Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria "Enzo Ferrari"

Corso di Laurea Triennale in Ingegneria Informatica

Data modeling: analisi delle notazioni e approfondimento dei tool esistenti

Relatore:

Prof.ssa Sonia Bergamaschi

Candidato:

Andrea Aureli

Anno Accademico 2020-2021

Sommario

INDICE DELLE FIGURE

1 INTRODUZIONE

La modellazione dei dati (o data modeling) è il processo di creazione di una rappresentazione visiva di un sistema informativo o di parti di esso. L'obiettivo è di mostrare in maniera coerente i tipi di dati utilizzati e archiviati all'interno del sistema, le relazioni tra questi dati, i modi in cui i dati possono essere raggruppati ed organizzati.

I dati possono essere modellati a vari livelli di astrazione. Il processo inizia raccogliendo informazioni sui requisiti aziendali dalle parti interessate e dagli utenti finali. Tali business rules vengono quindi tradotte in strutture dati per generare un database concreto. Il data modeling utilizza schemi standardizzati e tecniche formali. Ciò fornisce un modo comune, coerente e prevedibile di definire e gestire le risorse di dati all'interno di un'organizzazione.

Come ogni processo di progettazione, la modellazione di sistemi informativi inizia ad un alto livello di astrazione e diventa sempre più concreta e specifica. I modelli di dati possono essere generalmente suddivisi in tre categorie, che variano in base al loro grado di astrazione. Il processo inizia con un modello concettuale, procede con un modello logico e conclude con un modello fisico.

Modelli concettuali: sono modelli che permettono di rappresentare i dati in modo indipendente dal sistema che dovrà memorizzarli. Tali modelli descrivono i concetti del mondo reale e vengono utilizzati nelle prime fasi della progettazione di una base di dati. Un esempio è il modello E/R.

Modelli logici: sono i modelli utilizzati nei DBMS esistenti per l'organizzazione dei dati, sono indipendenti dalle strutture fisiche di memorizzazione dei dati. Ci sono i modelli relazionali, reticolari, gerarchici e ad oggetti.

Modelli fisici: forniscono uno schema che descrive il modo in cui i dati vengono archiviati fisicamente all'interno di un database. Mostrano le tabelle, le relazioni tra esse, nonché le chiavi primarie e quelle esterne. Possono includere proprietà specifiche del sistema di gestione del database (DBMS).

In questo lavoro vengono analizzate le notazioni esistenti che vengono utilizzate dai progettisti per rappresentare i modelli concettuali. In un primo momento, a partire da una descrizione testuale di una situazione reale, vengono realizzati due differenti modelli E/R seguendo la tradizionale notazione di Chen e la notazione Crow's foot. Si procede con il confronto tra le notazioni adottate e i limiti nella capacità espressiva delle notazioni. Viene mostrata anche la notazione adottata dal modello UML per la rappresentazione di basi di dati.

Il lavoro intende anche mostrare le potenzialità e i limiti di alcuni tool come "Star UML" e "DB Forge Studio" che propongono funzioni di forward e reverse engineering. Nel farlo si mettono in luce degli errori che provengono dal non riconoscimento di alcuni vincoli presenti nel database. Vengono anche analizzati e messi a confronto SQL Server e il Workbench di MySQL sempre per le funzioni di data modeling, forward e reverse engineering.

2 ESEMPIO COMPLETO DI PROGETTAZIONE CONCETTUALE E LOGICA

2.1 DESCRIZIONE TESTUALE DEL MODELLO

Si vuole creare un sistema informativo per un'agenzia immobiliare. L'agenzia ha diverse sedi, ognuna con un nome identificativo, un indirizzo e un recapito telefonico. Presso le sedi lavorano diversi dipendenti per i quali si riportano gli usuali dati anagrafici, inoltre ogni dipendente ha una data di assunzione ed un'eventuale data di licenziamento. Per ogni sede deve essere indicato un responsabile scelto tra i dipendenti, un dipendente può essere responsabile di una sola sede.

All'agenzia si rivolgono dei clienti, di cui si memorizzano gli usuali dati anagrafici e un recapito telefonico, per offrire o ricercare degli immobili. Per ogni richiesta di ricerca/offerta di un immobile viene aperta una pratica da un agente dell'agenzia (un dipendente dell'agenzia che svolge tale ruolo). Una pratica è identificata da un numero progressivo all'interno dell'anno e riporta la data di apertura.

Per ogni pratica, in caso di offerta deve essere indicato l'immobile e la tipologia dell'offerta: se è una vendita si riporta il prezzo di vendita; se è un affitto la richiesta mensile. Di un immobile viene riportato l'indirizzo, la tipologia e la metratura. Nel caso in cui la richiesta sia relativa ad una ricerca di un immobile, si memorizzano le preferenze del cliente, la tipologia ricercata (e.s. affitto/acquisto) e il budget a disposizione. Quando la pratica viene chiusa si riportano data e motivazione della chiusura (e.s. vendita, affitto, ritiro dell'offerta, etc.).

Un agente può fissare delle visite agli immobili per mostrarli ai clienti interessati, una visita è relativa a un cliente ed ha una data e un orario. Un cliente può visitare lo stesso immobile più volte in date diverse, con un massimo di tre visite totali. Un agente non può fissare più visite per la stessa data ed ora. Inoltre, si vuole tenere traccia del fatto che un cliente si presenti effettivamente alle visite, nel caso in cui un cliente non si presenti per più di due volte ad una visita il sistema lo deve etichettare come "cattivo cliente".

2.2 PROGETTAZIONE MODELLO E/R CON NOTAZIONE DI CHEN

2.2.1 INDIVIDUAZIONE DELLE ENTITÀ

- Sede
- Dipendente
- Responsabile

- Cliente
- Pratica
- Immobile
- Ricerca
- Vendita
- Affitto
- Visita

2.2.2 INDIVIDUAZIONE DELLE ASSOCIAZIONI

- Sede lavora Dipendente
- Sede ha Responsabile
- Agente apre Pratica
- Pratica per Cliente
- Offerta di Immobile
- Cliente – Visita – Immobile (Associazione ternaria)
- Agente segue Visita

2.2.3 INDIVIDUAZIONE VINCOLI NON ESPRIMIBILI

“Inoltre, si vuole tenere traccia del fatto che un cliente si presenti effettivamente alle visite, nel caso in cui un cliente non si presenti per più di due volte ad una visita il sistema lo deve etichettare come “cattivo cliente”.

2.2.4 RAFFINAMENTI

Nel sistema è presente un’entità Dipendente e un’entità Cliente che contengono quasi per interezza medesimi attributi e sono identificati dallo stesso attributo. Nell’entità Cliente poi verrà inserito un flag per identificare il “cliente cattivo” che non si presenta per più di due volte ad una visita. Per questa analogia tra le due entità si decide di inserire una gerarchia nel modello del tipo totale e sovrapposta in quanto potremmo avere un dipendente che si presenti verso l’agenzia anche come cliente. Inoltre un Dipendente può essere anche licenziato e quindi non fa più parte dei lavoratori dell’agenzia. L’entità Dipendente fa capo a sua volta ad una gerarchia che presenta alla base le entità Responsabile ed Agente. La proprietà di copertura della generalizzazione è parziale e sovrapposta: sovrapposta poiché il Responsabile di una Sede è scelto tra i Dipendenti e parziale poiché qualche Dipendente può esser stato licenziato e non è più Responsabile o Agente.

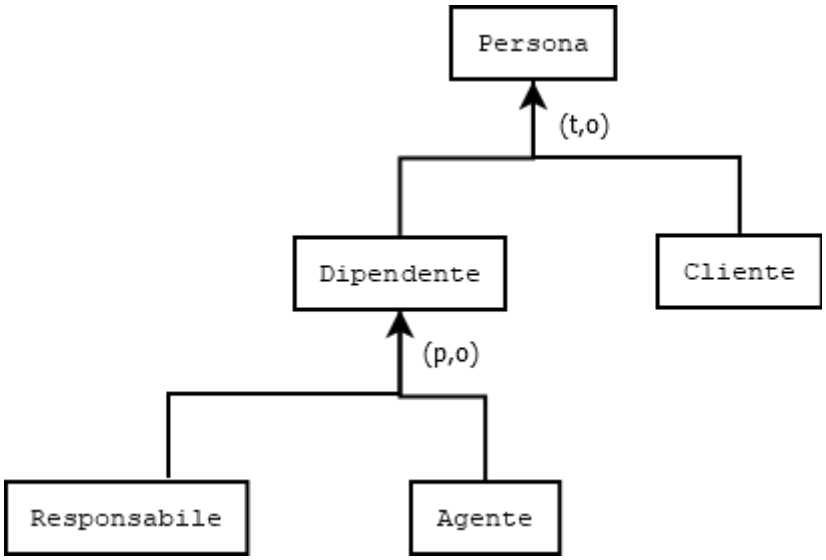


Figura 1 Schema scheletro prima gerarchia

Le Pratiche vengono divise tra Ricerche ed Offerte in modo totale ed esclusivo. A sua volta l’entità Offerta fa capo ad una gerarchia totale ed esclusiva che alla base presenta le entità Vendita ed Affitto. Si decide di inserire la sottoclasse Chiusa relativa alle Pratiche chiuse.

