1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

Progetto di Basi di Dati

Gestore di una base di dati di un negozio di videogiochi

Alessio Muzi mat. n°299329 | Informatica Applicata | A.S. 2021/2022

1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

Analisi dei requisiti

Il campo d'interesse preso in considerazione per lo sviluppo del progetto è un gestore delle varie transazioni e servizi forniti da un **negozio di videogiochi**. Ogni cliente, oltre ad effettuare acquisti di prodotti del negozio, ha la possibilità di sottoscrivere una tessera fedeltà. La tessera offre dei vantaggi come sconti su giochi e console. La tessera è divisa in 4 livelli: il livello 0 è gratis si ottiene al momento della registrazione, l'aumento di livello è a pagamento, più si aumenta di livello e più le offerte sono vantaggiose. Il cliente possessore di una tessera ha la possibilità di portare in negozio i propri giochi e console usate le quali sono sottoposte a controlli dell'integrità (su una scala da 0 a 10) che, se positivi, permettono che vengano ritirate e rivendute. Ad ogni prodotto usato viene dato un codice ed altre informazioni tra l'integrità. Il cliente può prenotare giochi e console prima dell'uscita effettiva dalla Software House solo se possiede una tessera. Inoltre l'amministrazione che gestisce il negozio si occupa di rilasciare documenti fiscali successivamente a una transazione e di organizzare eventi.

Ulteriori specifiche:

- Permettere di gestire le generalità dei clienti e delle Software House.
- Permettere di gestire i prodotti in vendita, sia nuovi che usati.
- Permettere di gestire le prenotazioni dei prodotti.
- Permettere di gestire la programmazione degli eventi.
- Permettere di gestire le tessere fedeltà e le offerte sui prodotti.
- Permettere di gestire lo staff del negozio.
- Rilasciare un documento fiscale quando avviene una transazione.
- Visualizzare il catalogo dei prodotti filtrato o ordinato secondo parametro\i.
- Autorizzare le prenotazioni esclusivamente a chi possiede una tessera fedeltà.

Ambiguità e correzioni:

- "Offerte" e "Sconti" sono sinonimi sostituiti con "Offerte".
- "Tessera" e "Carta" sono sinonimi sostituiti con "Tessera fedeltà".
- "Accessori" e "Gadget" sono sinonimi sostiuiti con "Gadget".
- "Staff" e "Amministrazione" sono sinonimi sostituiti con "Amministrazione".
- Per "Prodotti" si intende sia videogiochi che console in vendita, ma più in generale un qualsiasi oggetto venduto dal negozio (potrebbero essere anche gadget).
- Per "Eventi" si intende una qualsiasi manifestazione organizzata nel negozio con la partecipazione di qualche personalità (VIP) importante.

Glossario dei termini

Cliente: persona che fisicamente si reca nel negozio per effettuare l'acquisto di un prodotto. Può essere un cliente generico o un cliente registrato se possiede o meno una tessera fedeltà.

Amministrazione: proprietario (singolo o gruppo di persone) del negozio che gestisce l'intero sistema.

Prodotto: un qualsiasi oggetto venduto dal negozio, comprende videogiochi e console, sia nuove che usate, ma anche gadget.

Software House: azienda specializzata nello sviluppo di videogiochi, la quale rifornisce il negozio di prodotti da commercializzare.

Tessera fedeltà: permette al cliente che la possiede di poter preordinare i videogiochi, accedere a offerte e riportare console usate.

Offerte: un qualsiasi tipo di sconti e promozioni su prodotti forniti dal negozio.

Videogioco: il prodotto delle Software House, l'oggetto principalmente venduto dal negozio di videogiochi.

Console: un qualsiasi hardware specifico che permette di potere giocare un qualsiasi videogioco.

Gadget: un qualsiasi accessorio legato al mondo dei videogiochi venduto nel negozio.

Catalogo: il catalogo è l'elenco di tutti i prodotti in vendita nel negozio.

Documento fiscale: documento rilasciato al cliente che include importo e data di acquisito (ricevuta/fattura).

Evento: qualsiasi manifestazione organizzata dall'amministrazione del negozio che abbia un tema e richieda la partecipazione di un VIP legato al mondo dei videogiochi.

Prenotazione: se un cliente possiede una tessera fedeltà può lasciare una somma di soldi in negozio e prenotare il gioco prima della sua effettiva data di uscita.

Controllo dell'integrità: quando un cliente con tessera fedeltà porta un gioco o una console usata in negozio per poterla rivendere, vengono effettuati dei controlli dai dipendenti del negozio per assicurarsi che sia in condizioni accettabili.

Operazioni

Aggiornamenti:

1. Registrare un cliente.
2. Registrare una tessera fedeltà.
3. Registrare un prodotto.
4. Modificare gli attributi di un prodotto.
5. Creare un evento.
6. Creare un'offerta.
7. Eliminare un prodotto.
8. Rilasciare un documento fiscale.
9. Gestire i componenti dell'amministrazione.

Interrogazioni:

1. Visualizzare il catalogo (eventualmente ordinato e/o filtrato).
2. Visualizzare i clienti (divisi in generici e registrati).
3. Visualizzare solamente i videogiochi, le console o i gadget.
4. Visualizzare gli eventi.
5. Dato un videogioco, visualizzare la software house.
6. Visualizzare il prodotto più costoso.
7. Visualizzare le tessere rilasciate e i clienti a cui sono associate.
8. Visualizzare i documenti fiscali rilasciati in un dato giorno.
9. Dato un videogioco, visualizzare le offerte associate a una tessera fedeltà.

Progettazione concettuale

La **progettazione concettuale** di una Base di Dati richede di individuare le **entità** che la costituiscono e le **relazioni** (o associazioni) tra entità. Il modello concettuale usato è il **modello Entità-Relazione** (**E/R**). Di seguito sono descritte le entità necessarie per soddisfare la specifica analizzata nella sezione precedente:

Cliente:
Le entità associate sono Prodotto, Documento fiscale ed Evento
Il cliente può essere di due tipi:

- **Generico** (sottocategoria di Cliente):
  - IdCliente
  - CodFiscale

Le entità associate sono le stesse di Cliente

- **Registrato** (sottocategoria di Cliente):
  - IdCliente
  - CodFiscale
  - Nome
  - Cognome
  - Email
  - DataNascita
  - Indirizzo

Le entità associate sono le stesse di Cliente + Tessera fedeltà

Prodotto:
Le entità associate sono Cliente (Registrato) e Documento fiscale
Il prodotto può essere di tre tipi:

- **Console** (sottocategoria di Prodotto):
  - CodProd
  - Integrità
  - Prezzo
  - AnnoProduzione
  - Produttore
  - Modello

Le entità associate sono le stesse di Prodotto

- **Videogioco** (sottocategoria di Prodotto):
  - CodProd
  - Integrità
  - Prezzo
  - Piattaforma
  - Genere
  - DataUscita

Le entità associate sono le stesse di Prodotto + Software House

- **Gadget** (sottocategoria di Prodotto):
  - CodProd
  - Integrità
  - Prezzo
  - Tipo

Le entità associate sono le stesse di Prodotto

Evento:

- IdEvento
- Data
- Genere
- VIP

Le entità associate sono Cliente e Amministrazione

Software House:

- CodSH
- Nazionalità
- Indirizzo
  - Via
  - Numero Civico
  - CAP

L'unica entità associata è Videogioco

Offerta:

- CodOfferta
- Tipo
- DataInizio

- DataFine

Le entità associate sono Amministrazione e Tessera fedeltà

Amministrazione:

- CodFiscale
- Incarico
- Nome
- Cognome
- Email

Le entità associate sono Offerta ed Evento

Tessera fedeltà:

- IdTessera
- DataCreazione
- DataScadenza
- Livello

Le entità associate sono (Cliente) Registrato e Offerta

Documento fiscale:

- CodTransazione
- Importo
- Data

Le entità associate sono Cliente e Prodotto

Segue l'elenco dei **vincoli** necessari per rispettare la specifica:

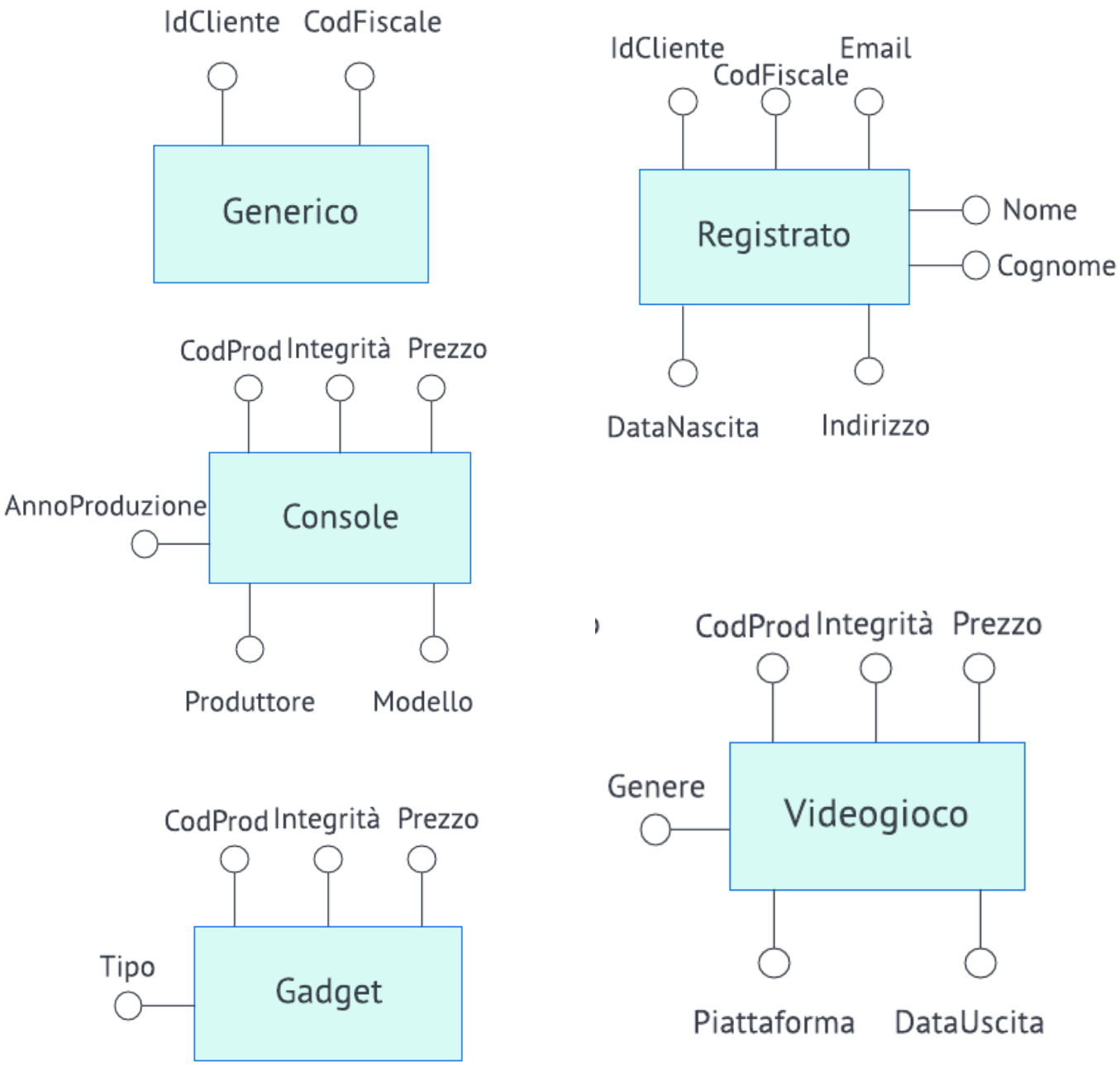
1. Un prodotto usato da aggiungere al catalogo deve avere integrità maggiore di 3.
2. Le prenotazioni per cliente registrato non devono superare i 4 prodotti.
3. L'offerta non può essere usata dopo la data di scadenza.
4. L'offerta può essere usata solo per un acquisto alla volta da clienti registrati.
5. La tessera non può superare il livello 4.

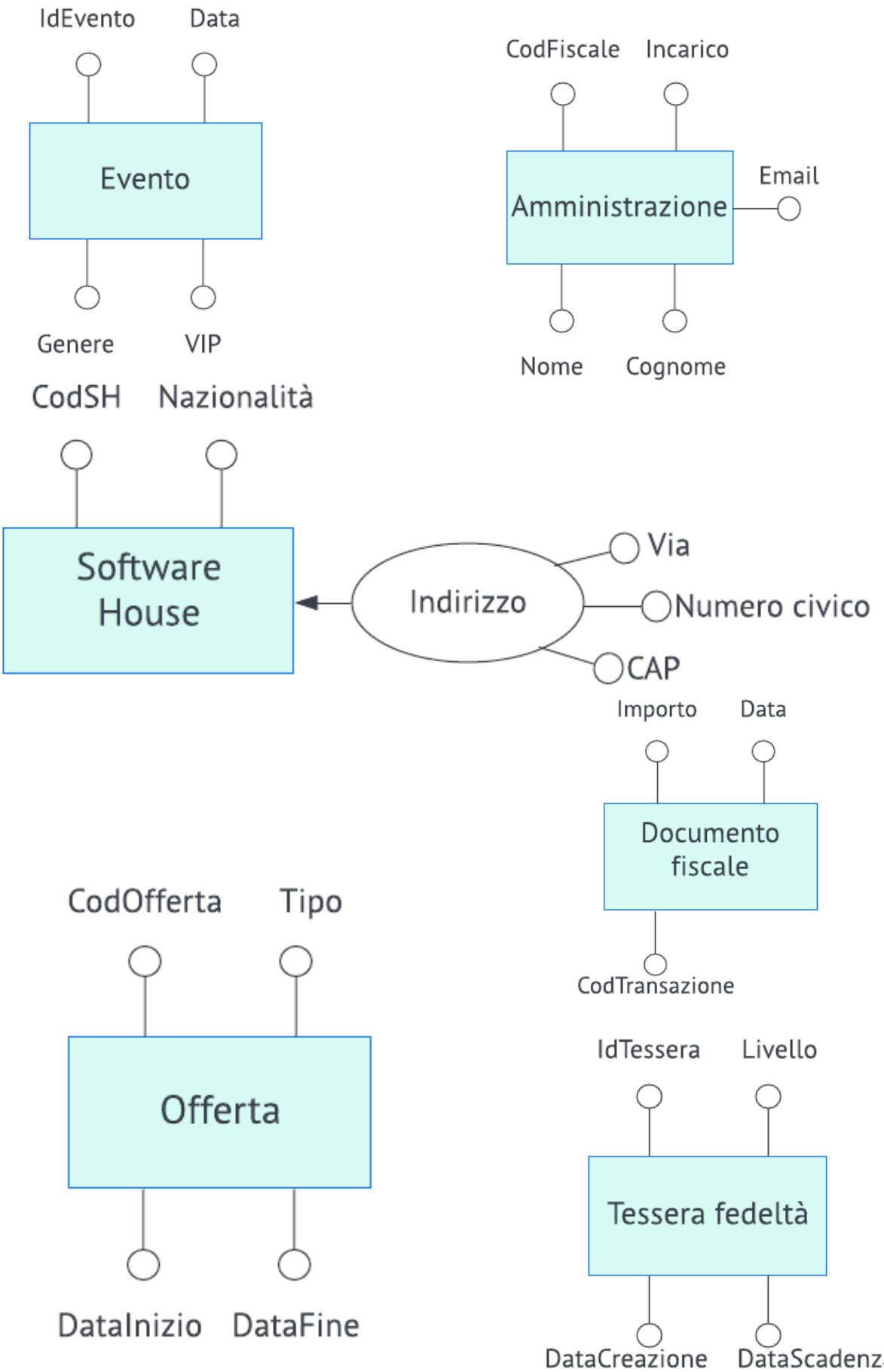
Le **relazioni** sono già state esplicitate come **entità associate** ma non sono ancora state descritte, poiché verranno approfondite nella prossima sezione, una volta disegnato e strutturato lo schema.

