letto',8);
47 INSERT into oggetto(nome,descrizione,categoriaID) values ('cuscino quadrato',' cuscino di forma qadrata',8);
48
49 INSERT into oggetto(nome,descrizione,categoriaID) values ('adesivo','adesivo classico',9);
50 INSERT into oggetto(nome,descrizione,categoriaID) values ('adesivo per pc','fsadfdsfsdfco',9);
51
52 INSERT into oggetto(nome,descrizione,categoriaID) values ('tenda per finestra','tenda molto bella',10);
53 INSERT into oggetto(nome,descrizione,categoriaID) values ('tenda per porta','moltasdsad',10);
```

Inserimento tabella taglia

```
70  insert into taglia(taglia) values ('XXS'),('XS'),('S'),('M'),('L'),('XL'),('XXL'),('GRANDE'),('MEDIO'),('PICCOLO');
71  insert into taglia(taglia) values ('UNICA')
```

Inserimento tabella colore

```
60  insert into colore(colore) values ('rosso');
61  insert into colore(colore) values ('unico');
62  insert into colore(colore) values ('bianco');
```

Inserimento tabella oggetto_colore

```
66  insert into oggetto_colore(coloreID,oggettoID) values (1,2);
67  insert into oggetto_colore(coloreID,oggettoID) values (2,2);
68  insert into oggetto_colore(coloreID,oggettoID) values (3,2);
69  insert into oggetto_colore(coloreID,oggettoID) values (4,2);
70  insert into oggetto_colore(coloreID,oggettoID) values (5,2);
71  insert into oggetto_colore(coloreID,oggettoID) values (6,2);
72  insert into oggetto_colore(coloreID,oggettoID) values (7,1);
```

Inserimento tabella carrello

```
24 def getColTag(ogg):
25     taglia = []
26     colore=[]
27     cur.execute('select * from taglia_oggetto')
28     tag =cur.fetchall()
29     for i in tag:
30         if (i[1] == ogg):
31             taglia.append(i[0])
32     cur.execute('select * from oggetto_colore')
33     col=cur.fetchall()
34     for i in col:
35         if (i[1] == ogg):
36             colore.append(i[0])
37     return {'taglia':random.choice(taglia),'colore':random.choice(colore)}
38
39 cur = db.cursor()
40 stri="insert into carrello(utenteID,artID,oggettoID,tagliaID,coloreID) values "
41 utente=1
42 while utente < 100000:
43
44     for i in range(1,random.randint(1,10)):
45         ogg = random.randint(1,21)
46         art = random.randint(4,100000)
47         colTag= getColTag(ogg)
48         stri = stri + "({},{},{},{},{})".format(utente,art,ogg,colTag['taglia'],colTag['colore'])
49         utente=utente+1
50     stri=stri[:-1]+';'
51     #print(stri)
52     cur.execute(stri)
53     print(utente)
54     db.commit()
55     db.close()
```

Inserimento tabella compravendite

```
23 def getColTag(ogg):
24     taglia = []
25     colore=[]
26     cur.execute('select * from taglia_oggetto')
27     tag =cur.fetchall()
28     for i in tag:
29         if (i[1] == ogg):
30             taglia.append(i[0])
31     cur.execute('select * from oggetto_colore')
32     col=cur.fetchall()
33     for i in col:
34         if (i[1] == ogg):
35             colore.append(i[0])
36     return {'taglia':random.choice(taglia),'colore':random.choice(colore)}
37
38 def getProprietaro(artid):
39     cur.execute('select utenteID from art where id='+str(artid))
40     pro=cur.fetchall()
41     return pro[0][0]
42
43 cur = db.cursor()
44 stri="insert into compravendite(compratore,venditore,artID,oggettoID,tagliaID,coloreID) values "
45 getProprietaro(4)
46 utente=1
47 while utente < 100000:
48
49     for i in range(1,random.randint(1,10)):
50         ogg = random.randint(1,21)
51         art = random.randint(4,100000)
52         colTag= getColTag(ogg)
53         venditore = getProprietaro(art)
54         stri = stri + "({},{},{},{},{})".format(utente,venditore,art,ogg,colTag['taglia'],colTag['colore'])
55         utente=utente+1
56     stri=stri[:-1]+';'
57 #print(stri)
58 cur.execute(stri)
59 print(utente)
60 db.commit()
61 db.close()
```

Inserimento tabella taglia

```
12 cur = db.cursor()
13 for i in range(1,22):
14     if(i==1 or i==3 or i==4 or i==13 or i==14 or i==20 or i==21 or i==22):
15         cur.execute("insert into taglia_oggetto(tagliaID,oggettoID,prezzo) values ({},{},{});".format(10,i[random.randint(10, 100)]))
16         db.commit()
17     elif(i==2 or i==5 or i==6 ):
18         for x in range(1,7):
19             cur.execute("insert into taglia_oggetto(tagliaID,oggettoID,prezzo) values ({},{},{});".format(x,i[random.randint(10, 100)]))
20             db.commit()
21     else:
22         for x in range(8,10):
23             cur.execute("insert into taglia_oggetto(tagliaID,oggettoID,prezzo) values ({},{},{});".format(x,i[random.randint(10, 100)]))
24             db.commit()
25 db.close()
```

Numero di Record per Tabella

```
SELECT table_name,table_rows FROM information_schema.tables WHERE  
table_schema = 'society6';
```

table_name	table_rows
art	195508
artInCat	948868
art_collection	449625
carrello	144560
categorieArt	26
categorieOggetto	10
collection	17472
colore	7
colore_abilitato	7658385
commenti	410550
compravendite	203764
logs	0
oggetto	21
oggetto_colore	41
piace_art	1845158
piace_post	2579329
post	392952
prodotto	4048737
segue	104933395
tag	498753
taglia	11
taglia_oggetto	47
utenti	103020

Query

Query 1:

selezionare tutti gli oggetti e i possibili colori che possono avere