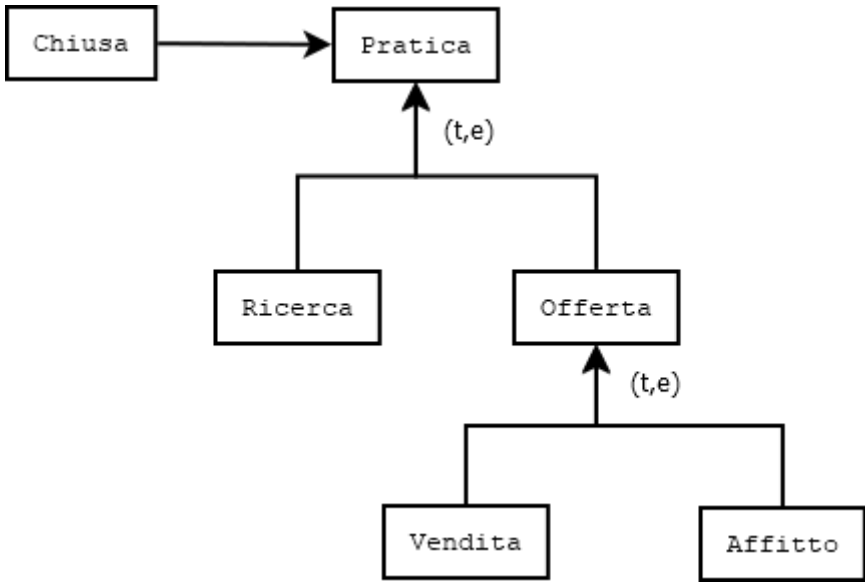


Figura 2 Schema scheletro seconda gerarchia

Per la parte che riguarda i Clienti e le Visite agli Immobili, si decide di reificare e creare l'entità Cliente_Immobile che risulta utile per rappresentare il vincolo che un cliente può visitare lo stesso immobile più volte in date diverse, con un massimo di tre visite totali. L'identificatore di Cliente_Immobile è una chiave composta costituita dalla chiave di Cliente e dalla chiave di Immobile in quanto nessuno dei due partecipa con una max-card uguale ad uno.

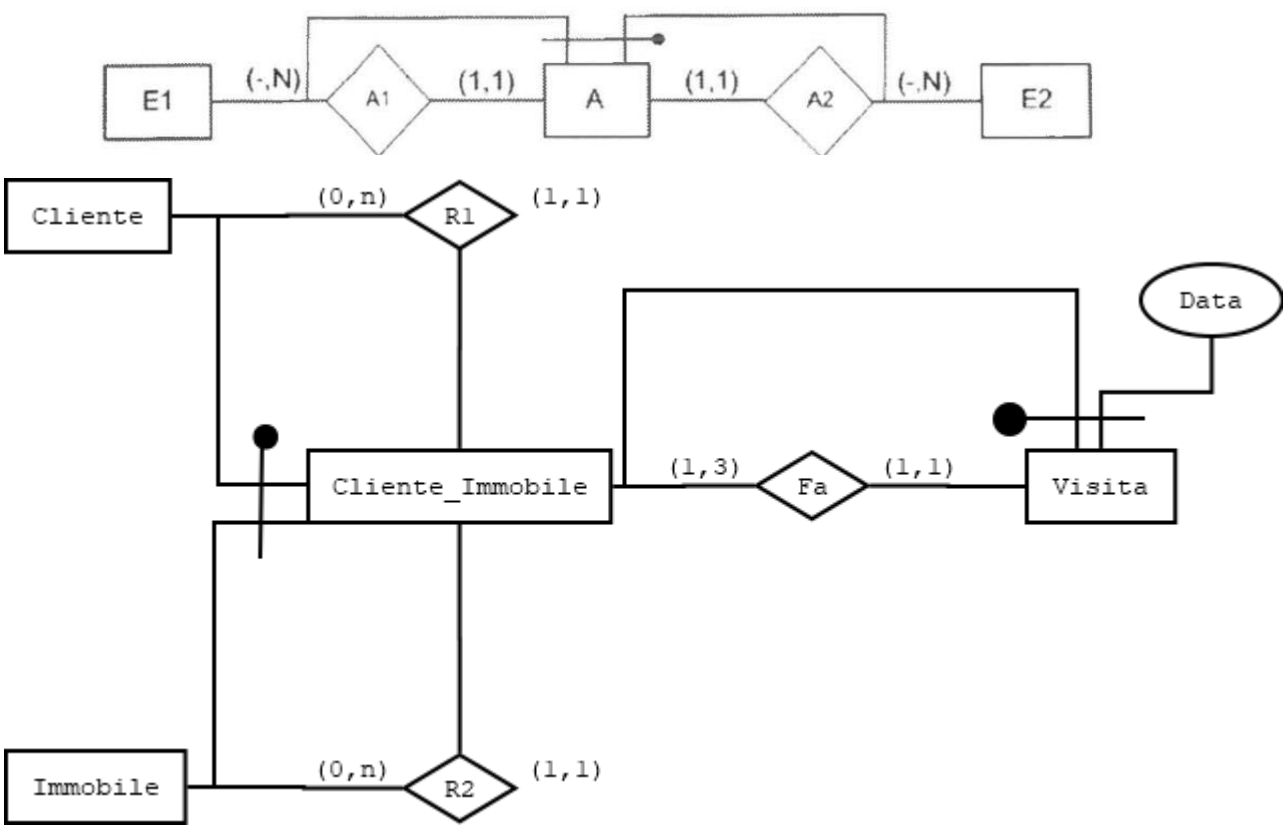


Figura 3 Esempio di reificazione

“Un agente può fissare delle visite agli immobili per mostrarli ai clienti interessati, una visita è relativa a un cliente ed ha una data e un orario.”

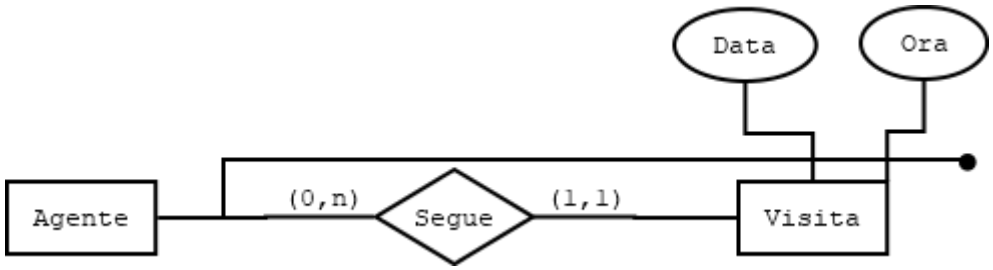


Figura 4 Chiave composta e mixed

2.2.5 MODELLO E/R CON NOTAZIONE DI CHEN COMPLETO

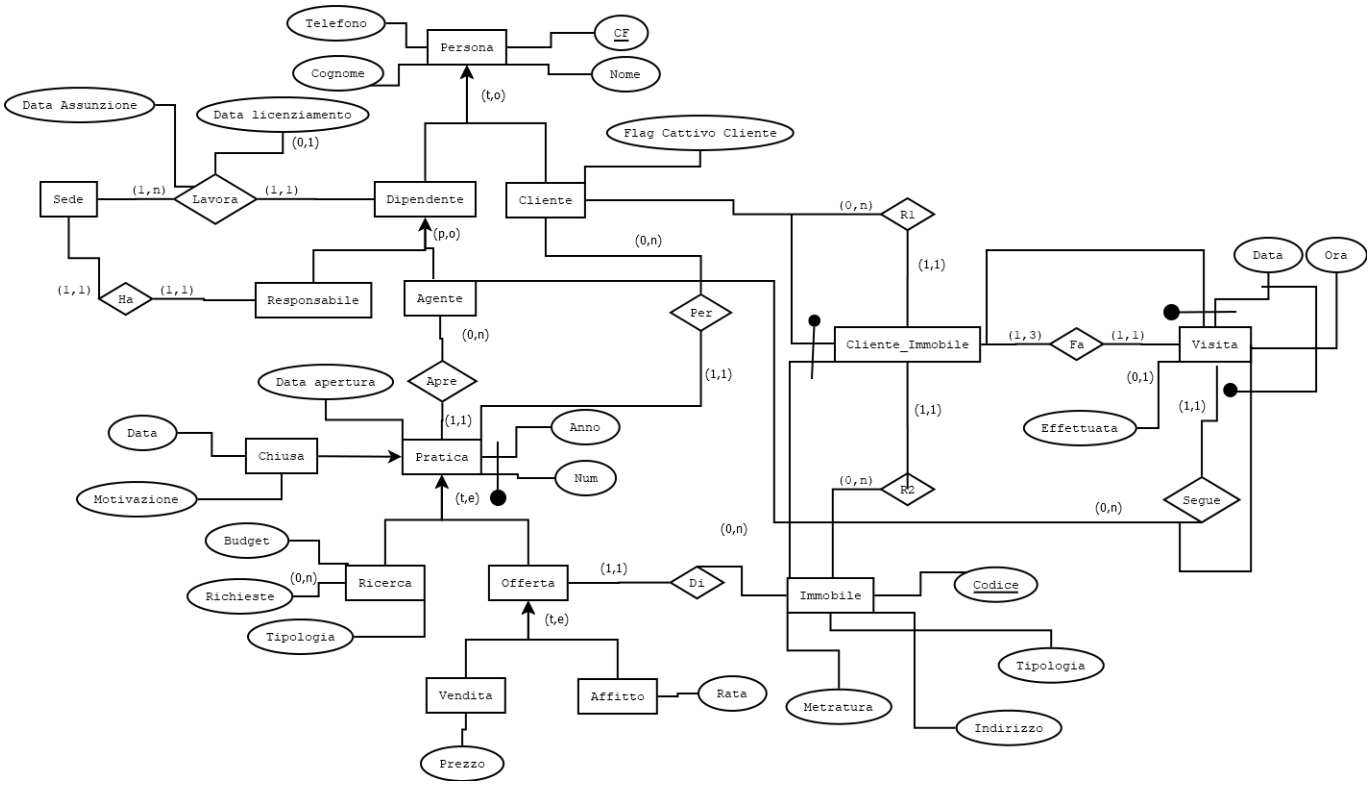


Figura 5 Diagramma E/R completo tramite software applicativo DIA

2.3 SCHEMA LOGICO

Per la gerarchia di Persona-Dipendente-Cliente si sceglie di effettuare un collasso verso il basso. È doveroso ricordare che il collasso verso il basso con una copertura parziale comporta

necessariamente della ridondanza. Un collasso verso l'alto costituirebbe un problema dato che Dipendente fa capo a sua volta ad un'ulteriore gerarchia e sarebbe complicato esprimerla. Per la gerarchia Dipendente-Agente-Responsabile, dato che si presenta con proprietà di copertura parziale e sovrapposta è stato effettuato un collasso verso l'alto. Le entità Agente e Responsabile non presentano attributi dunque non vi è la possibilità di eventuali valori NULL. L'associazione tra Sede e Responsabile è stata sciolta portando dentro la tabella Sede la chiave primaria di dipendente come Foreign Key e anche come Alternative Key in quanto era un'associazione (1,1) in entrambe le direzioni.