Schema E\R

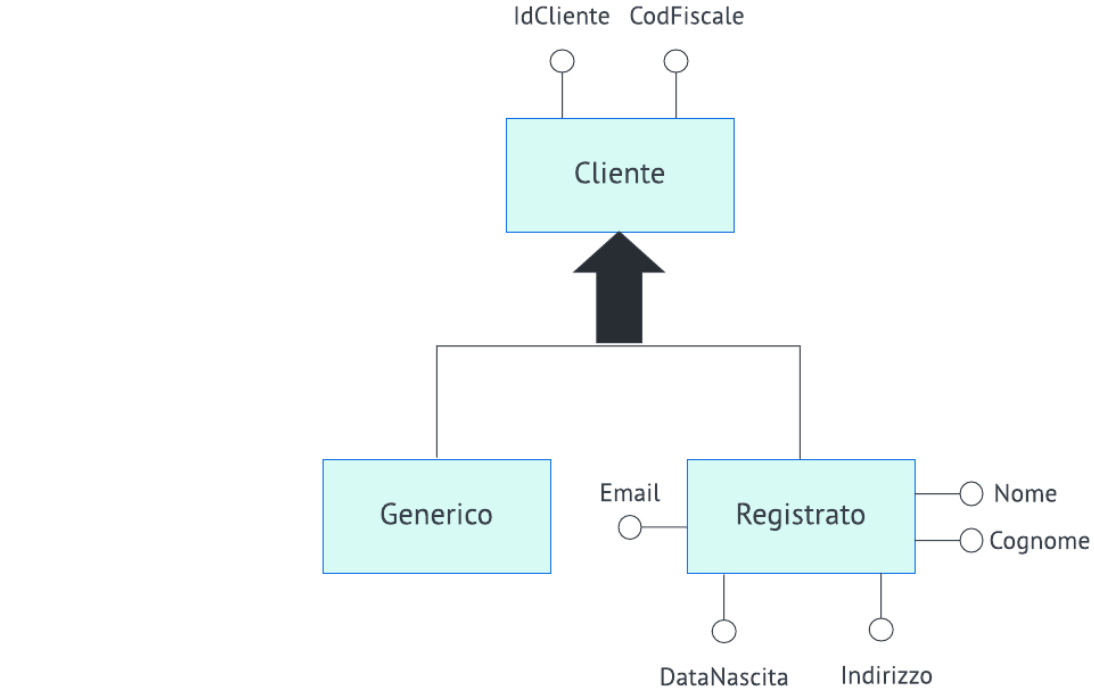
Dopo aver identificato e isolato i concetti principali, possiamo proseguire sviluppando lo **schema E\R**. Seguiamo le **regole concettuali** presentate nel corso di Basi di Dati per i simboli usati: rettangoli per le entità, rombi per le relazioni e cerchi per gli attributi. La strategia di progetteazione impiegata è **bottom-up**, quindi il flusso di definizione parte dai concetti elementari, i quali sono stati identificati nella sezione precedente, creando associazioni basandosi sulle interazioni descritte, fino ad arrivare allo schema finale. Lo **strumento** usato per disegnare i diagrammi è *lucidchart.com*.

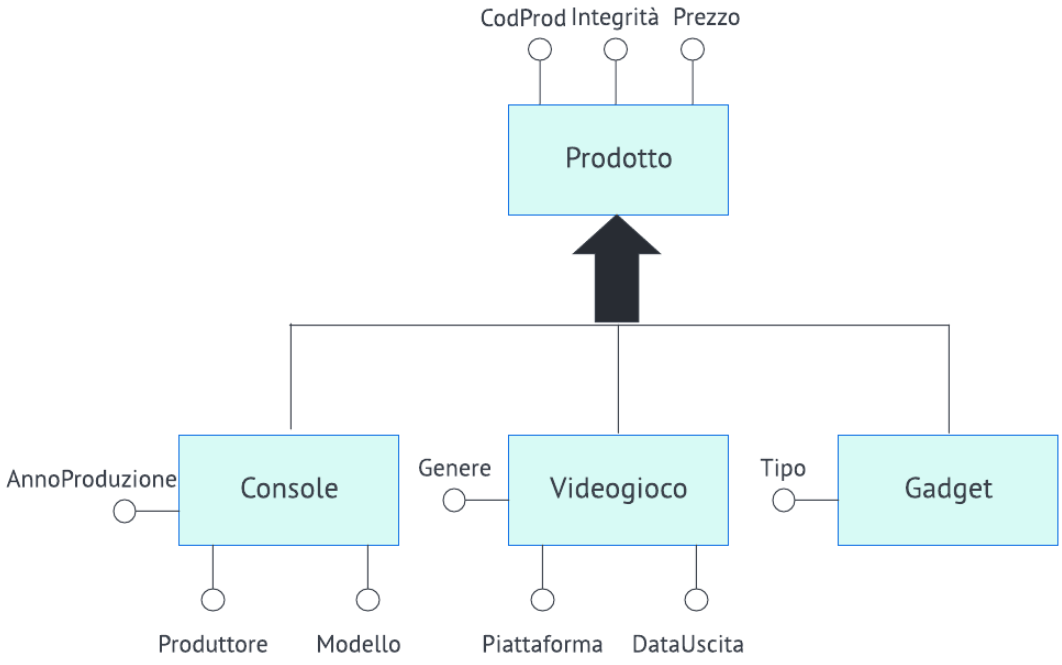
Seguono ora le entità definite rappresentate nello schema E\R con i loro attributi:



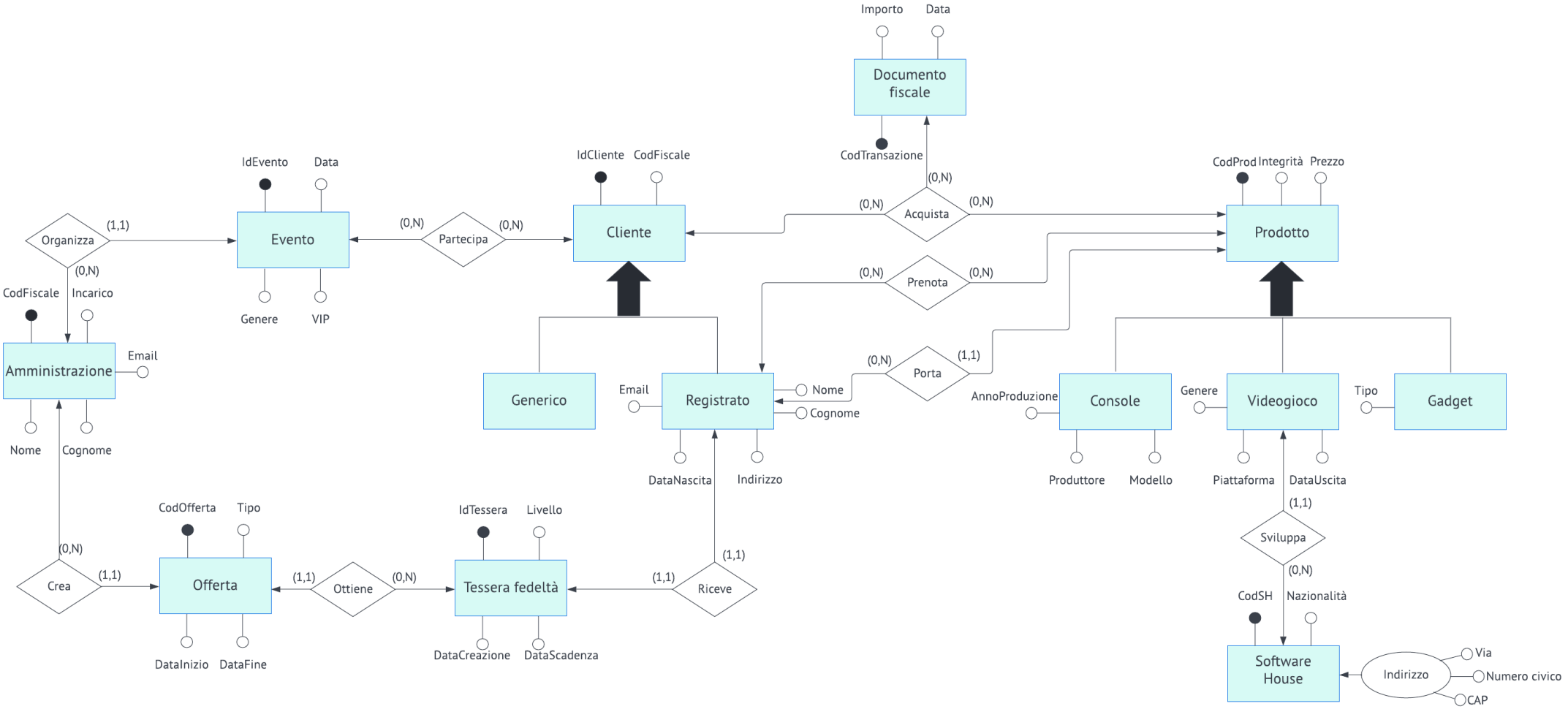

IdEvento
Data
Evento
Genere
VIP
CodFiscale
Incarico
Amministrazione
Email
Nome
Cognome
CodSH
Nazionalità
Software House
Indirizzo
Via
Numero civico
CAP
Importo
Data
Documento fiscale
CodTransazione
CodOfferta
Tipo
Offerta
DataInizio
DataFine
IdTessera
Livello
Tessera fedeltà
DataCreazione
DataScadenza

Da questa prima rappresentazione notiamo che, come ci aspettavamo in fase di definizione delle entità, sia Registrato e Generico, sia Videogioco, Console e Gadget sono rappresentabili in modo adeguato tramite delle **generalizzazioni**. Introduciamo quindi altre due entità chiamate **Cliente** e **Prodotto** che, tramite il costrutto della generalizzazione, riducono le entità simili ad una generale, la quale possiede tutti gli attributi comuni a ogni entità figlia:





Una volta definite le entità, possiamo collegarle tramite le **relazioni** descritte nella progettazione concettuale, completando lo schema con **cardinalità** adeguate (*min,max*) e attributi scelti per le **chiavi primarie** (*cerchi neri*), scelte giustificate nella sezione della progettazione logica:



Relazioni:

Vengono elencate e descritte di seguito le **relazioni** che sono state utilizzate per poter soddisfare la specifica (alcune di queste relazioni hanno degli attributi):

Acquista: ogni cliente (0,N) può acquistare un certo numero di prodotti (0,N) e produrre un certo numero di documenti fiscali (0,N).

Prenota: ogni cliente registrato (0,N) può prenotare un certo numero di prodotti (0,N).
Attributi: DataPrenotazione, DataArrivo, Quantità.

Porta: ogni cliente registrato (0,N) può portare in negozio un prodotto usato (1,1).
Attributi: Data, Integrità.

Riceve: ogni cliente registrato (1,1) ha un'unica tessera fedeltà attiva (1,1).

Partecipa: ogni cliente (0,N) può partecipare a un qualsiasi numero di eventi (0,N), il quale può ospitare un qualsiasi numero di clienti.

Organizza: ogni componente dell'amministrazione (0,N) organizza un evento (1,1) alla volta.

Crea: ogni componente dell'amministrazione (0,N) genera un'offerta (1,1).
Attributi: DataCreazione.

Ottiene: ogni tessera fedeltà (0,N) può ricevere un'offerta (1,1).
Attributi: DataAssegnazione.

Sviluppa: una software house (0,N) può sviluppare un qualsiasi numero di videogiochi (1,1). Un videogioco è sviluppato da una sola software house.

Progettazione logica

Ottimizzazione:
In questo passo della progettazione si prendono in esame il costo delle operazioni in termini di lettura\scrittura e volume dei dati per ottenere uno schema possibilmente privo di ridondanze ed efficiente. In particolare, ciò che interessa in questa fase è considerare due tipi di situazioni: **ridondanze concettuali** e **generalizzazioni**.

Le **ridondanze concettuali** avvengono nel caso in cui delle informazioni implicite sono derivabili da altri concetti ma sono allo stesso momento esplicitati da delle entità: per esempio, se abbiamo due entità Abitante e Città in relazione, potrebbe essere non necessario e poco efficiente indicare l'attributo "abitanti" per l'entità Città poiché è possibile scoprire questa informazione contando le istanze di Abitante in relazione con quella città in particolare. Considerando la struttura dello schema E\R descritto nella scorsa sezione non troviamo tali ridondanze.