```
select oggetto.nome,colore.colore from oggetto,colore,oggetto_colore  
where colore.id = oggetto_colore.coloreID and oggetto_colore.oggettoID  
= oggetto.id order by oggetto.nome;
```

nome	colore
adesivo	unico
adesivo per pc	unico
borraccia	unico
borsa	unico
canvas print	unico
coffe mug	giallo
coffe mug	verde
coffe mug	bianco
coffe mug	nero
coffe mug	rosso
cover iphone	unico
cover laptop	unico
cover samsung	unico
cuscino quadrato	unico
leggins	giallo
leggins	verde
leggins	bianco
leggins	nero
leggins	rosso
maglietta uomo	giallo
maglietta uomo	verde
maglietta uomo	bianco
maglietta uomo	unico
maglietta uomo	nero
maglietta uomo	rosso

Mostro le righe 0 - 24 (41 del totale, La query ha impiegato 0.0006 secondi)

Query 2:

seleziona tutti gli utenti con nome,email e quante art hanno pubblicato ordinato per il numero di art (da quello che ne ha pubblicate di più)

```
select distinct utenti.nome,utenti.email,count(utenti.nome) from
utenti JOIN art ON utenti.id = art.utenteID GROUP BY utenti.nome order
by count(utenti.nome) desc;
```

nome	email	count(utenti.nome)
Lisa10	Lisa10@yahoo.it	8
Maira9	Maira9@libero.it	8
Dominique8	Dominique8@gmail.it	8
Jacquelyn11	Jacquelyn11@libero.it	8
Analisa10	Analisa10@gestore.it	8
Iesha8	Iesha8@gmail.it	8
Debra9	Debra9@gestore.it	8
Joie10	Joie10@gestore.it	8
Demetra9	Demetra9@libero.it	8
Bill9	Bill9@gmail.it	8
Elina8	Elina8@yahoo.it	8
Chu11	Chu11@gestore.it	8
Stormy7	Stormy7@yahoo.it	8
Carlita9	Carlita9@yahoo.it	8
Carmen9	Carmen9@yahoo.it	8
Jessi8	Jessi8@libero.it	8
Gale11	Gale11@webmail.com	8
Mitsuko9	Mitsuko9@yahoo.it	8
Joleen8	Joleen8@yahoo.it	8
Chae9	Chae9@webmail.com	8
Filiberto8	Filiberto8@libero.it	8
Chantal9	Chantal9@gmail.it	8
Gianna11	Gianna11@libero.it	8
Giovanni11	Giovanni11@yahoo.it	8
Alejandro9	Alejandro9@gmail.it	8

Mostro le righe 0 - 24 (67361 del totale, La query ha impiegato 0.2491 secondi.)

Query 3:

selezionare tutti gli utenti che hanno comprato un prodotto di colore rosso con where(prima) e con join(seconda)

```
select utenti.nome from utenti,compravendite,colore where utenti.id = compravendite.compratore and compravendite.coloreID = colore.id and colore.colore = 'rosso';
```

```
select utenti.nome from utenti join compravendite inner join colore ON utenti.id = compravendite.compratore and compravendite.coloreID = colore.id and colore.colore = 'rosso';
```

nome
Akilah0
Akilah0
Alene0
Aliza0
Allena0
Allyson0
Alyce0
Ammie0
Amy0
Anderson0
Andy0
Annalee0
Annalee0
Annamaria0
Annett0
Arden0
Ariel0
Arlean0
Armando0
Armando0
Aron0
Ashanti0
Asley0
Aurore0
Ava0

Mostro le righe 0 - 24 (9579 del totale, La query ha impiegato 0.0005 secondi.)

Query 4:

selezionare rispettivamente id art il titolo dell'art e quante volte è stata venduta e quanti mi piace gli sono stati messi

```
select art.id,art.titolo,count(art.id) ,piace from (select art.id as idartP,count(piace_art.artID) as piace from art,piace_art where art.id =piace_art.artID GROUP BY piace_art.artID) as pic,art,compravendite where art.id = compravendite.artID and pic.idartP = art.id GROUP BY art.id;
```

id	titolo	count(art.id)	piace
4	artRVYZMF	5	51
6	artDTCK2J	1	47
7	art4L6WQW	2	65
9	artY6L7NE	3	50
10	art1W732F	5	43
11	art2LTZEY	1	62
12	artR50VV4	4	59
13	artCBFELX	3	62
15	artYUPOAL	2	49
16	artQD6K1Q	2	47
17	artVYV2HD	1	50
18	artFHS0UF	1	59
20	artR1JFGC	2	43
21	artWOHUD2	5	51
22	artYQZBCJ	3	50
23	artAP2R56	3	63
24	artRNHR5Y	1	36
25	artJA28RV	2	53
26	artSWUG72	3	50
27	artNJHN3K	1	55
28	artAJ3XW6	3	52
29	art6NMKNZ	3	50
30	artMQFUNK	1	54
31	artLSG229	4	47
32	artKW2H5I	1	44

Mostro le righe 0 - 24 (86563 del totale, La query ha impiegato 5.3908 secondi.)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra	
1	PRIMARY	compravendite	NULL	ALL		NULL	NULL	NULL	203764	100.00	Using temporary; Using filesort	
1	PRIMARY	art	NULL	eq_ref	PRIMARY	utenteID	PRIMARY	8	society6.compravendite.artID	1	100.00	
1	PRIMARY	<derived2>	NULL	ref	<auto_key0>	<auto_key0>	8	society6.compravendite.artID	10	100.00		
2	DERIVED	art	NULL	index	PRIMARY	utenteID	8		195508	100.00	Using index; Using temporary; Using filesort	
2	DERIVED	place_art	NULL	ref	PRIMARY	utenteID	PRIMARY	8	society6.art.id	9	100.00	Using index

ottimizzata:

```
select straight_join art.id,art.titolo,count(art.id) ,piace from compravendite,(select straight_join art.id as idartP,count(piace_art.artID) as piace from piace_art,art where art.id =piace_art.artID GROUP BY piace_art.artID) as pic,art where art.id =compravendite.artID and pic.idartP = art.id GROUP BY art.id;
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra	
1	PRIMARY	compravendite		NULL	ALL					203764	100.00	Using temporary; Using filesort
1	PRIMARY	<derived2>		NULL	ref	<auto_key0>	8	society6.compravendite.artID	10	100.00	NULL	
1	PRIMARY	art		NULL	eq_ref	PRIMARY,utenteID	8	society6.compravendite.artID	1	100.00	NULL	
2	DERIVED	piace_art		NULL	index	PRIMARY,utenteID	16		NULL	1845158	100.00	Using index
2	DERIVED	art		NULL	eq_ref	PRIMARY	8	society6.piace_art.artID	1	100.00	Using index	

Mostro le righe 0 - 24 (86563 del totale, La query ha impiegato 4.2712 secondi.)

La query è stata ottimizzata attraverso l'uso dello straight_join cambiando così l'ordine in cui mysql accede alle tabelle. andando a mettere come prima tabella compravendite che è la tabella più corposa

Query 5:

selezionare rispettivamente gli utenti che hanno comprato e il totale di quanto hanno speso

```
select utenti.nome,sum(taglia_oggetto.prezzo) from
utenti,compravendite,taglia_oggetto where utenti.id =
compravendite.compratore and compravendite.oggettoID =
taglia_oggetto.oggettoID and compravendite.tagliaID GROUP by utenti.id;
```

nome	sum(taglia_oggetto.prezzo)
caglio	687
Aaron0	596
Abbey0	774
Abbie0	506
Abdul0	285
Abe0	349
Abel0	98
Abigail0	920
Abraham0	676
Abram0	889
Ada0	1192
Adah0	838
Adalberto0	2865
Adaline0	98
Adam0	875
Adelaide0	229
Adele0	647
Adelia0	251
Adeline0	534
Adell0	247
Adella0	543
Adelle0	1059
Adena0	1365
Adina0	454
Adolph0	467

Mostro le righe 0 - 24 (38375 del totale, La query ha impiegato 1.0063 secondi.)

Query 7:

selezionare tutti i colori e quanti prodotti sono stati venduti con quel colore

```
select colore.colore, count(colore.id) from colore,compravendite where  
colore.id = compravendite.coloreID GROUP by colore.id;
```

colore	count(colore.id)
rosso	9579
nero	9629
bianco	9670
verde	9652
giallo	9756
unico	156046

Mostro le righe 0 - 5 (6 del totale, La query ha impiegato 0.1068 secondi.)

Query 8:

selezionare i primi 10 utenti con più follower e quanti follower hanno

```
select utenti.nome, count(kk.id) from segue,utenti,utenti as kk where  
utenti.id = segue.utente1 and kk.id = segue.utente2 GROUP by utenti.id  
limit 10
```

nome	count(kk.id)
caglio	803
Aaron0	803
Abbey0	803
Abbie0	803
Abby0	803
Abdul0	803
Abe0	803
Abel0	803
Abigail0	803
Abraham0	803

Mostro le righe 0 - 9 (10 del totale, La query ha impiegato 0.0244 secondi.)

Query 8:

selezionare i primi 10 utenti che seguono più utenti e quanti utenti seguono

```
select utenti.nome, count(kk.id) from utenti, segue, utenti as kk where  
utenti.id = segue.utente2 and kk.id = segue.utente1 GROUP by utenti.id  
limit 10
```

nome	count(kk.id)
caglio	103259
Aaron0	103259
Abbey0	103259
Abbie0	103259
Abby0	103259
Abdul0	103259
Abe0	103259
Abel0	103259
Abigail0	103259
Abraham0	103259

Mostro le righe 0 - 9 (10 del totale, La query ha impiegato 138.8962 secondi.)

Query 9:

selezionare tutte le art con le collection di cui fa parte
con group concat

```
select art.id, art.titolo, group_concat(collection.titolo) from  
art, collection, art_collection where collection.id =  
art_collection.collectionID and art.id = art_collection.artID GROUP by  
art.id
```

senza group_concat

```
select art.id,art.titolo,collection.titolo from  
art,collection,art_collection where collection.id =  
art_collection.collectionID and art.id = art_collection.artID
```

id	titolo	group_concat(collection.titolo)
1	ceci ne pas une pipe	collection numero10865, collection numero12263, coll...
2	artFMPLMR	collection numero4695, collection numero5355, collec...
3	artITL0WS	collection numero2271, collection numero11761
4	artRVYZMF	collection numero1017, collection numero11804
5	art362Y00	collection numero10095
6	artDTCK2J	collection numero2285, collection numero13075, colle...
7	art4L6WQW	collection numero2435
8	artW3KF72	collection numero10781, collection numero15471
9	artY6L7NE	collection numero1398, collection numero2138, collec...
10	art1W732F	collection numero15305
11	art2LTZEY	collection numero6726, collection numero9796, collec...
12	artR50VV4	collection numero7660
14	artAI5YJ9	collection numero1226, collection numero11662, colle...
15	artYUPOAL	collection numero3306, collection numero6017, collec...
16	artQD6K1Q	collection numero166, collection numero3597, collect...
17	artVYV2HD	collection numero1703, collection numero9684, collec...
18	artFHS0UF	collection numero16206, collection numero17308
20	artR1JFGC	collection numero16505
21	artWOHUD2	collection numero6117
22	artYQZBCJ	collection numero4939, collection numero6987, collec...
23	artAP2R56	collection numero8159, collection numero14471
24	artRNHR5Y	collection numero90, collection numero15221
25	artJA28RV	collection numero1938, collection numero6398
26	artSWUG72	collection numero3666, collection numero5704, collec...
27	artNJHN3K	collection numero3320, collection numero8731, collec...