Scendendo verso il basso troviamo l'entità Pratica, la quale ha una sottoclasse Chiusa e fa capo ad una gerarchia con alla base Ricerca ed Offerta. Si è scelto di effettuare un collasso verso il basso andando ad eliminare l'entità Pratica. La sottoclasse Chiusa è stata inglobata sia dentro Ricerca che dentro Offerta. Il collasso verso il basso è possibile dato che la copertura è totale, inoltre non si verificherà ridondanza dato che è anche esclusiva.

Nell'entità Ricerca è presente un attributo multivalore Richieste. La 1NF impone che, se un'entità E ha un attributo multiplo A, si crei una nuova entità EA che ha A come attributo singolo ed è collegata ad E. Questo è il caso in cui il valore può comparire una sola volta nella ripetizione. L'entità EA ha l'identificatore composto dall'entità E più un attributo identificatore sintetico.

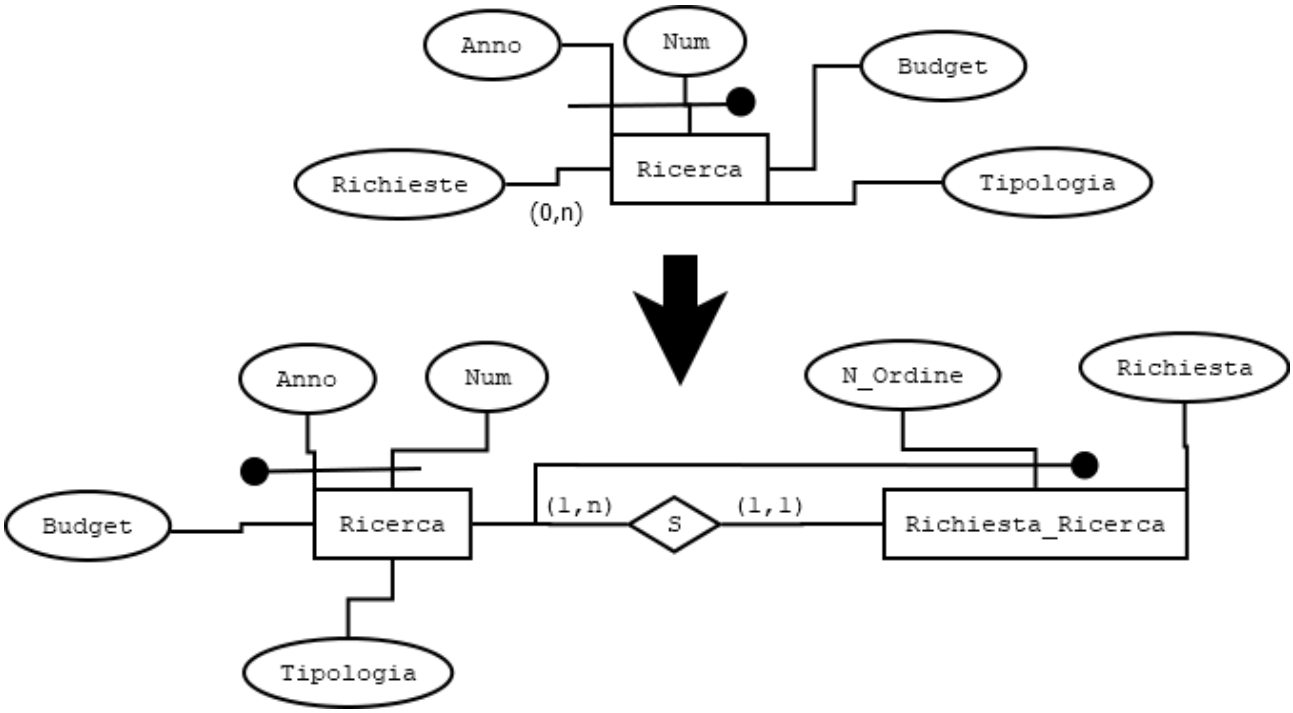


Figura 6 Semplificazione attributo multiplo

Per la gerarchia con a capo Offerta ed alla base Affitto e Vendita viene scelto di effettuare un collasso verso l'alto, portando dentro l'entità Offerta, l'attributo Prezzo e Rata che non potranno assumere

valore NULL contemporaneamente. Il collasso verso l'alto di una gerarchia totale ed esclusiva prevede l'inserimento di un selettore che potrà assumere N valori, quante sono le sottoentità(in questo caso due).

Per le associazioni binarie uno a molti è stata scelta la traduzione con due relazioni. La traduzione con tre relazioni potrebbe entrare in gioco nel caso della presenza di uno o più attributi sull'associazione, evento che qui non accade.

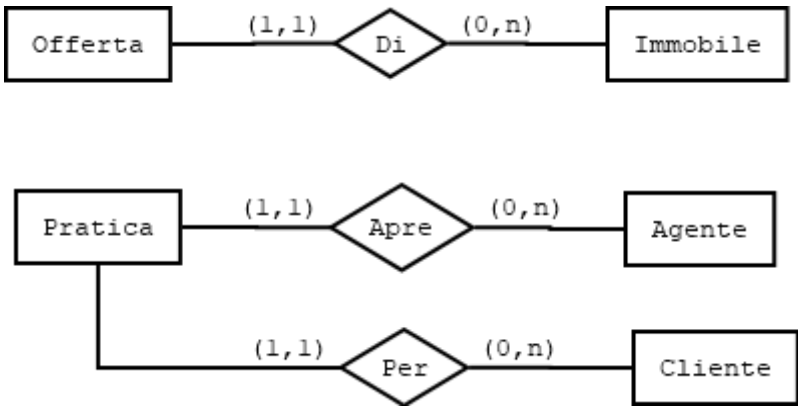


Figura 7 Associazioni binarie varie

Dentro l'entità Offerta viene portato l'identificatore di Immobile e dentro Pratica vengono portati dentro gli identificatori di Agente e di Cliente.

- **Dipendente** (CF, Nome, Cognome, Telefono, Data_Assunzione, Nome_Sede, Data_Licenziamento)

  FK Nome_Sede **REFERENCES** Sede

  Note: l'attributo Data_Licenziamento può essere NULL.

- **Cliente** (CF, Nome, Cognome, Telefono, Flag_Cattivo_Cliente)

  Note: l'attributo Flag_Cattivo_Cliente è un booleano

- **Sede** (Nome, Indirizzo, Telefono, CF_Responsabile)

  AK CF_Responsabile

  FK CF_Responsabile **REFERENCES** Dipendente

- **Ricerca** (Anno, Num, Data_Apertura, Budget, Data_Chiusura, Motivazione, CF_Cliente, Sel1, CF_Dipendente)

  FK CF_Cliente **REFERENCES** Cliente

  FK CF_Dipendente **REFERENCES** Dipendente

  Dominio Sel1 {Aperto, Chiuso}

  Note: Data_Chiusura e Motivazione assumono valori NULL se Sel1 è Chiuso. Altrimenti non possono essere NULL.

- **Richiesta_Ricerca** (N_Ordine, Anno, Num, Richiesta)

  FK Anno, Num **REFERENCES** Ricerca

- **Immobile** (Codice, Tipologia, Metratura, Indirizzo)

- **Offerta** (Anno, Num, Data_Apertura, Data_Chiusura, Motivazione, Prezzo, Rata, Sel, Sel1, Codice_Immobile, CF_Dipendente, CF_Cliente)

  Dominio Sel {Vendita, Affitto}

  Dominio Sel1 {Aperto, Chiuso}

  FK CF_Cliente **REFERENCES** Cliente

  FK CF_Dipendente **REFERENCES** Dipendente

  FK Codice_Immobile **REFERENCES** Immobile

  Note: Data_Chiusura e Motivazione assumono valori NULL se la pratica non è chiusa. Altrimenti non possono avere valori NULL. Prezzo e Rata possono assumere valori NULL a

seconda che l'offerta sia una vendita o un affitto. Non possono assumere il valore NULL contemporaneamente.

- **Cliente_Immobile** (CF_Cliente, Codice_Immobile)

  FK CF_Cliente **REFERENCES** Cliente

  FK Codice_Immobile **REFERENCES** Immobile

- **Visita** (Data, Ora, CF_Dipendente, CF_Cliente, Codice_Immobile, Effettuata)

  AK Data, CF_Cliente, Codice_Immobile

  FK CF_Dipendente **REFERENCES** Dipendente

  FK CF_Cliente, Codice_Immobile **REFERENCES** Cliente_Immobile

  Note: l'attributo Effettuata può assumere valore NULL

Vincolo non esprimibile: un cliente può visitare per un massimo di tre volte lo stesso immobile.