Le **generalizzazioni** (le quali non sono rappresentabili nel modello relazionale) invece vanno analizzate per determinare se, una volta eliminate, causano perdite di dati o performance. Questa verifica avviene tramite delle tabelle di stima delle istanze di tutte le entità e relazioni sia che le generalizzazioni siano presenti, sia che non lo siano. Una volta stimate queste quantità, bisogna verificare per le varie operazioni se esse comportano una perdita di performance o meno. Considerando le generalizzazioni che sono state impiegate nella costruzione dello schema e come vengono sostituite, possiamo affermare che è possibile risolvere entrambi le generalizzazioni senza perdere performance.

Tabella di stima delle tuple **con generalizzazione** (solo entità rilevanti):

Tipo	Concetto	Volume	Descrizione
E	Cliente	10000	Supponiamo ci siano 10000 utenti, divisi in parti uguali tra generici e registrati
E	Prodotto	2000	Supponiamo ci siano 2000 prodotti in vendita, dove 1500 sono videogiochi, 100 console e 400 gadget
E	Documento fiscale	20000	Stiamiamo che ogni cliente faccia almeno due acquisti
E	Tessera fedeltà	5000	Gli utenti con tessera sono 10000/2
R	Acquista	20000	Ogni cliente effettua due acquisti
R	Prenota	2500	Metà dei clienti registrati prenota un videogioco
R	Porta	1250	Un quarto dei clienti registrati porta un prodotto
R	Riceve	5000	Ogni cliente registrato ha una tessera

Tabella di stima delle tuple **senza generalizzazione** (solo entità rilevanti):

Tipo	Concetto	Volume	Descrizione
E	Generico	5000	Supponiamo ci siano 5000 clienti generici
E	Registrato	5000	Supponiamo ci siano 5000 clienti registrati
E	Console	100	Supponiamo ci siano 100 console
E	Videogioco	1500	Supponiamo ci siano 1500 videogiochi
E	Gadget	400	Supponiamo ci siano 400 gadget
E	Tessera fedeltà	5000	Gli utenti con tessera sono 5000
E	Documento fiscale	20000	Stiamiamo che ogni cliente faccia almeno due acquisti
R	Acquista	20000	Ogni cliente effettua due acquisti
R	Prenota	2500	Metà dei clienti registrati prenota un videogioco
R	Porta	1250	Un quarto dei clienti registrati porta un prodotto
R	Riceve	5000	Ogni cliente registrato ha una tessera

Da una analisi delle stime delle operazioni effettuate risulta come nei due casi, eliminare la generalizzazione non comporta una perdita di performance. In particolare, le operazioni principali non subiscono peggioramenti per operazioni di lettura\scrittura (read\write) in alcun caso.

Inoltre, per completare lo schema, sono state scelte per chiavi principali delle chiavi semplici ed interne (dove necessario, generando **id autoincrementanti**).

Entrambi le modifiche - l'eliminazione delle generalizzazione e la scelta di chiavi principali – verranno gestite e rappresentate nella sezione successiva, cioè la fase di ristrutturazione dello schema.

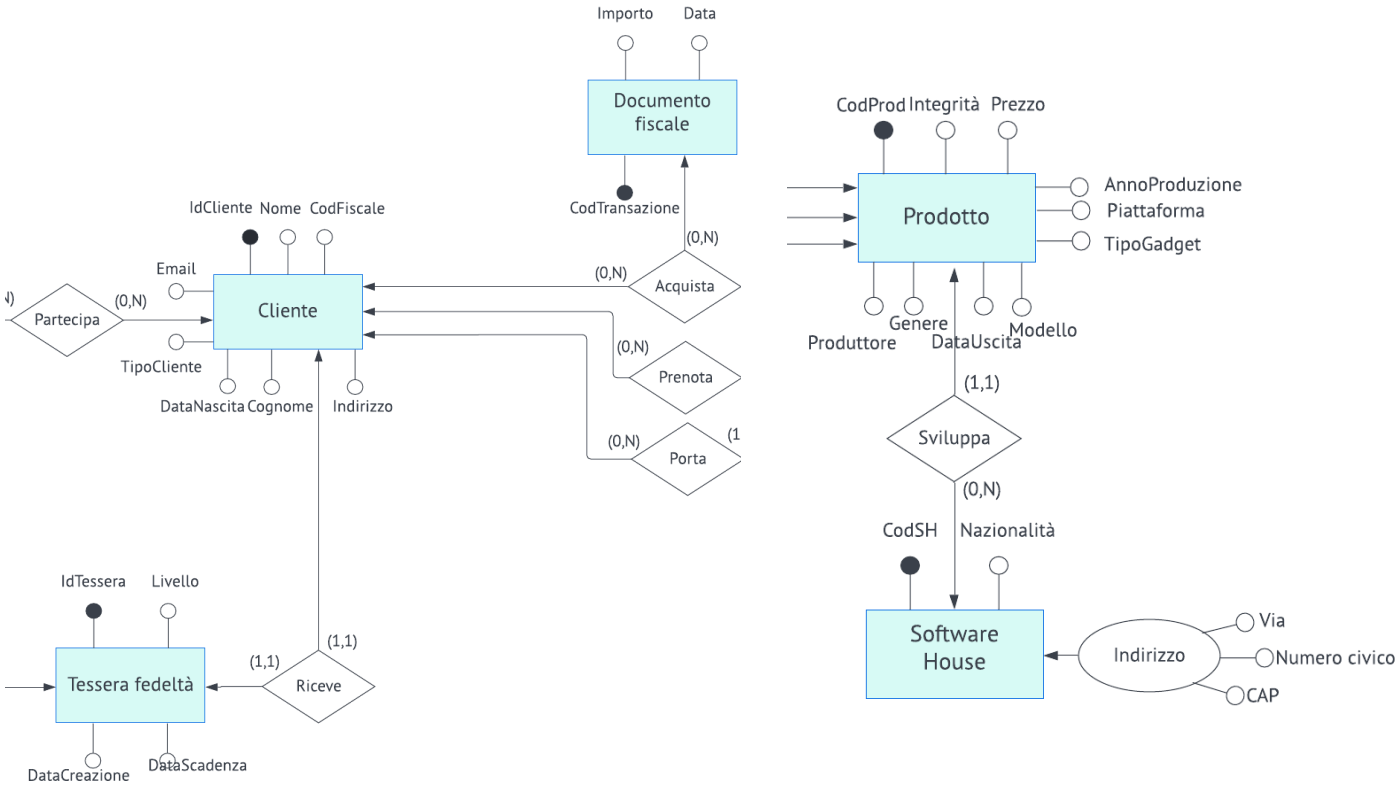
Ristrutturazione dello schema E\R

Analisi ridondanze:

Non sono presenti ridondanze logiche o concettuali rilevanti.