Mostro le righe 0 - 24 (170834 del totale, La query ha impiegato 0.0473 secondi.)

Query 10:

selezionare tutti i prodotti con il nome dell'art e dell'oggetto

```
select art.titolo,oggetto.nome from
```

titolo	nome
ceci ne pas une pipe	cover iphone
ceci ne pas une pipe	maglietta uomo
ceci ne pas une pipe	cover samsung
ceci ne pas une pipe	cover laptop
ceci ne pas une pipe	maglietta uomo
ceci ne pas une pipe	leggins
ceci ne pas une pipe	wall art
ceci ne pas une pipe	canvas print
ceci ne pas une pipe	tappetino
ceci ne pas une pipe	tenda da doccia
ceci ne pas une pipe	tavolo
ceci ne pas une pipe	sgabello
ceci ne pas une pipe	coffe mug
ceci ne pas une pipe	borraccia
ceci ne pas une pipe	zaino
ceci ne pas une pipe	borsa
ceci ne pas une pipe	piumone
ceci ne pas une pipe	cuscino quadrato
ceci ne pas une pipe	adesivo
ceci ne pas une pipe	adesivo per pc
ceci ne pas une pipe	tenda per finestra
artFMPLMR	cover iphone
artFMPLMR	maglietta uomo
artFMPLMR	cover samsung
artFMPLMR	cover laptop

Mostro le righe 0 - 24 (4215540 del totale, La query ha impiegato (1 min 4.34 sec)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
											NULL
1	SIMPLE	oggetto	NULL	ALL	PRIMARY	NULL	NULL	NULL	21	100.00	
1	SIMPLE	art	NULL	ALL	PRIMARY	NULL	NULL	NULL	195508	100.00	Using join buffer (Block Nested Loop)
1	SIMPLE	prodotto	NULL	eq_ref	PRIMARY,artID	PRIMARY	12	society6.oggetto.id,society6.art.id	1	100.00	Using index

questa query può essere ottimizzata di molto attraverso l'uso di uno straight_join andando a cambiare l'ordine in cui mysql visita le tabelle andando a visitare prima oggetto poi art e infine prodotto

```

select straight_join art.titolo,oggetto.nome from prodotto,oggetto,art
where oggetto.id = prodotto.oggettoID and art.id = prodotto.artID

```

Mostro le righe 0 - 24 (4215540 del totale, La query ha impiegato 12.12 sec)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	prodotto	NULL	index	PRIMARY,artID	artID	8		NULL	4048737	100.00
1	SIMPLE	oggetto	NULL	eq_ref	PRIMARY	PRIMARY	4	society6.prodotto.oggettoID	1	100.00	NULL
1	SIMPLE	art	NULL	eq_ref	PRIMARY	PRIMARY	8	society6.prodotto.artID	1	100.00	NULL

Query 11:

selezionare gli utenti ciccioni

```

select utenti.nome from utenti,compravendite,taglia where utenti.id =
compravendite.compratore and compravendite.tagliaID = taglia.id and
taglia.taglia = 'XL';

```

nome
Ahmad0
Akilah0
Aletha0
Amiee0
Anderson0
Andy0
Armando0
Aurore0
Barbara0
Barbara0
Barbra0
Barbra0
Bernadine0
Blake0
Brandy0
Bula0
Caitlin0
Cameron0
Candis0
Carmelia0
Cathie0
Cathryn0
Celina0
Chara0
Cherrie0

Mostro le righe 0 - 24 (4917 del totale, La query ha impiegato (0.12 sec))

Query 12:

selezionare tutti gli utenti che hanno comprato lo stesso prodotto

```
con select nidificata
select utenti.nome from utenti,compravendite,(select utenti.id as
id1,compravendite.oggettoID as oggID,compravendite.artID as arID from
utenti,compravendite where utenti.id = compravendite.compratore) as
comp where utenti.id = compravendite.compratore and utenti.id !==
comp.id1 and compravendite.oggettoID = comp.oggID and
compravendite.artID = comp.arID limit 1000;
1000 rows in set (2 min 10.84 sec)
```

nome

Albertha0

Amanda0

Barbara0

Beau0

Candyce0

Cariota0

Charlena0

Danille0

Elane0

Esmeralda0

Francine0

Glennie0

Hilton0

Illuminada0

Julian0

Kimi0

Lottie0

Mabelle0

Margot0

Mitsuko0

Necole0

Quintin0

Rayna0

Samantha0

Taina0

con alias

```
select utenti.nome from utenti,compravendite as
ven,compravendite,utenti as com where utenti.id =
compravendite.compratore and com.id = ven.compratore and
utenti.id != com.id and compravendite.artID = ven.artID and
compravendite.oggettoID= ven.oggettoID limit 1000;
```

```
1000 rows in set (1 min 44.02 sec)
```

In questo caso la query è stata ottimizzata attraverso l'uso di una alias al posto di una select nidificata portando un incremento di velocità di quasi il 50% si noti che se la query non viene limitata tempo incrementata di mezz'ora e oltre (ho bloccato la query dopo mezz'ora)

