**ItalyComics**

# 1. Abstract

“**ItalyComics**” è un software per l’organizzazione delle fiere comics (ad esempio “GamesWeek”, “Romics”, “Comicon”).

Con “**ItalyComics**” è possibile tenere traccia delle diverse fiere e di tutti gli aspetti che le caratterizzano, come l’infrastruttura (padiglioni, stand) e i partecipanti.

I partecipanti dovranno inserire i loro dati personali all’acquisto del biglietto, che potrà avere agevolazioni in base all’età o status (cosplayer, vip), inoltre in questo modo è possibile evitare la rivendita di biglietti in modo illegale e controllare che il numero dei partecipanti sia minore o uguale alla capacità totale della fiera.

Ogni responsabile gestisce uno stand all’interno della fiera, con la possibilità di vendere articoli o semplicemente attirare visitatori interessati a un certo tema come videogiochi o anime.

# 2. Analisi dei requisiti

## 2.1 Descrizione testuale

Nella base di dati sono presenti i dati delle **Fiere** registrate nell’applicazione. Di ogni fiera è noto il suo nome, l’anno in cui si svolge e la città in cui si svolge.

Ogni fiera è strutturato in più **aree**, **esterne** o interne(**padiglioni**), collegate tra loro. Le aree sono collegate tra loro e ogni area è caratterizzata da un codice. In particolare, i padiglioni hanno una capacità massima per garantire la sicurezza dei visitatori.

I padiglioni conterranno molti **stand** caratterizzati da un codice identificativo(id) e un nome.

Lo stand potrà vendere degli **articoli**, quali fumetti, gadget, vestiti, etc; distinti dal numero di serie (S/N) e il nome.

Ogni stand riguarda uno o più **temi**, che possono essere **giochi, film, serie tv, fumetti, anime, manga.** Un tema ha un nome, una data di uscita, e un autore.

La base di dati tiene conto anche di ogni **persona** che visita la fiera comprando un **biglietto.** A seconda della categoria della persona (**bambino, cosplayer, giornalista, studente, staff)**, si può comprare un tipo di biglietto: una persona normale pagherà il biglietto **intero,** i bambini, i cosplayer e gli studenti pagheranno il biglietto **ridotto,** giornalisti e membri dello staff avranno un biglietto **vip,** che potranno usare come pass per accedere a zone non accessibili ad utenti normali.

Di ogni biglietto è noto: il suo codice identificativo, la data in cui è stato emesso, la durata della validità del biglietto, e il prezzo.

I dati di ogni persona permettono di conoscere: il codice fiscale, il nome e cognome, la data di nascita, l’indirizzo, la mail e il numero di telefono. In particolare: di ogni cosplayer è conosciuto il nome del personaggio; di ogni giornalista il nome del giornale per cui lavora; di ogni studente il nome della scuola che frequenta. Un membro dello staff, che può essere un **dipendente** o un **responsabile**, ha un settore di appartenenza e uno stipendio.

## 2.2 Glossario dei termini

| **Termine** | **Descrizione** | **Collegamenti** |
| --- | --- | --- |
| Fiera | Una determinata edizione della fiera | Area, Persona |
| Area | Una specifica area della fiera | Area, Stand |
| Stand | Uno stand che può vendere articoli ai visitatori | Area, Persona, Articolo, Tema |
| Articolo | Un articolo disponibile alla vendita | Stand |
| Tema | Un tema che può essere un gioco, un film, una serie tv, un fumetto, un anime o un manga | Stand |
| Persona | Una persona che ha visitato la fiera | Fiera, Stand, Biglietto |
| Biglietto | Biglietto emesso alla fiera e venduto ad una persona | Persona |

## 2.3 Operazioni

| **Operazione** | **Frequenza** |
| --- | --- |
| Inserimento di una nuova persona | 1.000 al giorno |
| Inserimento nuovo biglietto | 1.000 al giorno |
| Controllo del tipo di biglietto di una persona | 500 al giorno |
| Inserimento nuova fiera | 150 all’anno |
| Inserimento nuovo stand | 100 al mese |
| Inserimento nuovo articolo | 1.000 al mese |
| Inserimento nuovo tema | 50 al mese |

# 

# 3. Progettazione Concettuale

## 3.1 Lista entità

* Fiera
  + Nome varchar (50) PK
  + Anno int PK
  + Città varchar(50)
  + CapienzaTotale int
* Area
  + Codice varchar(4) PK

Area si specializza in:

* + Esterno
  + Paglione
    - CapacitàStand int
    - CapienzaPersone int
* Stand
  + ID varchar(8) PK
  + Nome varchar(50)
  + Guadagno decimal(7,2)
* Articolo
  + S/N varchar(32) PK
  + Nome varchar(50) PK
  + Prezzo decimal(5,2)
* Tema
  + Nome varchar(50) PK
  + DataUscita date PK
  + Autore varchar(50)

Tema si specializza in:

* + Gioco
  + Film
  + Serie TV
  + Fumetto
  + Anime
  + Manga
* Persona
  + CF varchar(16) PK
  + Nome varchar(50)
  + Cognome varchar(50)
  + DataNascita date
  + Mail varchar(30)
  + Telefon o varchar(20)
  + Indirizzo
    - Stato varchar(20)
    - Città varchar(30)
    - CAP int
    - Via varchar(20)
    - N\_Civico int

Persona si può specializzare in:

* + Bambino
  + Cosplayer
    - Personaggio varchar(50)
  + Giornalista
    - Giornale varchar(50)
  + Studente
    - Scuola varchar(50)
  + Staff
    - Settore varchar(50)
    - Stipendio decimal(7,2)

Staff si specializza in:

* + - Responsabile
    - Dipendente
* Biglietto
  + Codice varchar(16) PK
  + Data date
  + Validità date
  + Prezzo decimal(5,2)

Biglietto si specializza in:

* + Ridotto
  + Intero
  + VIP

## 3.2 Tabella delle relazioni

| **Relazione** | **Entità coinvolte** | **Descrizione** | **Attributi** |
| --- | --- | --- | --- |
| Appartenenza | Biglietto(1:1)  Persona (1:1) | Un biglietto è posseduto da una sola persona.  Una persona possiede un solo biglietto | Nessuno |
| Visita | Persona(1:N)  Fiera(1:N) | Una persona può visitare più fiere.  Una fiera è visitata da più persone | Nessuno |
| Composizione | Fiera(1:N)  Area(1:1) | Una fiera è composta da una o più aree.  Un’area fa parte di una sola fiera | Nessuno |
| Collegamento | Area(0:N) | Un’area può essere collegata ad una o più aree. | Nessuno |
| Contenuto | Padiglione(0:N)  Stand(1:1) | Un padiglione può contenere zero o più stand.  Uno stand è contenuto in un solo padiglione | Nessuno |
| Vendita | Stand(0:N)  Articolo(0:N) | Uno stand può vendere zero o più articoli.  Lo stesso articolo può essere in zero o più stand diversi | Quantità: int >=0 |
| Tematica | Stand(1:N)  Tema(1:N) | Uno stand ha 1 o più temi.  Un tema è riguardato da uno o più stand | Nessuno |
| Lavoro | Staff(1:1)  Stand(1:N) | Un membro dello staff lavora in uno solo stand.  In un stand lavorano una o più persone dello staff. | Nessuno |

## 

## 

## 3.3 Schema Concettuale

# 4. Progettazione Logica

## 4.1 Ristrutturazione

### 4.1.1 Analisi delle ridondanze

L’attributo “**guadagno**” dell’entità Stand è ridondante in quanto può essere calcolato sommando i prezzi degli articoli venduti.

Questo attributo occuperebbe spazio in memoria senza portare alcun beneficio, con l’aggiunta del rischio di inconsistenza, quindi verrà eliminato.

L’attributo “**CapienzaTotale**”dell’entità Fiera è ridondante in quanto può essere calcolato facendo la somma delle capienze dei padiglioni.

| **Concetto** | **Tipo** | **Volume** |
| --- | --- | --- |
| Fiera | E | 5.000 |
| Padiglione | E | 100.000 |
| Composizione | R | 100.000 |

* OPERAZIONE 1: Inserimento nuova Padiglione (100 volte all’anno)
* OPERAZIONE 2: Visualizzazione capienza totale di una fiera (5.000 volte al giorno)

**CON RIDONDANZA:**

* OPERAZIONE 1:

Una fiera ha in media 15 padiglioni

| **Concetto** | **Costrutto** | **Accessi** | **Tipo** | **Frequenza** |
| --- | --- | --- | --- | --- |
| Fiera | E | 1 | S | x100 (anno) |
| Composizione | R | 15 | L | x100 (anno) |
| Composizione | R | 1 | S | x100 (anno) |

* OPERAZIONE 2:

| **Concetto** | **Costrutto** | **Accessi** | **Tipo** | **Frequenza** |
| --- | --- | --- | --- | --- |
| Fiera | E | 1 | S | x5.000 (giorno) |