2.3.1 IMPLEMENTAZIONE SU MICROSOFT SQL SERVER

Codice per la creazione delle tabelle nel database:

```
create table Cliente(
        CF char(16) primary key,
        Nome varchar(15) not null,
        Cognome varchar(15) not null,
        Telefono integer not null,
Flag_Cattivo_Cliente bit not null,
        check (Telefono between 0 and 9999999999)
);
```

```
create table Dipendente(
        CF char(16) primary key,
        Nome varchar(15) not null,
        Cognome varchar(15) not null,
        Telefono integer not null,
        Data_Assunzione date not null,
        Data_Licenziamento date,
        check (Telefono between 0 and 9999999999)
);


create table Sede(
        Nome nvarchar(50),
        Indirizzo varchar(50),
        Telefono integer,
        CF_Responsabile char(16) not null,
        check (Telefono between 0 and 9999999999),
        FOREIGN KEY (CF_Responsabile) REFERENCES Dipendente(CF),
        UNIQUE(CF_Responsabile)
);


create table Ricerca(
        Anno date check(Anno between 1900 and 2100),
        Num integer,
        Data_Apertura date not null,
        Budget integer not null,
        Data_Chiusura date,
        Motivazione nvarchar(100),
        CF_Cliente char(16),
        Sel1 nvarchar not null check(Sel1 = 'Aperto' OR Sel1 = 'Chiuso'),
```

```
	CF_Dipendente char(16),
	FOREIGN KEY (CF_Cliente) REFERENCES Cliente(CF),
	FOREIGN KEY (CF_Dipendente) REFERENCES Dipendente(CF),
	check((Sel1 = 'Aperto' AND Data_Chiusura IS NULL AND Motivazione IS NULL) OR (Sel1 = 'Chiuso' AND Data_Chiusura IS NOT NULL AND Motivazione IS NOT NULL)),
	primary key (Anno, Num)
);

create table Richiesta_Ricerca(
	N_Ordine integer,
	Anno integer check(Anno between 1900 and 2100),
	Num integer,
	Richiesta integer not null,
	FOREIGN KEY (Anno, Num) REFERENCES Ricerca(Anno, Num),
	primary key(Anno, Num, N_Ordine)
);

create table Immobile(
	Codice integer primary key,
	Tipologia nvarchar(20) not null,
	Metratura integer not null,
	Indirizzo nvarchar(50) not null
)

create table Offerta(
	Anno integer check(Anno between 1900 and 2100),
	Num integer,
	Data_Apertura date not null,
	Data_Chiusura date,
	Motivazione nvarchar(100),
```

```
	Prezzo integer,
	Rata integer,
	Sel bit not null,
	Sel1 bit not null,
	Codice_Immobile integer,
	CF_Dipendente char(16),
	CF_Cliente char(16),
	FOREIGN KEY (Codice_Immobile) REFERENCES Immobile(Codice),
	FOREIGN KEY (CF_Dipendente) REFERENCES Dipendente(CF),
	FOREIGN KEY (CF_Cliente) REFERENCES Cliente(CF),
	check((Sel = 'Vendita' AND Prezzo IS NOT NULL AND Rata IS NULL) OR (Sel = 'Affitto' AND Prezzo IS NULL AND Rata IS NOT NULL)),
	check((Sel1 = 'Aperto' AND Data_Chiusura IS NULL AND Motivazione IS NULL) OR (Sel1 = 'Chiuso' AND Data_Chiusura IS NOT NULL AND Motivazione IS NOT NULL)),
	primary key(Anno, Num)
);

create table Cliente_Immobile(
	CF_Cliente char(16),
	Codice_Immobile integer,
	FOREIGN KEY (Codice_Immobile) REFERENCES Immobile(Codice),
	FOREIGN KEY (CF_Cliente) REFERENCES Cliente(CF),
	primary key (CF_Cliente, Codice_Immobile)
);

create table Visita(
	Giorno date,
	Ora time,
	Codice_Immobile integer not null,
	CF_Dipendente char(16),
	CF_Cliente char(16) not null,
	Effettuata varchar,
```

unique(Codice_Immobile, Giorno, CF_Cliente),

FOREIGN KEY (CF_Cliente, Codice_Immobile) REFERENCES Cliente_Immobile(CF_Cliente, Codice_immobile),

FOREIGN KEY (CF_Dipendente) REFERENCES Dipendente(CF),

primary key (Giorno, Ora, CF_Dipendente)

);

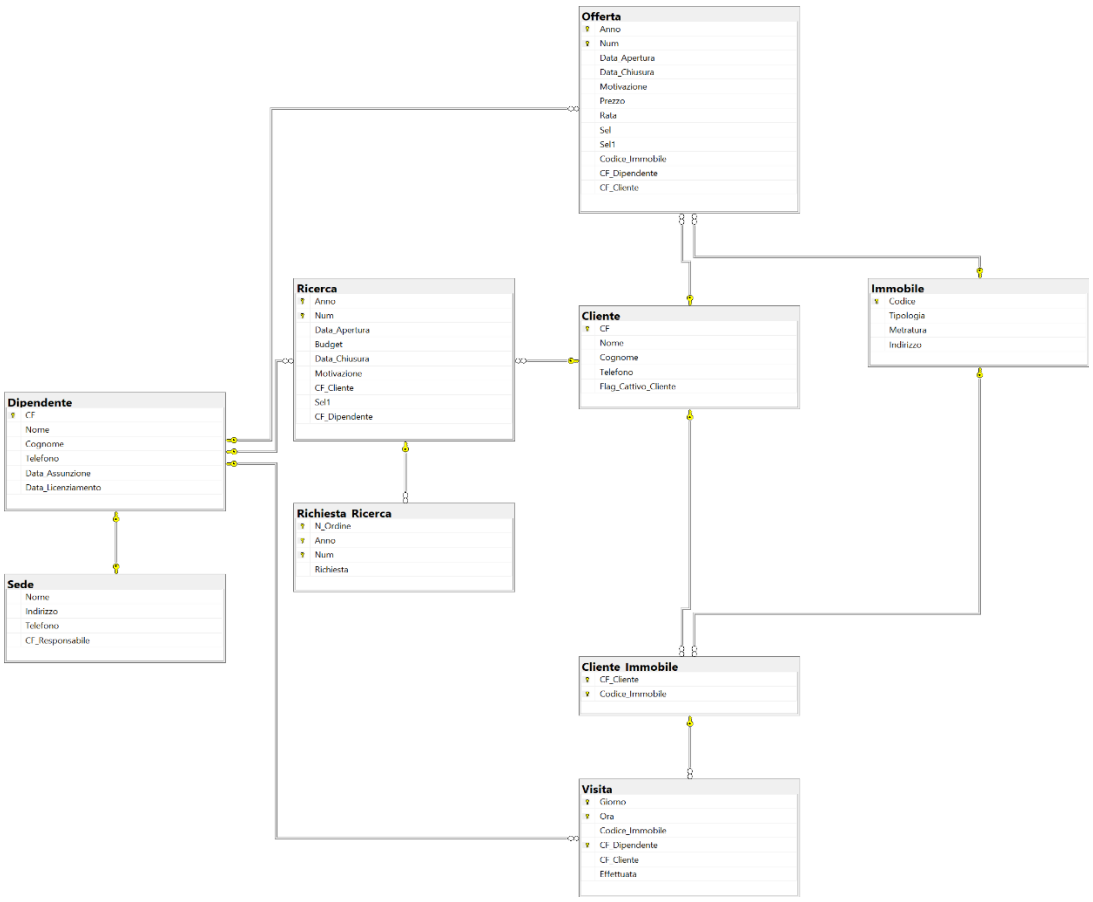


Figura 8 Database diagram realizzato con SQL Server

2.4 PROGETTAZIONE MODELLO E/R CON NOTAZIONE CROW'S FOOT

Dopo aver illustrato il modello E/R con la notazione di Chen si passa alla progettazione tramite l'ausilio della notazione Crow's Foot. Per la realizzazione del diagramma è stato utilizzato il software applicativo StarUML.

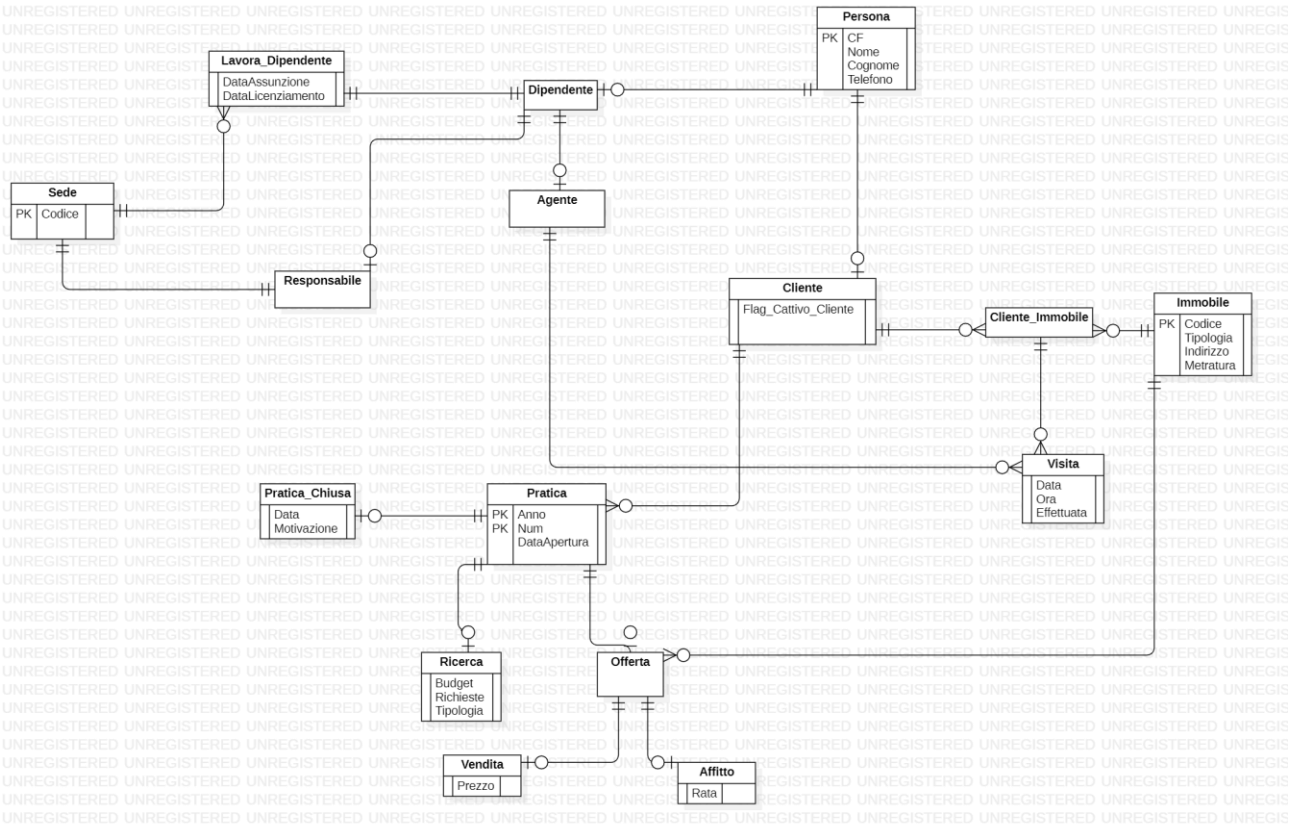


Figura 9 Modello E/R con notazione Crow's foot

2.4.1 DOCUMENTAZIONE ALLEGATA AL MODELLO E/R CON NOTAZIONE CROW'S FOOT

- L'attributo "Richieste" presente nell'entità "Ricerca" costituisce un attributo multivalore.

- Un cliente può visitare lo stesso immobile per un massimo di tre volte. La relazione tra Cliente_Immobile e Visita prevede che una visita sia associata con cardinalità (1,1) a Cliente_Immobile mentre, al contrario, Cliente_Immobile sia associato con cardinalità (1,3) con Visita.