Query 13:

selezionare tutte le art degli utenti che hanno 'mar' contenuto nel nome

```
select art.id,art.titolo,utenti.nome from art,utenti where art.utenteID  
= utenti.id and utenti.nome LIKE '%mar%'
```

id	titolo	nome
10733	artOISD1D	Anamaria1
104167	art1R0EA2	Anamaria10
104168	artWQKV43	Anamaria10
144203	artJLCWOJ	Anamaria10
144204	artVLAM8B	Anamaria10
144205	art0DGCTS	Anamaria10
144206	artX0UD0O	Anamaria10
114534	art9HRPI6	Anamaria11
114535	art2JTCJL	Anamaria11
114536	art7U0Z53	Anamaria11
154497	artTMCG2V	Anamaria11
154498	artS8MRDF	Anamaria11
154499	artP3692A	Anamaria11
166142	artFAS4PQ	Anamaria16
166143	art35V1RC	Anamaria16
166144	artRXNAW	Anamaria16
186891	artE9V6P3	Anamaria18
186892	art9AY85G	Anamaria18
186893	artO6JF1C	Anamaria18
197305	artUZUHUL	Anamaria19
197306	artNN2HDD	Anamaria19
197307	artBM80TJ	Anamaria19
197308	art0L7OJ2	Anamaria19
21046	artK197VL	Anamaria2
21047	artUGBKAJ	Anamaria2

Mostro le righe 0 - 24 (6759 del totale, La query ha impiegato (0.19 sec))

Query 14:

selezinare le art che hanno almeno 70 mi piace ,che sono state vendute almeno 5 volte e che sono presenti almeno 6 volte nel carrello

```
select art.id,art.titolo from compravendite,art,carrello,piace_art  
where art.id = piace_art.artID and art.id = carrello.artID and  
compravendite.artID= art.id GROUP by art.id having  
count(piace_art.artID) > 70 and count(compravendite.artID)>5 and  
count(carrello.artID)>6
```

id	titolo
6	artDTCK2J
7	art4L6WQW
9	artY6L7NE
10	art1W732F
11	art2LTZEY
12	artR50VV4
13	artCBFELX
15	artYUPOAL
16	artQD6K1Q
20	artR1JFGC
24	artRNHR5Y
25	artJA28RV
26	artSWUG72
28	artAJ3XW6
29	art6NMKNZ
31	artLSG229
32	artKW2H5I
34	artDLS52I
35	artL58MKW
36	art5CM0IY
37	artNL5YZ7
39	artN72D71
40	artIQQ40K
41	artCTYY77
45	art76VDU5

Mostro le righe 0 - 24 (8844 del totale, La query ha impiegato (5.33 sec))

Query 15:

selezionare il nome di tutte le categorieArt in cui vi sono almeno 3 art

```
select categorieArt.nome from categorieArt,art,artInCat where art.id = artInCat.artID and artInCat.categoriaID = categorieArt.id GROUP by categorieArt.id having count(art.id) >3;
```

Mostro le righe 0 - 24 (26 del totale, La query ha impiegato (11.86 sec))

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	categorieArt	NULL	index	PRIMARY	PRIMARY	4	NULL	26	100.00	NULL
1	SIMPLE	artInCat	NULL	ref	PRIMARY,artID	PRIMARY	4	society6.categorieArt.id	37954	100.00	Using index
1	SIMPLE	art	NULL	eq_ref	PRIMARY	PRIMARY	8	society6.artInCat.artID	1	100.00	Using index

```
select straight_join categorieArt.nome from artInCat,categorieArt,art where art.id = artInCat.artID and artInCat.categoriaID = categorieArt.id GROUP by categorieArt.id having count(art.id) >3;
```

Mostro le righe 0 - 24 (26 del totale, La query ha impiegato (2.89 sec))

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	artInCat	NULL	index	PRIMARY,artID	artID	8	NULL	948868	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	categorieArt	NULL	eq_ref	PRIMARY	PRIMARY	4	society6.artInCat.categoriaID	1	100.00	NULL
1	SIMPLE	art	NULL	eq_ref	PRIMARY	PRIMARY	8	society6.artInCat.artID	1	100.00	Using index

Questa query grazie all'uso dello strict join è stata migliorata del circa 70% nello specifico si preferito visitare prima la tabella artInCat che come possiamo vedere è grande poi categorieArt e infine art

nome
categoria 1
categoria 2
categoria 3
categoria 4
categoria 5
categoria 6
categoria 7
categoria 8
categoria 9
categoria 10
categoria 11
categoria 12
categoria 13
categoria 14
categoria 15
categoria 16
categoria 17
categoria 18
categoria 19
categoria 20
categoria 21
categoria 22
categoria 23
categoria 24
categoria 25

Query 15:

selezionare il nome della categoria ,id categoria, e quante art ci sono nella categoria e il titolo delle art nella categoria

select

```

    categorieArt.id,categorieArt.nome,count(art.id),group_concat(art.titolo
) from categorieArt,art,artInCat where art.id = artInCat.artID and
artInCat.categoriaID = categorieArt.id GROUP BY categorieArt.id;

```

Mostro le righe 0 - 24 (26 del totale, La query ha impiegato (12.39 sec))