Supponendo che gli accessi in scrittura costino il doppio:

Costo annuo: 100\*2 + 100\*15 + 100\*2 + 5.000\*2\*365 = 3.651.900

**SENZA RIDONDANZA:**

* OPERAZIONE 1:

| **Concetto** | **Costrutto** | **Accessi** | **Tipo** | **Frequenza** |
| --- | --- | --- | --- | --- |
| Composizione | R | 1 | S | x100 (anno) |

* OPERAZIONE 2:

| **Concetto** | **Costrutto** | **Accessi** | **Tipo** | **Frequenza** |
| --- | --- | --- | --- | --- |
| Fiera | E | 15 | S | x5.000 (giorno) |

Supponendo che gli accessi in scrittura costino il doppio:

Costo annuo: 100\*2 + 5.000\*15\*2\*365 = 54.750.200

In conclusione l’analisi della ridondanza dimostra che conviene mantenere l’attributo “CapienzaMassima”.

### 4.1.2 Eliminazione delle generalizzazioni

| **Generalizzazione** | **Risoluzione** |
| --- | --- |
| Biglietto ← Intero, Ridotto, Vip | Le entità intero, ridotto e vip vengono accorpate in biglietto perché ci possono essere prezzi diversi in base alla fascia di età. |
| Staff ← Responsabile, Dipendente | Le entità Responsabile e Dipendente vengono accorpate in Staff perché rappresentano lo stesso concetto, viene aggiunto un attributo opzionale su staff:   * Staff: bool |
| Persona ⇐ Bambino, Cosplayer, Studente, Giornalista, Staff | Le entità Bambino, Cosplayer, Studente, Giornalista e Staff vengono accorpate in Persona perché rappresentano lo stesso concetto, vengono aggiunti due attributi a Persona:   * Tipo: varchar(20), NOT NULL   Gli attributi giornale, staff, scuola e personaggio diventano opzionali.  La relazione Lavoro ora coinvolge le entità Persona (0:1) e Stand (1:N) |
| Tema ← Gioco, Film, Serie tv, Fumetto, Anime, Manga | Le entità Gioco, Film, Serie tv, Fumetto, Manga e Anime vengono accorpate in Tema perché rappresentano lo stesso concetto. |
| Area ← Esterno, Padiglione | Le entità Esterno e Padiglione vengono accorpate in Area poiché rappresentano lo stesso concetto.  Viene aggiunto l’attributo:   * Esterno: bool, NOT NULL   Gli attributi CapienzaPersone e CapacitàStand diventano opzionali.  La relazione Contenuto ora coinvolge Area(0:N) e Stand(1:1) |

### 4.1.3 Scelta degli identificatori primari

Nell’entità “**Stand**” la chiave è composta da un attributo “id” assieme alla chiave esterna di “Padiglione”. Facendo un upgrade di Id, si è scelto di usarlo come unico identificatore primario. Id rappresentava solo l’ordine dello stand all’interno del padiglione, ora identifica univocamente ogni stand nel DB.