- Le entità "Dipendente" e "Cliente" sono figlie dell'entità "Persona".

- Una “Persona” può essere o un “Dipendente” o un “Cliente”, anche entrambi contemporaneamente. Non esistono altri ruoli per una “Persona”.

- Le entità “Responsabile” ed “Agente” sono figlie dell’entità “Dipendente”.

- Un “Responsabile” può ricoprire anche il ruolo di “Agente” e viceversa. Un Dipendente può anche non essere né un “Responsabile” né un “Agente”.

- Le entità “Ricerca” ed “Offerta” sono figlie dell’entità “Pratica”.

- Una “Pratica” può essere o una “Ricerca” o una “Offerta”, ma non entrambi.

- Le entità “Vendita” ed “Affitto” sono figlie dell’entità “Offerta”.

- Un’ “Offerta” può essere o una “Vendita” o un “Affitto”, ma non entrambi.

- L’entità “Pratica_Chiusa” è un subset di “Pratica”. Una “Pratica” può essere chiusa o meno.

- Nel caso in cui un cliente non si presente per più di due volte ad una visita il sistema lo deve etichettare come “cattivo cliente”.

2.5 DIFFERENZE TRA LE NOTAZIONI ADOTTATE

Uno schema E/R è di solito corredato di una documentazione di supporto in modo tale da descrivere vincoli di integrità non esprimibili in E/R e per permettere una lettura più semplice e veloce del diagramma. Le descrizioni di queste proprietà non direttamente esprimibili prendono il nome di regole aziendali (o **business rules)**. Le regole aziendali possono assumere la forma di una descrizione di un concetto (entità, associazione e attributo), di un vincolo di integrità o di una derivazione (dato derivato). Tali regole aziendali dovranno poi essere implementate in SQL attraverso la definizione di trigger, definizione di vincoli in linguaggio SQL o con procedure in un linguaggio di programmazione.

Qui siamo di fronte ad un esempio non particolarmente complesso e articolato. Nella progettazione del diagramma E/R con notazione di Chen riusciamo ad esprimere tutti i vincoli a parte il flag "Cattivo_Cliente" che deve essere settato quando un cliente non si presenta per più di due volte ad una visita. Nella progettazione con la notazione Crow's foot si presenta, invece, la necessità di allegare una documentazione al nostro diagramma che permetta una visione corretta e veloce del diagramma.

In primis, nella notazione Crow's foot non esiste la rappresentazione di gerarchie, ovvero la possibilità di illustrare delle classi padri da cui discendono dei figli. È necessario in questo caso esprimere a parole la presenza della gerarchia oltre a rappresentare un'associazione tra le due entità. Solo con la rappresentazione dell'associazione si raggiungerebbe uno schema logico errato.

Inoltre vanno ricordate le proprietà di copertura delle generalizzazione della notazione di Chen:

- Totale ed esclusiva (t,e)
- Parziale ed esclusiva (p,e)
- Parziale e sovrapposta (p,o)
- Totale e sovrapposta (t,o)

Anche queste proprietà di copertura, che risultano fondamentali per l'implementazione in SQL, risultano esprimibili solo a parole e non direttamente sul diagramma data l'assenza delle gerarchie di generalizzazione.

Un'altra differenza che si può notare è la varietà di cardinalità che abbiamo sulle associazioni. La notazione di Crow's Foot permette solamente di rappresentare delle cardinalità (0,1), (1,1), (0,n) e (1,n). Tutte le cardinalità diverse da queste devono essere espresse a parole nella documentazione, come nel nostro caso con l'associazione tra "Cliente_Immobile" e "Visita".

Va anche sottolineato il caso in cui si presentino degli attributi sulle associazioni. Con la traduzione nello schema logico potrebbe essere deciso di inglobare negli attributi in una delle due entità, oppure di trasformare l'associazione in una nuova entità. Nella notazione Crow's foot non è possibile inserire degli attributi sulle associazioni, dunque il modello ci porta a creare fin da subito una nuova entità con quegli attributi. Ciò potrebbe comportare una traduzione errata dell'associazione e la conseguente creazione di tabelle non necessarie all'interno del database. Nella notazione UML (Unified Modeling Language) non si possono assegnare attributi alle associazioni ma vanno create delle classi di associazione, che descrivono le proprietà di un'associazione e sono collegate ad essa tramite una linea tratteggiata.

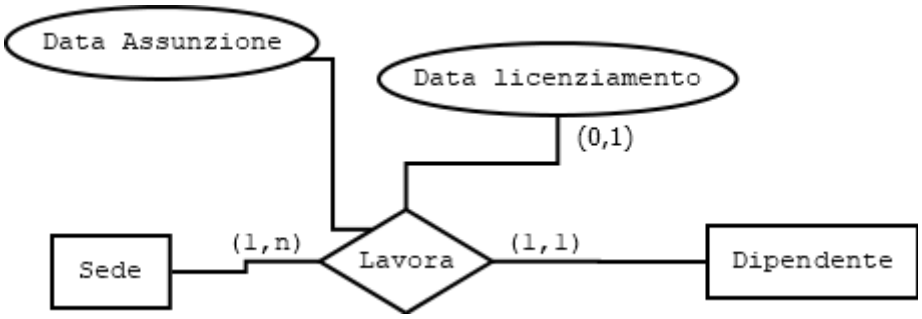


Figura 10 Attributi sull'associazione con la notazione di Chen

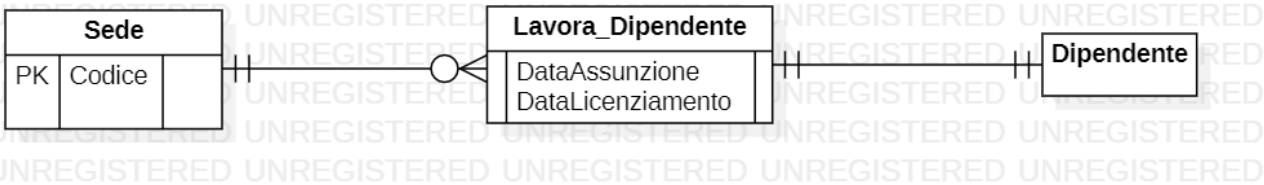


Figura 11 Necessità di creare l'entità Lavora_Dipendente

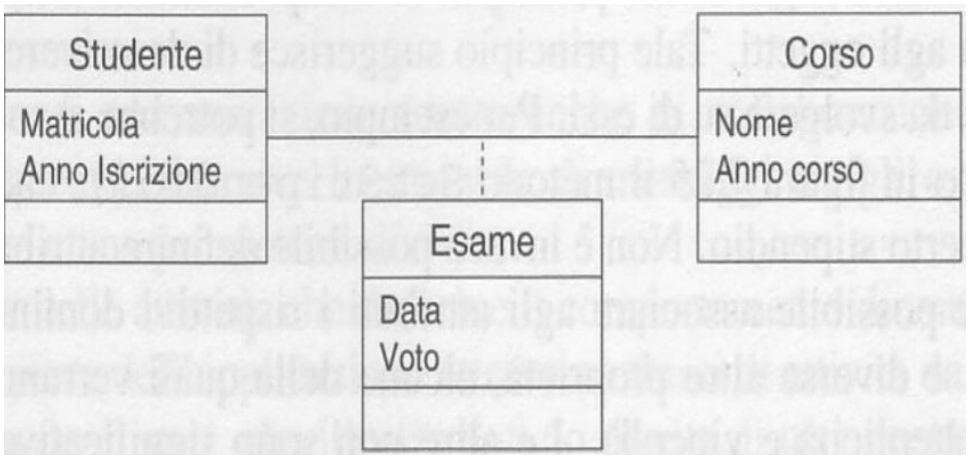


Figura 12 Creazione della classe di associazione in UML

Inoltre non è possibile rappresentare nella notazione Crow's foot gli attributi composti. Per inserirli nel modello è necessario subito creare una nuova entità oppure riportare gli attributi che costituiscono l'attributo composto all'interno dell'entità in questione, perdendo, di conseguenza, la forte capacità espressiva della notazione di Chen. Nell'esempio riportato in fig. 13, se inserissimo gli attributi all'interno dell'entità Persona, ognuno dei cinque attributi semplice può avere la sua cardinalità che però è indipendente da quella degli altri.

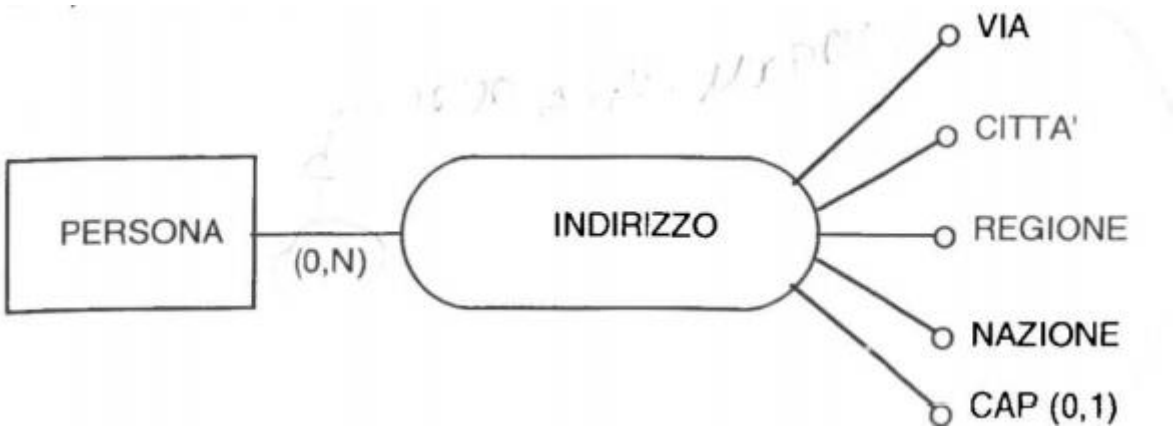


Figura 13 Attributo composto nella notazione di Chen

Un'ulteriore differenza fondamentale da rimarcare è l'assenza di associazioni n-arie (un'associazione che connette più di due entità) nella notazione Crow's Foot. Il progettista è costretto a reificare per collegare due entità, per poi collegare l'entità appena creata tramite un'associazione (1,n) alla terza entità.