id	nome	count(article_id)	group_concat(article.titulo)
1	categoria 1	36001	art1W732F,artCBFELX,artYUPOAL,artWOHUD2,artYZQBCJ...
2	categoria 2	36209	art4L6WQW,artR50V4,artCBFELX,artAI5YJ9,artSWUG72,...
3	categoria 3	36076	artFMPPLMR,artTLOWS,art362Y00,artYZQBCJ,artLSG299,...
4	categoria 4	36424	artRVYZMF,artAI5YJ9,artWOHUD2,artYZQBCJ,artSWUG72,...
5	categoria 5	36242	art2LTZEY,artWOHUD2,artYZQBCJ,artAP2R56,artSWUG72,...
6	categoria 6	36160	ceci ne pas une pipe,artCBFELX,artAP2R56,artRNHR5Y,...
7	categoria 7	36068	artFMPPLMR,artAI5YJ9,artR1JFGC,artAJ3XW6,artMQFUNK,...
8	categoria 8	36381	art1W732F,artCBFELX,artQD6K1Q,artR1JFGC,artWOHUD2,...
9	categoria 9	36184	ceci ne pas une pipe,artW3KF72,artY6L7NE,art2LTZEY,...
10	categoria 10	35965	artRVYZMF,artY6L7NE,artCBFELX,artAI5YJ9,artJA28RV,...
11	categoria 11	36355	art362Y00,art4L6WQW,artCBFELX,artAI5YJ9,artQD6K1Q,...
12	categoria 12	35895	artYUPOAL,artAP2R56,art6NMKNZ,artMQFUNK,artKW2H5I,...
13	categoria 13	36119	ceci ne pas une pipe,art1W732F,artYZQBCJ,artAJ3XW6,...
14	categoria 14	36355	artTLOWS,artY6L7NE,art2LTZEY,artR50V4,artCBFELX,...
15	categoria 15	36346	artRVYZMF,art1W732F,artYUPOAL,artYZQBCJ,artRNHR5Y,...
16	categoria 16	36286	artW3KF72,artFHS0UF,artSWUG72,art6NMKNZ,artMQFUNK,...
17	categoria 17	36282	artFHS0UF,artKW2H5I,artDLS52I,artNL5YZ7,artCTYY77,...
18	categoria 18	36404	ceci ne pas une pipe,art362Y00,art1W732F,art2LTZEY,...
19	categoria 19	36365	artTLOWS,artW3KF72,artVVY2HD,artFHS0UF,artWOHUD2,...
20	categoria 20	36212	artVVY2HD,artR1JFGC,artYZQBCJ,artAP2R56,artAJ3XW6,...
21	categoria 21	36320	art1W732F,artCBFELX,artFHS0UF,artR1JFGC,artAP2R56,...
22	categoria 22	36250	ceci ne pas une pipe,art2LTZEY,artAI5YJ9,artR1JFGC,...
23	categoria 23	36288	art4L6WQW,art1W732F,artVVY2HD,artJNHN3K,artAJ3XW6,...
24	categoria 24	36233	art362Y00,artW3KF72,artVVY2HD,artJNHN3K,artAJ3XW6,...
25	categoria 25	36033	art4L6WQW,artY6L7NE,artCBFELX,artYZQBCJ,artJNHN3K,...

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	categorieArt	NULL	index	PRIMARY	PRIMARY	4	NULL	26	100.00	NULL
1	SIMPLE	artInCat	NULL	ref	PRIMARY,artID	PRIMARY	4	society6.categorieArt.id	37954	100.00	Using index
1	SIMPLE	art	NULL	eq_ref	PRIMARY	PRIMARY	8	society6.artInCat.artID	1	100.00	NULL

```
select straight join
```

```
categorieArt.id,categorieArt.nome,count(art.id),group_concat(art.titolo  
 ) from artInCat,categorieArt,art where art.id = artInCat.artID and  
 artInCat.categoriaID = categorieArt.id GROUP by categorieArt.id;
```

26 rows in set, 26 warnings (5.95 sec)

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra	
1	SIMPLE	artInCat	NULL	index	PRIMARY	artID	8		NULL	948868	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	categorieArt	NULL	eq_ref	PRIMARY	PRIMARY	4	society6.artInCat.categorieID	1	100.00	NULL	
1	SIMPLE	art	NULL	eq_ref	PRIMARY	PRIMARY	8	society6.artInCat.artID	1	100.00	NULL	

Query 16:

selezionare tutti gli utenti seguiti da uno specifico utente

```
SELECT straight_join kk.nome,NOW() from utenti as  
kk,utenti,segue where kk.nome != utenti.nome and kk.id=  
segue.utentel and utenti.id = segue.utente2 and utenti.nome =  
'Adelina0';
```

Mostro le righe 0 - 24 (103259 del totale, La query ha impiegato 47.73 sec.)

nome
Aaron0
Aaron1
Aaron10
Aaron11
Aaron12
Aaron13
Aaron14
Aaron15
Aaron16
Aaron17
Aaron18
Aaron19
Aaron2
Aaron3
Aaron4
Aaron5
Aaron6
Aaron7
Aaron8
Aaron9
Abbey0
Abbey1
Abbey10
Abbey11
Abbey12

Query 17:

selezionare i followers di uno specifico utente

```
select utenti.nome from utenti,segue, utenti as kk where  
kk.nome != utenti.nome and kk.id= segue.utentel and utenti.id =  
segue.utente2 and kk.nome = 'Adelina0';
```

Mostro le righe 0 - 24 803 rows in set (0.03 sec)

Query 18:

selezionare tutte le art che hanno il nome del tag che contiene 'PORC'

```
select art.titolo,tag.nome from art,tag where art.id = tag.artID  
and tag.nome LIKE '%PORC%';
```

Mostro le righe 0 - 2 (3 del totale, La query ha impiegato (0.27 sec).)

titolo	nome
artBRNHX7	VPORCW
artE1E5IU	PORCBV
artNG9HJC	HPORCM

Query 19:

selezionare il titolo di tutte le categorie,art e tutte le collection con almeno una art in comune con titolo dell'art' che contiene la parola 'artx' e che è stata venduta di meno di 7 volte e quante volte ha venduto

```
select art.titolo, count(art.id), collection.titolo, categorieArt.nome
from compravendite, collection, categorieArt, art_collection, artInCat, art
where
categorieArt.id = artInCat.categoriaID and collection.id =
art_collection.collectionID and art.id = artInCat.artID and art.id =
art_collection.artID
and art.titolo like '%artx%' and compravendite.artID = art.id GROUP by
art.id having count(art.id) < 7;
```

Mostro le righe 0 - 24 (266 del totale, La query ha impiegato 0.5446 secondi.)