### 

### 

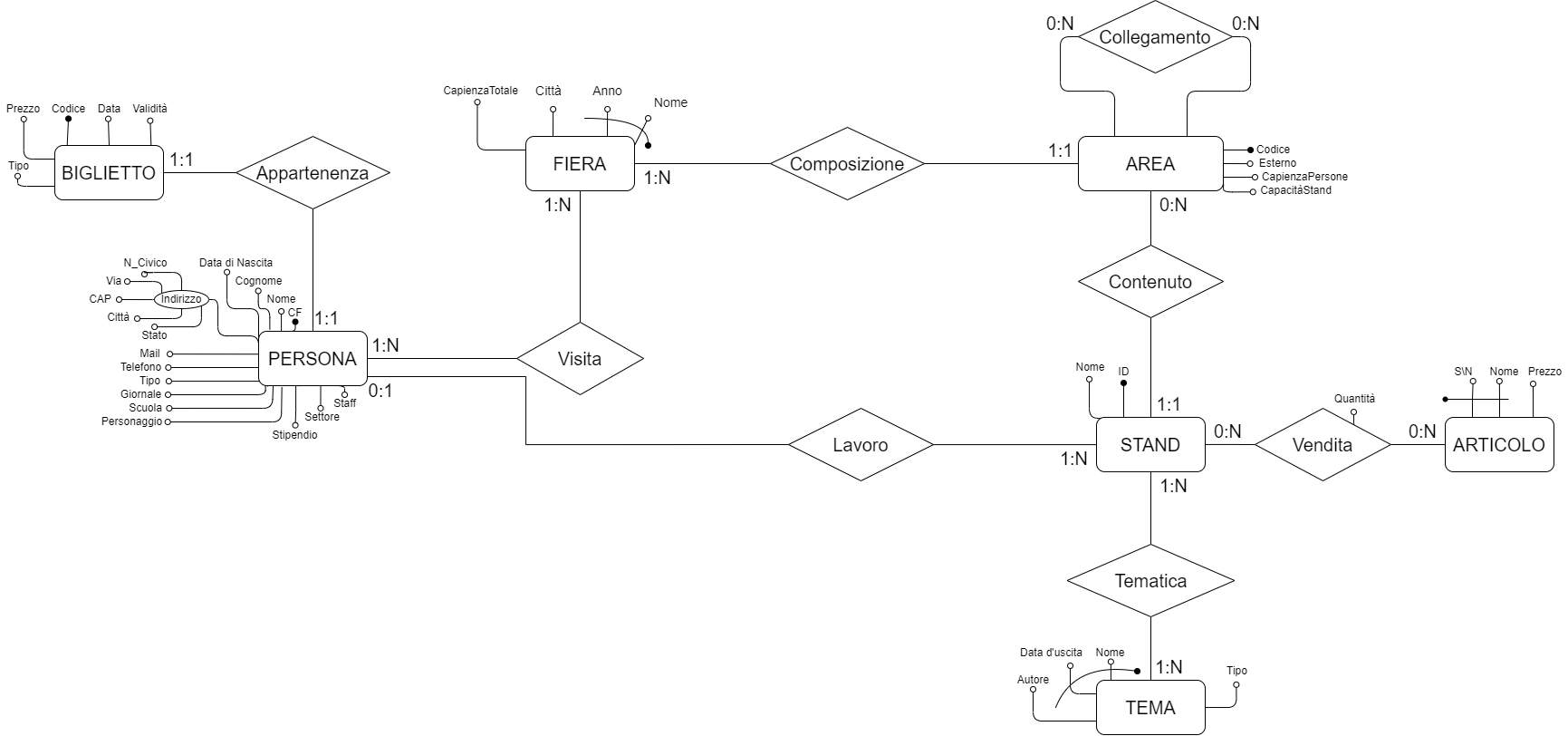
### 

### 

### 

### 

### 4.1.4 Schema E-R ristrutturato



## 4.2 Creazione delle tabelle

(A→B indica che B è chiave esterna di A)

**Fiera** (Anno, Nome, Città, CapienzaTotale)

**Area** (Codice, Esterno, CapienzaPersone, CapacitàStand, Fiera, Anno)

* Fiera, Anno → Fiera.Nome, Fiera.Anno

**Collegamento** (Area1, Area2)

* Area1 → Area.Codice
* Area2 → Area.Codice

**Stand** (ID, Nome, Area)

* Area → Area.Codice

**Articolo** (S/N, Nome, Prezzo)

**Vendita** (Stand, Articolo, Nome, Quantità)

* Stand → Stand.ID
* Articolo, Nome → Articolo.S/N, Articolo.Nome

**Tematica** (Stand, Tema, Data)

* Stand → Stand.ID
* Tema, Data → Tema.Nome, Tema.DataUscita

**Tema** (Nome, DataUscita, Autore, Tipo)

**Visita** (Persona, Fiera, Anno)

* Persona → Pesrona.CF
* Fiera, Anno → Fiera.Nome, Fiera.Anno

**Persona** (CF, Nome, Cognome, DataNascita, Stato, Città, CAP, Via, N\_Civico, Mail, Telefono, Tipo, Giornale, Scuola, Personaggio, Settore, Stipendio, Staff, Biglietto)

* Biglietto → Biglietto.Codice

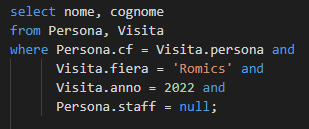
**Biglietto** (Codice, Data, Validità, Prezzo, Tipo)

# 5. Query e Indici

## 5.1 Query

1. Trova le persone che hanno visitato una determinata Fiera (solo visitatori)

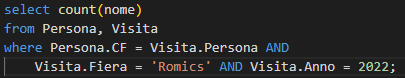
Es: Romics 2022

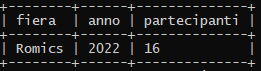




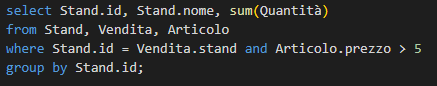
1. Conta quante persone hanno partecipato ad una determinata Fiera (anche staff)