Eliminazione generalizzazioni:

Per eliminare le generalizzazioni, effettuiamo un accorpamento rispettivamento con le due e tre entità figlie: in questo modo, le entità Cliente e Prodotto erediteranno tutti gli attributi e relazioni, senza perdere nessuna informazione. Inoltre verranno aggiunti due attributi, per distinguere il tipo di cliente\prodotto. Di seguito abbiamo gli schemi E\R correttamente modificati:

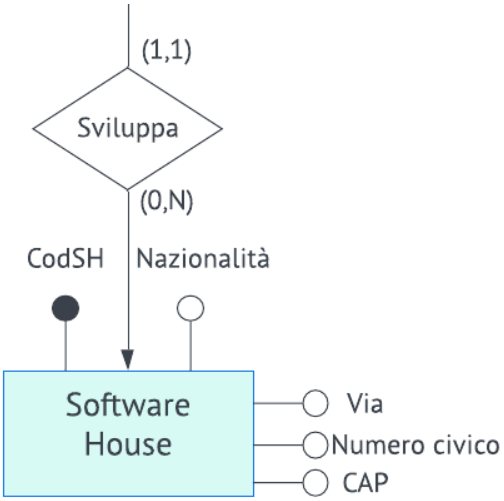


Partizionamento/accorpamento entità:

Non è il caso di partizionare o accorpare entità, poiché non genera vantaggi di alcun tipo.

Eliminazione attributi multivalore\composto:

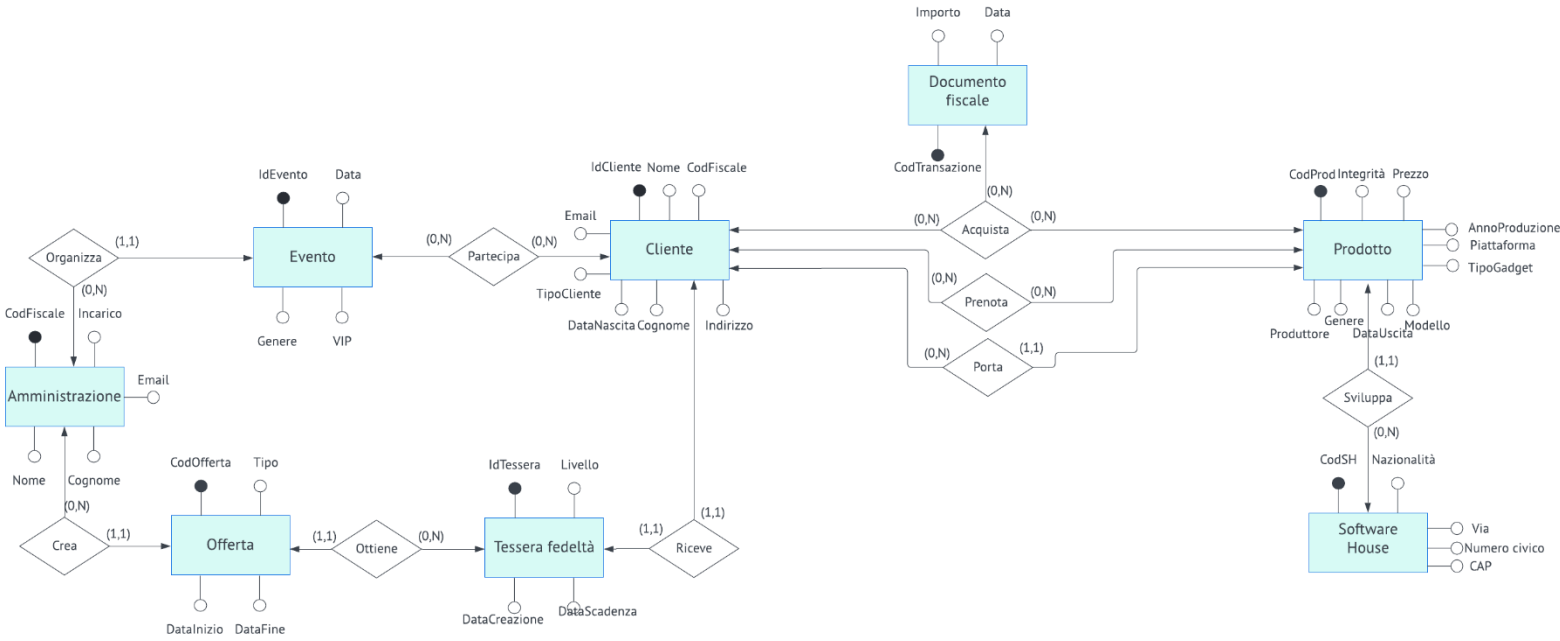
Nel nostro schema esiste un attributo multivalore\composto che va eliminato poichè il modello relazionale non è capace di rappresentare questi concetti. L'attributo è "Indirizzo" dell'entità Software House: il corretto svolgimento della ristrutturazione prevece che esso venga eliminato e gli attributi che vengono persi in questo modo sono semplicemente aggiunti all'entità principale.



Scelta degli identificatori principali:

Gli identificatori principali sono già stati definiti e sono stati scelti Id autoincrementati o altri codici come codici fiscali, i quali non possono assumere valori nulli, sono chiavi primarie semplici (costituite da un solo attributo) e tutte costituite da attributi interni.

Viene quindi presentato lo schema E\R completamente e correttamente ristrutturato:



Traduzione nello schema relazionale

Una volta creato e ristrutturato lo schema, possiamo ora tradurlo nel modello utilzzato, in questo caso quello relazionale. Quando necessario, verranno effettuate delle particolari traduzioni delle relazioni grazie allo studio sulle cardinalità (1-N, 1-1). Evidenziamo con una sottolineatura le chiavi primarie.

Cliente(IdCliente, TipoCliente, Nome, Cognome, CodFiscale, Email, DataNascita, Indirizzo)

Prodotto(CodProd, CodSH, Integrità, IdCliente, Prezzo, TipoProd, Piattaforma, DataUscita, TipoGadget, Modello, Produttore, AnnoProduzione, Genere)

SoftwareHouse(CodSH, Nazionalità, Via, CAP, NumCivico)

TesseraFedeltà(IdTessera, IdCliente, DataCreazione, DataScadenza, Livello)

Offerta(CodOfferta, CodFiscale, DataCreazione, DataInizio, DataFine, Tipo, IdTessera, DataAssegnazione)

DocumentoFiscale(CodTransizione, Importo, Data)

Evento(IdEvento, CodFiscale, VIP, Data, Genere)

Amministrazione(CodFiscale, Incarico, Nome, Cognome, Email)

Prenota(IdCliente, CodProd, Quantità, DataPrenotazione, DataArrivo)

Acquista(IdCliente, CodProd, CodTransizione, Data)

Partecipa(IdCliente, IdEvento)

Vincoli di integrità referenziale:
Prodotto.CodiceSH->SoftwareHouse

Prodotto.IdCliente, Tessera.IdCliente, Acquista.IdCliente, Partecipa.IdCliente ->Cliente

Acquista.CodiceProd, Prenota.CodiceProd ->Prodotto

Acquista.CodiceFattura->DocumentoFiscale

Offerta.IdTessera->Tessera

Offerta.CodFiscale, Evento.CodFiscale->Amministrazione

Partecipa.IdEvento->Evento

Prenota.IdCliente->Cliente

Normalizzazione

Il processo di **normalizzazione** (il quale è uno strumento di verifica, non di progettazione) garantisce che lo schema sia privo di anomalie, non ci siano perdita di dipendenze o ridonanze dannose. Esistono varie forme normali, ogni forma è requisito delle successive.