titolo	count(art.id)	titolo	nome
artXSLNGV	6	collection numero1224	categoria 9
artX4OBAO	2	collection numero7166	categoria 15
artX7NPMK	4	collection numero12823	categoria 12
artXKUQHV	4	collection numero13097	categoria 2
artXKXXKY3	4	collection numero2179	categoria 2
artX4K5OE	3	collection numero4074	categoria 24
artXQJ6XO	3	collection numero8132	categoria 25
artXSDFEB	2	collection numero10419	categoria 24
artXOZVVS	4	collection numero6340	categoria 8
artXT4Z2Y	2	collection numero4403	categoria 16
artXTPC6G	6	collection numero5377	categoria 24
artX4C36C	1	collection numero4483	categoria 12
artXSZUZ5	2	collection numero10332	categoria 11
artXHZY1M	6	collection numero14360	categoria 2
artXGW2AZ	6	collection numero1486	categoria 3
artXYYONV	6	collection numero14567	categoria 9
artXVD3FU	6	collection numero5092	categoria 4
artXSPU9R	2	collection numero13160	categoria 23
artXBPCPV	3	collection numero5565	categoria 21
artXCGZQI	3	collection numero12049	categoria 13
artXHJASN	3	collection numero11208	categoria 21
artXZDDXM	4	collection numero7486	categoria 23
artXO14F4	3	collection numero11892	categoria 1
artXLENGQ	4	collection numero8853	categoria 6
artXKM8RK	2	collection numero6523	categoria 26

Query 20:

selezionare nomi degli oggetti e per ognuno determinare quante volte è stato venduto

```
select oggetto.nome , count(oggetto.id)
from oggetto, compravendite where
oggetto.id = compravendite.oggettoID
GROUP by(oggetto.id);
```

nome	count(oggetto.id)
cover iphone	9732
maglietta uomo	9726
cover samsung	9725
cover laptop	9930
maglietta uomo	9622
leggins	9784
wall art	9684
canvas print	9807
tappetino	9663
tenda da doccia	9668
tavolo	9538
sgabello	9725
coffe mug	9679
borraccia	9701
zaino	9663
borsa	9899
piumone	9798
cuscino quadrato	9759
adesivo	9742
adesivo per pc	9752
tenda per finestra	9735

Mostro le righe 0 - 20 (21 del totale, La query ha impiegato 0.1462 secondi.)

Query 21:

selezionare il nomi degli **oggetti** e il colore con i quali sono stati venduti e quanti oggetti sono stati venduti con quel colore

```
select oggetto.nome,colore.colore, count(colore.colore) from
compravendite,colore,oggetto where oggetto.id = compravendite.oggettoID
and colore.id = compravendite.coloreID GROUP by oggetto.id,colore.id;
```

nome	colore	count(colore.colore)
cover iphone	unico	9732
maglietta uomo	unico	9726
cover samsung	unico	9725
cover laptop	unico	9930
maglietta uomo	rosso	1877
maglietta uomo	nero	1891
maglietta uomo	bianco	1919
maglietta uomo	verde	1949
maglietta uomo	giallo	1986
leggins	rosso	1931
leggins	nero	1957
leggins	bianco	1977
leggins	verde	1918
leggins	giallo	2001
wall art	unico	9684
canvas print	unico	9807
tappetino	rosso	1898
tappetino	nero	1917
tappetino	bianco	1980
tappetino	verde	1947
tappetino	giallo	1921
tenda da doccia	unico	9668
tavolo	rosso	1956
tavolo	nero	1900
tavolo	bianco	1813

Mostro le righe 0 - 24 (41 del totale, La query ha impiegato 0.2384 secondi.)

Query 22:

selezionare il nome dell'**'oggetto'** e il colore con il quale è stato venduto di più e quanti oggetti sono stati venduti con quel colore

```
select tab.ogg, max( tab.kk ), tab.cc from
(select oggetto.nome as ogg,colore.colore as cc, count(colore.colore)
as kk from compravendite,colore,oggetto where oggetto.id =
```

```

compravendite.ogettoID and colore.id = compravendite.coloreID GROUP by
oggetto.id,colore.id order by kk desc)as tab
GROUP by tab.ogg;

```

ogg	max(tab.kk)	cc
adesivo	9742	unico
adesivo per pc	9752	unico
borraccia	9701	unico
borsa	9899	unico
canvas print	9807	unico
coffe mug	1981	bianco
cover iphone	9732	unico
cover laptop	9930	unico
cover samsung	9725	unico
cuscino quadrato	9759	unico
leggins	2001	giallo
maglietta uomo	9726	unico
piumone	9798	unico
sgabello	9725	unico
tappetino	1980	bianco
tavolo	1956	rosso
tenda da doccia	9668	unico
tenda per finestra	9735	unico
wall art	9684	unico
zaino	9663	unico

Mostro le righe 0 - 19 (20 del totale, La query ha impiegato 0.2357 secondi.)

Interrogazione Algebra Relazionale

Algebra Relazionale

L' algebra relazionale è un linguaggio di interrogazione di basi di dati.

E' un linguaggio procedurale in cui le operazioni complesse vengono specificate descrivendo il procedimento da seguire per ottenere la soluzione.

L'algebra relazionale è basata su concetti di tipo algebrico.

Sostanzialmente questo linguaggio è costituito da un insieme di operatori, definiti su relazioni, che producono ancora relazioni come risultati

Query Algebra Relazionale

Selezionare NOME degli UTENTI e quanto hanno speso in totale

Query1:

```
π utente.nome, sum(taglia_oggetto.prezzo) (
    σ ( UTENTI ⋙ utente.id = compravendite.compratore COMPRAVENDITE
        ⋙ compravendite.oggettoid = taglia_oggetto.oggettoid and compravendite.tagliaID
        = taglia_oggetto.tagliaID TAGLIA_OGGETTO
    )
)
```

Query2:

Selezionare NOME degli UTENTI e il titolo di tutte le art rispettive.

```
π utente.nome, art.titolo(
    σ ( UTENTI ⋙ utenti.id = art.utenteID ART
    )
)
```


MongoDB

MongoDB è un DBMS non relazionale, orientato ai documenti.

E' classificato come un database di tipo NoSQL poiché si allontana dalla struttura tradizionale basata sulle tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico, rendendo l'integrazione di alcuni tipi di applicazioni più facili e veloci.