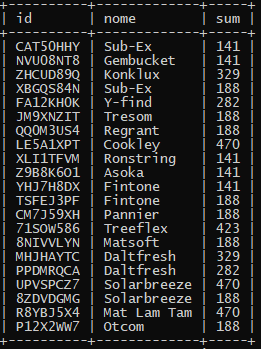
Es: Romics 2022



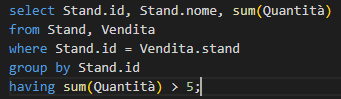


1. Trova quanti articoli ha venduto ogni stand con prezzo maggiore di 5€





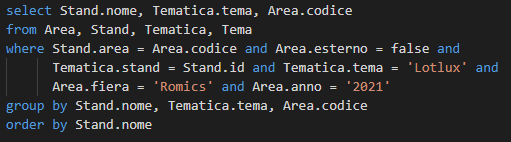
1. Trova quali stand hanno venduto più di 5 articoli

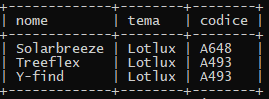




1. Mostra gli stand che appartengono ad un padiglione di una determinata fiera e che hanno una certa tematica

Es: Romics 2021, Lotlux





## 5.2 Indici

Questo DB è utilizzato su larga scala, è quindi necessaria la valutazione di inserimento di indici.

I nomi dei temi vengono letti molto, essendo che ogni visitatore cercherà lo stand che più lo appassioni. Ciò richiede una certa velocità nella ricerca all’interno di questa relazione. Da queste constatazioni ricaviamo l’indicizzazione della tabella “Tema” sugli attributi “Nome” e “DataUscita”.

create index idx\_Temi on Tema(Nome, DataUscita)

# 6. Codice C++

## 6.1 Descrizione dell’utilizzo del codice

Il codice C++ per l’esecuzione delle query consiste in un unico file .cpp, che va compilato attraverso il comando:

g++ main.cpp -L dependencies/lib -l pq -o Queries

Prima di poter compilare il file è necessario verificare la presenza dei file **libpq.dll** e **libpq.lib** in “./dependencies/lib” e **libpq-fe.h**, **pg\_consig\_ext.h**, **postgres\_ext.h** in “./dependencies/include”, dove “./” è il percorso della directory che contiene il file .cpp.

Per eseguire il codice, basta avviare l’eseguibile “Queries”.

La prima cosa che il programma chiede sono il nome del database e la password per accedere, successivamente, se si riesce ad accedere correttamente al database, mostrerà a schermo la lista delle query, identificate da un numero da 1 a 5, eseguibili all’interno del programma.

Per eseguire una query bisogna inserire da tastiera il numero della query scelta, mentre per terminare l’esecuzione del programma va inserito ‘0’. Alcune query (la numero 1, 2 e 5) richiedono l’inserimento di alcuni parametri da parte dell’utente:

* 1 e 2: Viene mostrata prima la lista di tutte le Fiere da cui poter scegliere, poi la lista di tutti gli anni disponibili nel database per quella fiera.
* 5: Oltre alla lista della fiera e degli anni disponibili, viene inoltre mostrata la lista dei temi presenti nel database da cui poter scegliere

## 6.2 Documentazione del codice

Funzioni utilizzate dal codice:

PGconn\* connect(const char\* host, const char\* user, const char\* db, const char\* pass, const char\* port)

Ritorna una connessione al database, utilizzando i parametri passati come credenziali di accesso. Se la connessione non va a buon fine mostra un messaggio di errore e termina il programma.

PGresult\* execute(PGconn\* conn, const char\* query)

Esegue una query passata come stringa ritornando il risultato. Se l’esecuzione non va a buon fine mostra un messaggio di errore e termina il programma.

void printQuery(PGresult\* res)

Stampa a schermo sotto forma di tabella il risultato res di una query. La funzione è in grado di gestire la dimensione delle colonne automaticamente.

void printLine(int campi, int\* maxChar)

Funzione ausiliaria di printQuery che stampa una riga di separazione per la tabella.

char\* chooseFiera(PGconn\* conn)

Viene utilizzata per mostrare un elenco di Fiere da una query per poter sceglierne una.

char\* chooseAnno(PGconn\* conn)

Viene utilizzata per mostrare un elenco di Anni da una query per poter sceglierne uno.

char\* chooseTema(PGconn\* conn)

Viene utilizzata per mostrare un elenco di Temi da una query per poter sceglierne uno.