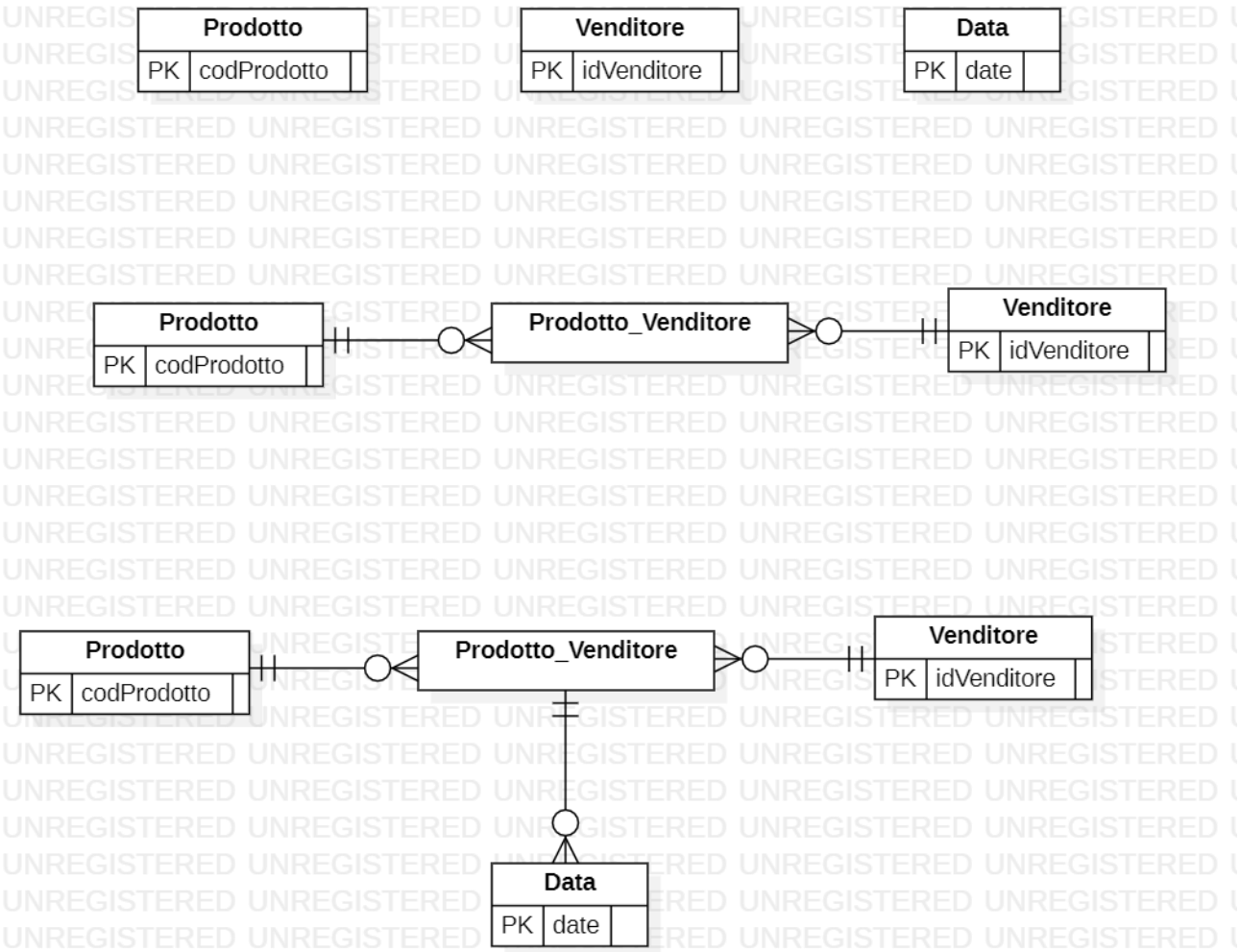


Figura 14 Passaggi per rappresentare un'associazione n-aria

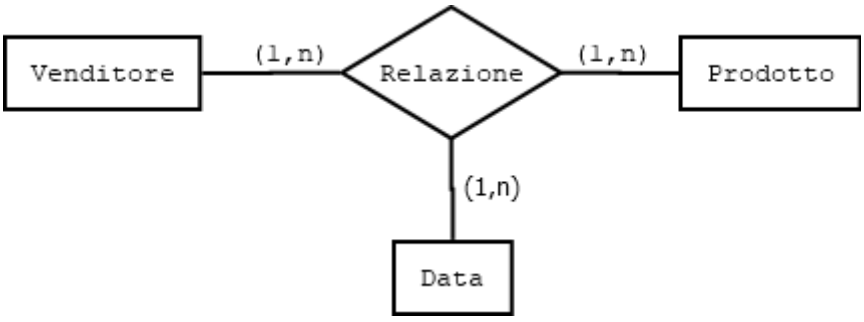


Figura 15 Associazione n-aria con notazione di Chen

Questo obbligo dettato dall'assenza di notazione facilita la traduzione standard di una associazione n-aria. È doveroso però ricordare che è possibile, estendendo la regola "Traduzione con due

relazioni" data per le associazioni binarie uno-a-uno e uno-a-molti, praticare una traduzione non standard di un'associazione n-aria:

Un'associazione n-aria R tra E1, E2, …, En può essere compattata in una delle entità Ei che partecipa con molteplicità unitaria (cioè max-card(Ei, R) = 1) includendo in Ei gli attributi di R e le chiavi primarie di tutte le altre entità come foreign key; tali foreign key saranno vincolate ad essere NOT NULL se e solo se min-card(Ei, R) = 1. Ogni chiave primaria di Ej != Ei tale che Ej partecipa con molteplicità unitaria (cioè max-card (Ej, R) = 1) sarà chiave alternativa.

È possibile considerare un esempio di associazione ternaria:

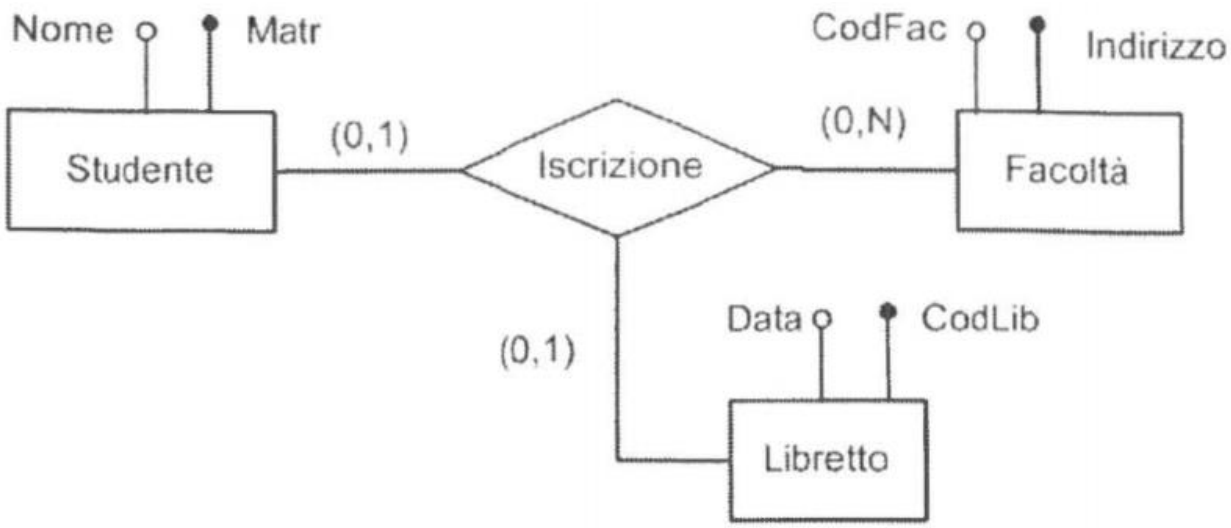


Figura 16 Esempio associazione ternaria

TRADUZIONE STANDARD:

Libretto (<u>CodLib</u>, Data)

Studente (<u>Matr</u>, Nome)

Facoltà (<u>CodFac</u>, Indirizzo)

Indirizzo (<u>Matr</u>, CodFac, CodLib)

AK: CodLib

FK: Matr **REFERENCES** Studente

FK: CodFac **REFERENCES** Facoltà NOT NULL

FK: CodLib **REFERENCES** Libretto NOT NULL

ESEMPIO DI TRADUZIONE NON STANDARD:

Libretto (CodLib, Data)

Facoltà (CodFac, Indirizzo)

Studente (Matr, Nome, CodFac, CodLib)

CodLib UNIQUE

FK: CodFac **REFERENCES** Facoltà

FK: CodLib **REFERENCES** Libretto

Per evitare di avere uno studente iscritto senza libretto e viceversa, bisognerà inserire il vincolo in SQL tramite: CHECK ((CodFac IS NULL AND CodLib IS NULL) OR (CodFac IS NOT NULL AND CodLib IS NOT NULL)).

2.6 REVERSE ENGINEERING OPERATO DA DUE SOFTWARE DIFFERENTI SUL CODICE SQL

Dopo l'implementazione del codice SQL risulta fondamentale produrre una corretta e completa documentazione per facilitare la comprensione del database sia ai potenziali clienti che ai progettisti che dovranno modificarlo in futuro. La documentazione di un database prevede la descrizione delle tabelle ma anche diagrammi che permettono di visionare velocemente le relazioni presenti tra le tabelle.

Purtroppo Microsoft SQL Server non include alcun metodo per generare documentazione dello schema della tabella. Per la realizzazione di tale diagramma si fa ricorso all'utilizzo di due software disponibili su Internet: "StarUML" e "db forge studio for SQL Server". StarUML è un tool UML che supporta la generazione di codice (forward engineering) e il reverse engineering. Il software applicativo si può scaricare gratuitamente da Internet al seguente link:

“**https://staruml.io/download**”. Db forge studio for SQL Server si può scaricare al seguente link: “**https://www.devart.com/dbforge/sql/studio/”**. È disponibile una versione di prova gratuita di 30 giorni ma, in seguito, per continuare ad utilizzare l’applicazione sono disponibili varie versioni a partire da 59.99$. DB Forge si presenta come un GUI tool per il development, il management e l’amministrazione dei database.