1FN

Uno schema è in **prima forma normale** se tutti gli attributi assumono valori in domini **atomici** e coerenti. Inoltre, non esistono tuple uguali o attributi uguali nella stessa tabella. Durante la creazione dello schema abbiamo utilizzato solamente domini atomici (numeri, stringhe, date) quindi la 1FN è banalmente verificata.

2FN

Uno schema è in **seconda forma normale** quando è in 1FN e non possiede attributi non-chiave parzialmente dipendenti dalla chiave. Possiamo affermare che il nostro schema è in 2FN in quanto ogni chiave da noi usata è costituita da un unico attributo.

3FN

Uno schema è in **terza forma normale** quando è in 2FN e tutti gli attributi non chiave dipendono dalla chiave e i campi non chiave non dipendono da altri campi non chiave. Il nostro schema è in terza forma normale in quanto tutti gli attributi dipendono direttamente dalla chiave e non ci sono attributi che dipendono da altri che non appartengano alla chiave.

Forma normale di Boyce-Codd (BCNF)

Uno schema è in **forma normale Boyce-Codd** quando per ogni dipendenza funzionale $X \rightarrow Y$ (se $\forall$coppia di tuple t_1 t_2, se $t_1[X] = t_2[X]$ allora $t_1[Y] = t_2[Y]$, in questo caso si dice che **X determina Y**), X è una superchiave. Possiamo considerare le DF come una generalizzazione delle (super)chiavi, le quali sono accettabili e non dannose.

Analizziamo ora le dipendenze funzionali:

Cliente:
"Cliente" è in BCNF perchè IdCliente e CodFiscale sono le uniche chiavi, con CodFiscale che dipende da IdCliente.

Prodotto:
"Prodotto" è in BCNF perchè CodProd è l'unica chiave.

SoftwareHouse:
"SoftwareHouse" è in BCNF perchè CodSH è l'unica chiave.

TesseraFedeltà:
“TesseraFedeltà” è in BCNF perchè IdTessera è l’unica chiave.

Offerta:
“Offerta” è in BCNF perchè CodOfferta è l’unica chiave.

DocumentoFiscale:
“DocumentoFiscale” è in BCNF perchè CodTransizione è l’unica chiave.

Evento:
“Evento” è in BCNF perchè IdEvento è l’unica chiave.

Amministrazione:
“Amministrazione” è in BCNF perchè CodFiscale è l’unica chiave.

Prenota:
“Prenota” è in BCNF perchè IdCliente,CodProd è l’unica chiave.

Acquista:
“Acquista” è banalmente in BCNF.

Partecipa:
“Partecipa” è banalmente in BCNF.

Poiché tutte le tabelle sono in BCNF, possiamo affermare che **l’intero schema lo è**.

Progettazione fisica

La **progettazione fisica** è lo step finale della progettazione della base di dati (prima dell'implementazione) e prevede l'analisi dei sistemi di memorizzazione e l'utilizzo di memorie secondarie che, al contrario dalle memoria primarie, hanno un rapporto spazio-costo vantaggioso e conferiscono persistenza. Sono usate terminologie ad-hoc: un attributo viene chiamato "**campo**", una tabella "**file**" e una tupla "**record**".

L'inserimento dei record all'interno di un file è casuale, quindi quando si cercherà un record specifico, la macchina dovrà effettuare una ricerca sull'intero file. Nel caso questo record sia l'ultimo o non sia presente, ci troviamo nel **caso pessimo**. Per questo motivo sono state sviluppate tecniche che impiegano strutture ausiliare come gli **indici**, le quali però incidono sulla performance e a seconda se sono **statiche** o **dinamiche** necessitano di **riordinamenti**. Esistono inoltre diversi tipi di organizzazione dei record.

Dal manuale di MySQL, ricaviamo il valore in byte allocato per ogni tipo di dato:

- `INT` usa 4 byte.
- `CHAR` usa 1 byte.
- `DATE` usa 10 byte.

Organizzazione seriale:
In questo tipo di organizzazione i record sono registrati in ordine di inserimento. Prendiamo in considerazione l'operazione di ricerca di un record. Il file può essere **ordinato** o **disordinato** rispetto all'attributo scelto come chiave di ricerca.
Prendiamo per esempio una possibile implementazione dell'entità "Cliente" del database:

```
CREATE TABLE cliente(
 IdCliente      int(10),          → 40 Byte
 Nome           varchar(20),      → 20 byte
 Cognome        varchar(10) ,     → 40 byte
 TipoCliente    varchar(20),      → 20 byte
 CodFiscale     varchar(16),      → 16 byte
 Email          varchar(40),      → 40 byte
 DataNascita    date              → 10 byte
 Indirizzo      varchar(40)       →  40 byte
);
```

Ogni record occupa **226 Byte**.
Si è stimato un volume di **10000** clienti nel database.
Assumiamo che i blocchi di memoria abbiano dimensione **1kB**.
Da cui possiamo dedurre che il numero di blocchi occupati dalla tabella:
NB = NR * 226B/1kB = 10000 * 226B/1kB = 2260 blocchi.

Organizzazione sequenziale:
I record sono indicizzati e per tanto è possibile l'accesso diretto all'i-esimo record tramite l'indice i. Questo tipo di strutture possiede pro e contro: infatti essi indicizzano in base ad uno o più campi il file e memorizzano gli indirizzi dei blocchi di memoria che contengono i record del file, ordinati secondo quello specifico attributo. Per questo, l'uso nelle operazioni di interrogazione è molto efficientie, mentre nel caso di una operazione di aggiornamento oltre al normale costo di aggiornamento, ci sarà il costo per ordinare tale aggiornamento all'interno anche dell'Indice. Per constatre ciò si può prendere in considerazione una tabella dove avvengono molte operazioni di interrogazione (SELECT).

Prendendo per esempio la tabella Cliente, avremo quindi 1131 **accessi** nel caso medio e 2260 in quello pessimo.

B+-tree:
Un B+-tree è un albero in cui i record sono contenuti solo nelle foglie, mentre i nodi interni hanno la funzione di determinare il percorso da seguire durante la ricerca. Le foglie quindi hanno funzione di lista. L'utilizzo di un B+-tree in combinazioni garantisce performance migliori di un qualsiasi altro tipo di ordinamento, soprattutto quando vengono effettuate operazioni di SELECT.

Implementazione del database in MySQL

Una volta completata la progettazione, possiamo passare all'implementazione del database, delle query, degli aggiornamenti e di un popolamento d'esempio tramite MySQL.