Tra i punti chiave di MongoDB troviamo:

- Query ad hoc
- Indicizzazione
- Alta affidabilità
- Sharding e bilanciamento dei dati
- File storage
- Aggregazione
- Capped collection

Creazione DataBase Collection

Utilizzando un CMD entriamo in Mongo e tramite il comando USE creiamo il nostro DataBase per MongoDB e allo stesso tempo Mongo effettua il cambio di database su questo appena creato.

```
> use society6
switched to db society6
> []
```

MongoDB immagazzina tutti i documenti in delle Collection. Le Collezioni sono l'analogo delle Tabelle nei DataBase Relazionali. Per creare la nostra Collection scegliamo il Ramo del nostro DataBase di MySQL dove sono presenti le Tabelle :

- UTENTI
- SEGUE

Sono state scelte queste due tabelle in quanto sono quelle che contengono più record circa 119104360 si vuole quindi testare come i due tipi di database reagiscono a una mole così grande di dati.

```

SELECT utenti.id,utenti.nome,utenti.email,utenti.password,segueU.nome
as segue FROM utenti,utenti as segueU,segue where segue.utente1 =
utenti.id and segue.utente2 = segueU.id into outfile
'/mnt/c/Users/naddi/Documents/database society6/utentiSegue.csv' fields
enclosed by '"' terminated by ',' lines terminated by '\n'

```

Attraverso questo(sopra) comando andiamo a esportare le due tabelle nel file utentiSegue.csv
successivamente andremo a importarli in mongoDB con il seguente comando

```
naddi@LAPTOP-J8AVGLI6:/mnt/c/Users/naddi/Documents/database society6$ mongoimport -d society6 -c utentiSegue --type csv --file utentiSegue.csv --fields ["id","nome","email","password","segue"][]
```

Ottenendo infine un Collection chiamata utentiSegue. Effettuando una FIND possiamo ottenere un estratto del contenuto presente nella collection.

La find restituendo tutti i dati della Collection (quindi una quantità di dati abbastanza grande), non ci stampa tutto il risultato, ma si ferma, mostrandoci il comando IT che se digitato, può mostrarcici altri dati.

```

> db.utentiSegue.find().pretty()
{
  "_id" : ObjectId("5ba9da8472309860c992aaa2"),
  "id" : 3,
  "nome" : "caglio",
  "email" : "caglio@as.it",
  "password" : "bsaaba",
  "segue" : "Aaron0"
}
{
  "_id" : ObjectId("5ba9da8472309860c992aaa3"),
  "id" : 3,
  "nome" : "caglio",
  "email" : "caglio@as.it",
  "password" : "bsaaba",
  "segue" : "Adah0"
}
{
  "_id" : ObjectId("5ba9da8472309860c992aaa4"),
  "id" : 3,
  "nome" : "caglio",
  "email" : "caglio@as.it",
  "password" : "bsaaba",
  "segue" : "Adalberto0"
}
{
  "_id" : ObjectId("5ba9da8472309860c992aaa5"),
  "id" : 3,
  "nome" : "caglio",
  "email" : "caglio@as.it",
  "password" : "bsaaba",
  "segue" : "Adaline0"
}
{
  "_id" : ObjectId("5ba9da8472309860c992aaa6"),
  "id" : 3,
  "nome" : "caglio",
  "email" : "caglio@as.it",
  "password" : "bsaaba",
  "segue" : "Abbey0"
}

```

Query MongoDB

Query1:

selezionare tutti gli utenti che sono seguiti da un utente specifico (mongoDB)

a fine di stimarne il tempo la query viene impostata nel seguente modo

```
216  var inizio = new Date()
217  db.utentiSegue.find({nome:'Adam0'}, {_id:0,id:0,email:0,password:0,nome:0})
218  var fine= new Date()
219  var risultato = fine - inizio
220  risultato/1000 + ' secondi'

> var fine= new Date()
> var risultato = fine - inizio
> risultato/1000 + ' secondi'
0.015 secondi
```

come possiamo vedere la stessa query ha impiegato 0.015 secondi andiamo adesso a confrontare la medesima su mysql

```
SELECT kk.nome from utenti as kk,utenti,segue where utenti.id= segue.utente1 and kk.id = segue.utente2 and utenti.nome = 'Adam0';

+-----+
| Charmaine0 |
| Charolette0 |
| Chas0       |
| Chase0      |
| Chasidy0    |
| Chasity0   |
+-----+
803 rows in set (0.01 sec)
```

su mysql query ha impiegato 0.01 risultato simile a mongoDB

Query2:

selezionare i followers di uno specifico utente (mongoDB)

```
203  var inizio = new Date()
204  db.utentiSegue.find({segue:'Adam0'}, {_id:0,id:0,email:0,password:0,segue:0})
205  var fine= new Date()
206  var risultato = fine - inizio
207  risultato/1000

> var fine= new Date()
> var risultato = fine - inizio
> risultato/1000
0.035
```

come possiamo vedere la query ha impiegato 0.035 secondi andiamo adesso a confrontare la medesima su mysql

```
188 SELECT straight_join kk.nome FROM utenti AS kk,utenti,segue WHERE kk.nome != utenti.nome AND kk.id = segue.utente1 AND utenti.id = segue.utente2 AND utenti.nome = 'Adamò';
```

```
Zulma14  
Zulma15  
Zulma16  
Zulma17  
Zulma18  
Zulma2  
Zulma3  
Zulma4  
Zulma5  
Zulma6  
Zulma7  
Zulma8  
| Zulma9  
+-----+  
103258 rows in set (20.26 sec)  
  
mysql> |
```

su mysql query ha impiegato 20.26 risultato molto differente rispetto a mongoDB (più veloce di circa 20 secondi) che risulta più adatto al fine di gestire basi di dati di grandi dimensioni.