Per StarUML è necessario accedere agli Extension Manager, nella sezione tool, e scaricare il pacchetto “Entity Relationship DataModel Generation” per utilizzare la semantica e la notazione propria del modello E/R. Il software dispone di molti pacchetti per permettere il forward e il reverse engineering per molti linguaggi di programmazione (Java, Python…etc). Questi strumenti permettono all’utente di completare la documentazione del proprio progetto con il minimo sforzo.

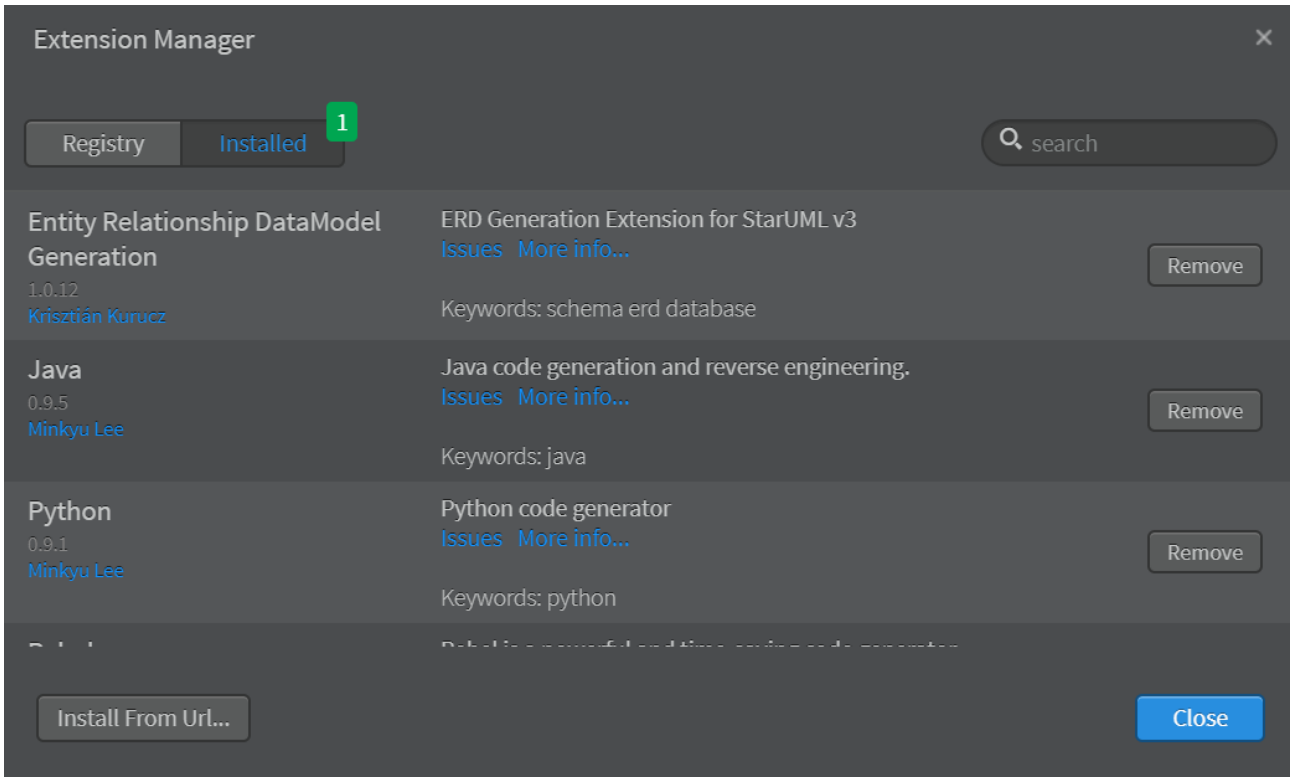


Figura 17 Finestra Extension Manager in StarUML

2.6.1 CONNESSIONE AL DATABASE

Per permettere ad entrambi i software applicativi di accedere ai database del nostro DBMS, è necessario instaurare una connessione al server. I dati che bisogna fornire sono: Username, Password, Server IP, Table Schema, Server Port e Table Catalog. Dopo aver fornito questi dati, sarà possibile visualizzare tutte le tabelle del database e potrà essere effettuato un refresh ogniqualvolta venga effettuata una modifica al database.

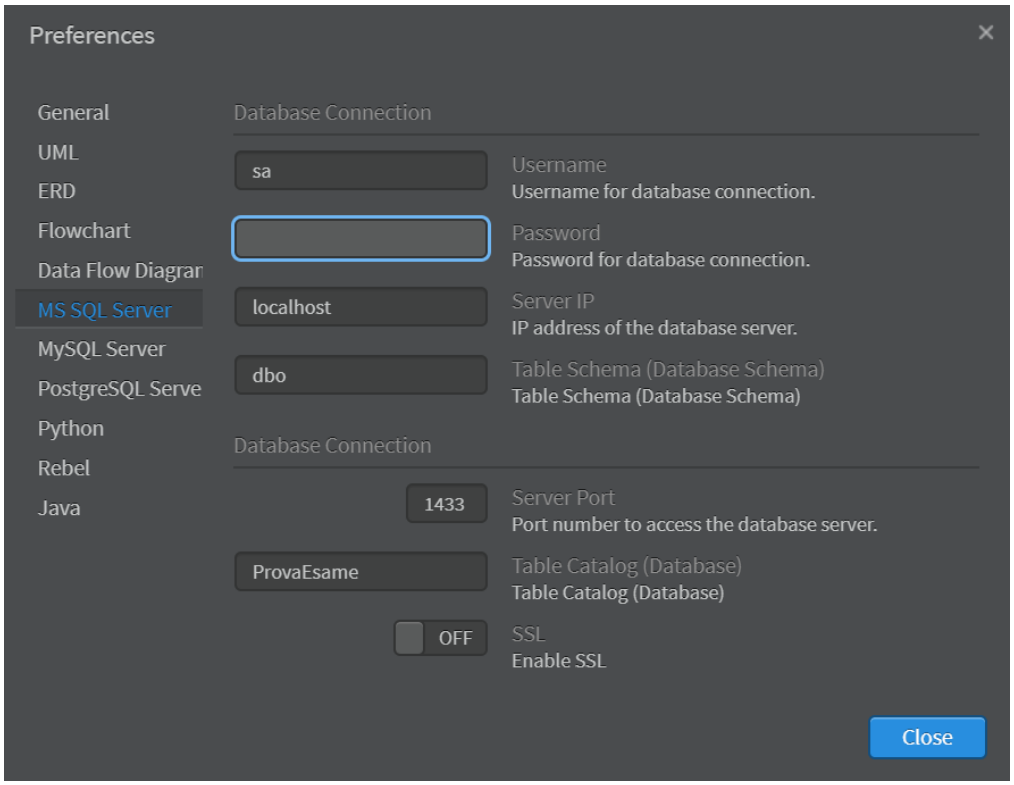


Figura 18 Connessione database da StarUML

Dopo aver concluso la connessione, con un movimento di drag e drop, sarà possibile trasportare e visualizzare gli attributi, i vincoli e le relazioni tra le tabelle del database. Si è voluto riportare il modello con la notazione Crow's foot di due software applicativi differenti per evidenziare delle criticità che potrebbero sorgere nell'utilizzo di questi. Il controllo della realizzazione di un corretto reverse engineering è fondamentale in questi casi. Gli errori sono sorti nel lavoro di StarUML e, ad un occhio poco attento, potrebbero sfuggire, soprattutto con modelli che rappresentano grandi database.