Creazione del database:

```
CREATE SCHEMA NegozioVideogiochi
```

Creazione delle tabelle:

```
CREATE TABLE cliente(
     IdCliente      int          AUTO_INCREMENT PRIMARY KEY,
     Nome           varchar(20)  NOT NULL,
     Cognome        varchar(15)  NOT NULL,
     TipoCliente    varchar(10)  NOT NULL,
     CodFiscale     varchar(16)  NOT NULL,
     Email          varchar(40)  NULL,
     DataNascita    date         NULL,
     Indirizzo      varchar(40)  NULL
);

CREATE TABLE prodotto(
     CodProd         varchar(10)  PRIMARY KEY,
     CodSH           int          NULL REFERENCES sh(codSH),
     IdCliente       int          NULL REFERENCES cliente(IdCliente),
     Integrità       int          NOT NULL CHECK (Integrità > 3),
     Prezzo          double       NOT NULL CHECK (Prezzo > 0),
     TipoProd        varchar(25)  NOT NULL,
     Piattaforma     varchar(15)  NULL,
     TipoGadget      varchar(25)  NULL,
     Modello         varchar(25)  NULL,
    Produttore      varchar(50)  NULL,
    Genere          varchar(20)  NULL,
    DataUscita      date         NULL,
    AnnoProduzione  date         NULL
);

CREATE TABLE documentoFiscale(
     CodTransizione     int    AUTO_INCREMENT PRIMARY KEY,
     Data               date   NOT NULL,
     Importo            double NOT NULL (CHECK Importo > 0)
);
```

```
CREATE TABLE amministrazione(
     CodFiscale    varchar(16) PRIMARY KEY,
     Nome          varchar(20) NOT NULL,
     Cognome       varchar(15) NOT NULL,
     Incarico      varchar(20) NOT NULL,
     Email         varchar(25) NOT NULL
);

CREATE TABLE softwareHouse(
     CodSH         int         AUTO_INCREMENT PRIMARY KEY,
     Nazionalità   varchar(20) NULL,
     Via           varchar(30) NULL,
     Cap           varchar(5)  NULL,
     NumeroCivico  varchar(3)  NULL
);

CREATE TABLE tesseraFedeltà(
     IdTessera      int         AUTO_INCREMENT PRIMARY KEY,
     IdCliente      int         REFERENECES cliente(Id),
     DataCreazione  date        NOT NULL,
     DataScadenza   date        NOT NULL,
     Livello        int         NOT NULL CHECK (Livello < 4)
);

CREATE TABLE evento(
     IdEvento      int         AUTO_INCREMENT PRIMARY KEY,
     CodFiscale    varchar(16) NOT NULL REFERENCES amm(CodFiscale),
     Vip           varchar(50) NULL,
     Genere        varchar(25) NULL,
      Data          date        NOT NULL
);

CREATE TABLE offerta(
     CodOfferta       int         AUTO_INCREMENT PRIMARY KEY,
     CodFiscale       varchar(16) NOT NULL REFERENCES amm(CodF),
     IdTessera         int         REFERENCES tessera(Id),
     DataCreazione    date        NOT NULL,
     DataInizio       date        NOT NULL,
     DataFine         date        NOT NULL,
     DataAssegnazione date        NOT NULL,
     Tipo             varchar(20) NOT NULL,
);
```

```
CREATE TABLE prenota(
     IdCliente         int          REFERENCES cliente(Id),
     CodProd           varchar(10) NOT NULL REFERENCES prod(codP),
     Quantità          int          NOT NULL,
     DataPrenotazione  date         NOT NULL,
     DataArrivo        date         NOT NULL,
    PRIMARY KEY(IdCliente, CodProd)
);

CREATE TABLE acquista(
     IdCliente         int          REFERENCES cliente(IdCliente),
     CodProd           varchar(10) REFERENCES prodotto(CodProd),
     CodTransizione    int          REFERENCES docFiscale(CodT),
     Data              date NOT NULL,
      PRIMARY KEY(IdCliente, CodProd, CodTransizione)
);

CREATE TABLE partecipa(
     IdCliente       int  REFERENCES cliente(IdCliente),
     IdEvento         int  REFERENCES evento(IdEvento),
      PRIMARY KEY(IdCliente, IdEvento)
);
```

Aggiornamenti e query

Aggiornamenti:

1. Registrare un cliente

```
INSERT INTO cliente(Nome, Cognome, TipoCliente, CodFiscale, Email,
DataNascita, Indirizzo)

VALUES(Alessio, Muzi, Generico, MZULSSC11H21V, tizio.caio@gmail.com,
12/03/2000, Via Pascoli 32);
```

2. Registrare una tessera fedeltà

```
INSERT INTO tesseraFedeltà(IdCliente, DataCreazione,  DataScadenza,
Livello)

VALUES(23, 02/08/2022, 02/08/2027, 3);
```

3. Registrare un prodotto

```
INSERT INTO prodotto(CodProd, CodSH, IdCliente, Integrità, Prezzo,
TipoProd, Piattaforma, TipoGadget, Modello, Produttore, Genere,
DataUscita, AnnoProduzione);

VALUES(H1GY892LF0, 12, NULL, 8, 69.90, Videogioco, Xbox, NULL, NULL,
NULL, Sparatutto, 11/11/2022, NULL);
```

4. Modificare un prodotto

```
UPDATE prodotto
SET Prezzo = 25.50
WHERE CodProd = 12;
```

5. Creare un evento

```
INSERT INTO evento(CodFiscale, Vip, Genere, Data)

VALUES(ANPCSS12H21C12, Cliff Blezinsky, Torneo, 11/09/2022);
```

6. Creare un’offerta

```
INSERT INTO offerta(CodFiscale, IdTessera, DataCreazione, DataInizio,
DataFine, DataAssegnazione, Tipo)
```

```
VALUES(ANPCSS12H21C12, 23, 02/02/2022, 10/02/2022, 10/04/2022,
05/02/2022, Sconto 20% su console Nintendo);
```

7. Eliminare un prodotto

```
DELETE FROM prodotto
WHERE CodProd = SHF12LL09Q;
```

8. Inserire un componente dell'amministrazione

```
INSERT INTO amministrazione(CodFiscale, Nome, Cognome, Incarico,
Email)

VALUES(TUPCSS23H21C12, Mario, Rossi, Cassiere, mario.rossi@gmail.com);
```

9. Rilasciare un documento fiscale

```
INSERT INTO documentoFiscale(Data, Importo)

VALUES(07/07/2022, 74.00);
```

Query:

1. Visualizzare il catalogo delle console ordinate per i loro prezzi.

```
SELECT Modello, Prezzo
FROM prodotto
WHERE TipoProd = "Console"
ORDER BY Prezzo
```

2. Visualizzare i clienti registrati.

```
SELECT Nome, Cognome
FROM cliente
WHERE TipoCliente = "Registrato"
```

3. Visualizzare gli eventi di un certo giorno.

```
SELECT IdEvento, Genere
FROM evento
WHERE Data = "02/08/2022"
```

4. Dato un videogioco, visualizzare la software house che lo ha sviluppato.

```
SELECT CodSH
FROM prodotto as p, softwareHouse as s
WHERE TipoProd = "Videogioco" AND p.CodSH = s.CodSH AND CodProd =
GH123GKSF
```

5. Visualizzare il prodotto più costoso.

```
SELECT CodProd
FROM prodotto
WHERE Prezzo = SELECT max(Prezzo)
               FROM prodotto
```

6. Visualizzare le tessere rilasciate e i clienti a cui sono associate.

```
SELECT IdTessera, IdCliente, Nome, Cognome, Livello
FROM tesseraFedeltà as t, cliente as c
WHERE t.IdCliente = c.IdCliente
```

7. Visualizzare i documenti fiscali rilasciati in un dato giorno.

```
SELECT CodTransizione, Importo
FROM documentoFiscale
WHERE Data = "08/09/2022"
```