2.6.2 **DATABASE DIAGRAM GENERATO TRAMITE STAR UML**

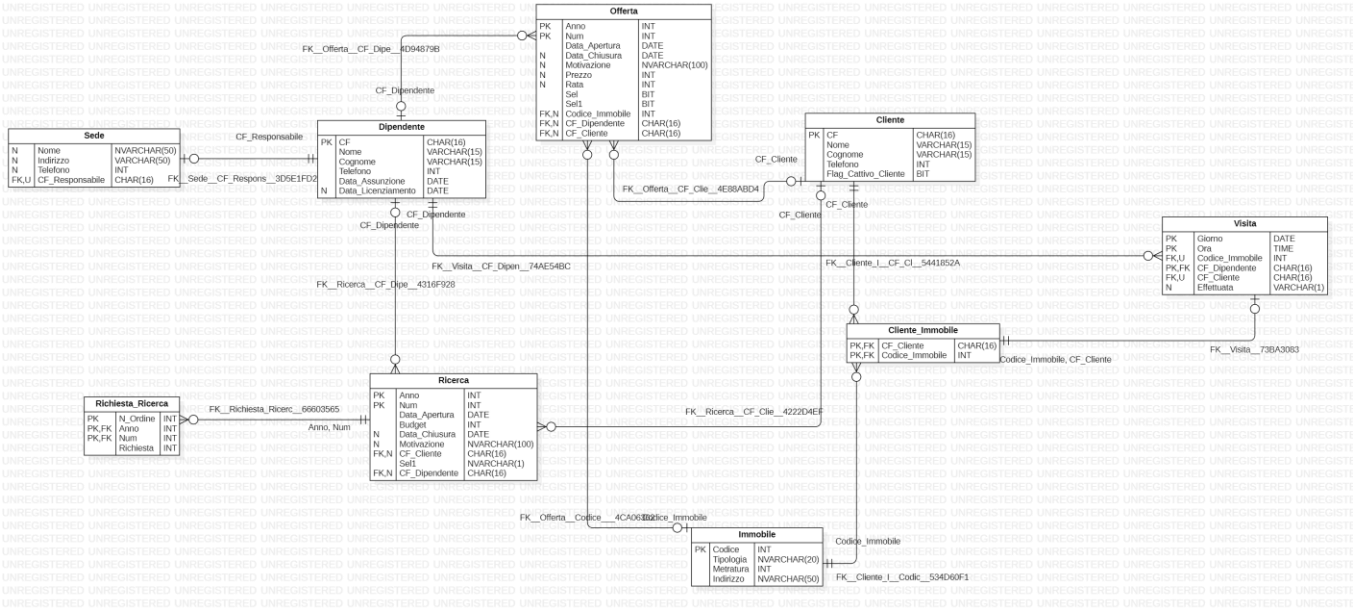


Figura 19 Database diagram generato con Star UML

2.6.3 DATABASE DIAGRAM GENERATO TRAMITE DB FORGE STUDIO FOR SQL SERVER

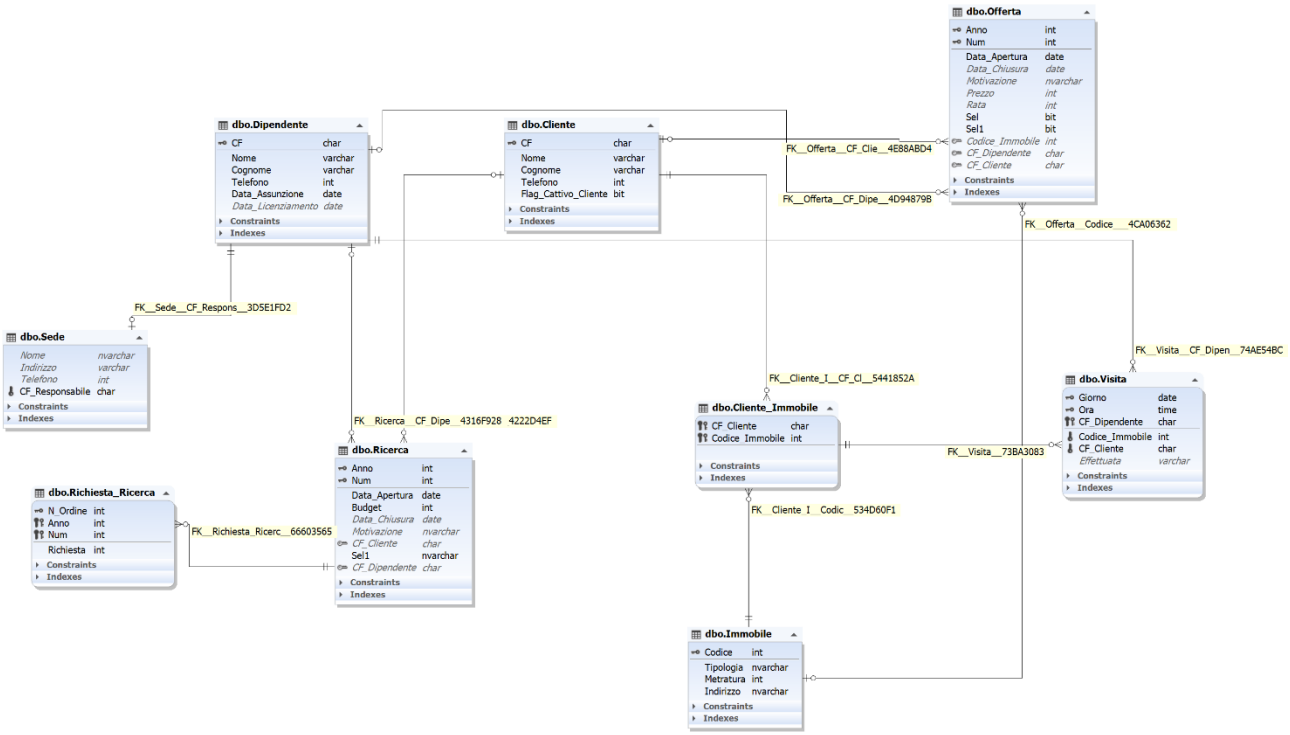


Figura 20 Database diagram generato con DB Forge Studio

2.6.4 DIFFERENZE ED ANALOGIE TRA I DUE DIAGRAMMI

Entrambi i diagrammi mostrano le chiavi primarie e le relazioni tra le tabelle tramite delle associazioni. I due diagrammi riportano per quasi tutte le relazioni tra le tabelle la giusta cardinalità, tranne l'associazione tra "Cliente_Immobile" e "Visita".

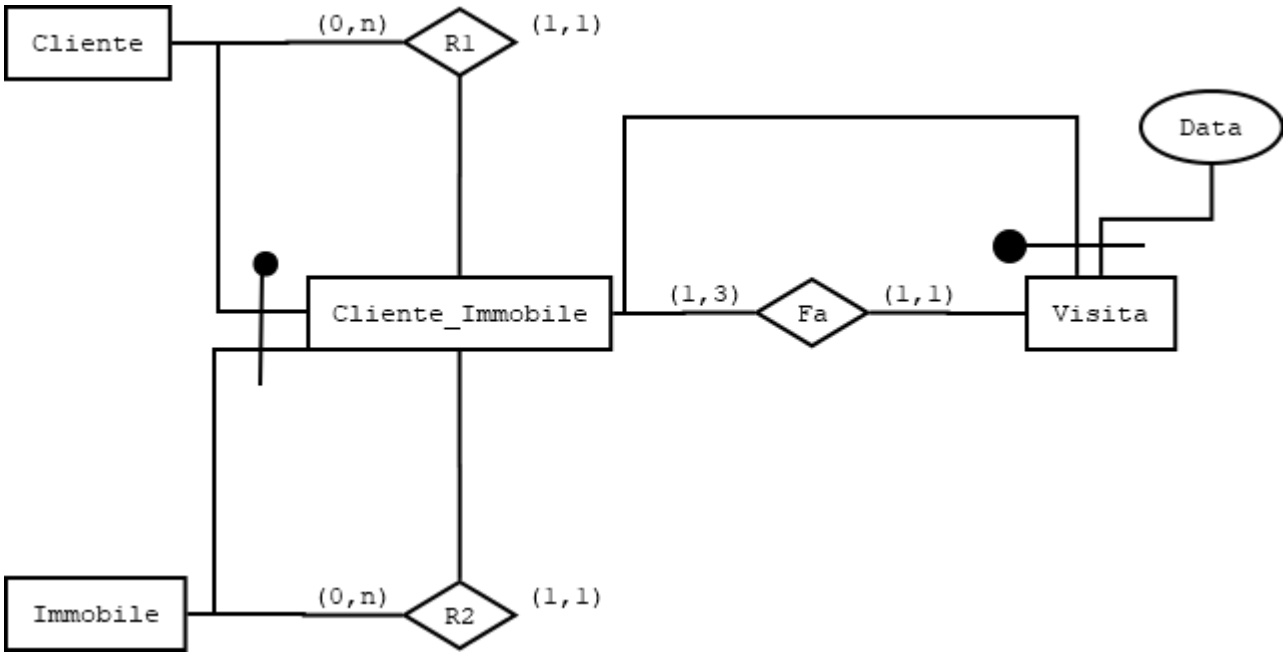


Figura 21 Relazione tra Cliente_Immobile e Visita

CODICE SQL DELLE DUE TABELLE:

```
create table Cliente_Immobile(
        CF_Cliente char(16),
        Codice_Immobile integer,
        FOREIGN KEY (Codice_Immobile) REFERENCES Immobile(Codice),
        FOREIGN KEY (CF_Cliente) REFERENCES Cliente(CF),
        primary key (CF_Cliente, Codice_Immobile)
);

create table Visita(
        Giorno date,
        Ora time,
        Codice_Immobile integer not null,
        CF_Dipendente char(16),
        CF_Cliente char(16) not null,
        Effettuata varchar,
        unique(Codice_Immobile, Giorno, CF_Cliente),
```

FOREIGN KEY (CF_Cliente, Codice_Immobile) REFERENCES Cliente_Immobile(CF_Cliente, Codice_immobile),

FOREIGN KEY (CF_Dipendente) REFERENCES Dipendente(CF),

primary key (Giorno, Ora, CF_Dipendente)

);

La tabella Visita presenta come primary key la combinazione degli attributi (Giorno, Ora, CF_Dipendente). Inoltre la tabella ha uno **UNIQUE** constraint che copre gli attributi (Codice_Immobile, Giorno, CF_Cliente). Dato che il vincolo non è legato solamente alla chiave primaria della tabella Cliente_Immobile ma anche al giorno in cui viene effettuata la visita, la relazione dovrebbe essere di uno-a-molti tra Cliente_Immobile e Visita.

Figura 22 Relazione realizzata da StarUML

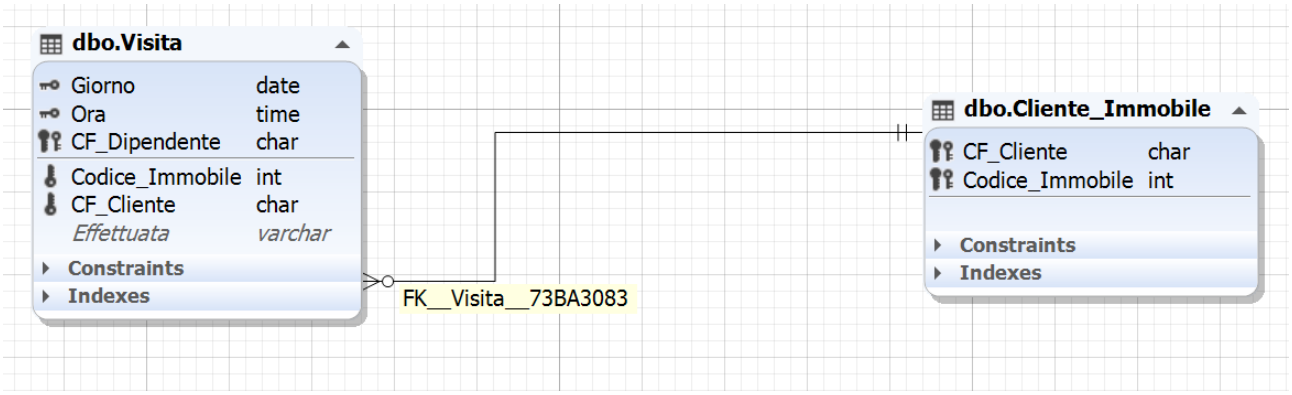


Figura 23 Relazione realizzata con DB Forge Studio

Il software StarUML commette un errore grossolano dimostrando di non saper riconoscere uno UNIQUE constraint basato su più attributi appartenenti a tabelle differenti.

2.6.5 ULTERIORE ERRORE COMMESSO DA STARUML

Si prende in considerazione questo modello E/R che rappresenta un'associazione binaria uno ad uno. Viene tradotta a livello logico, implementata su SQL Server e viene applicato il reverse engineering tramite i due differenti tool:

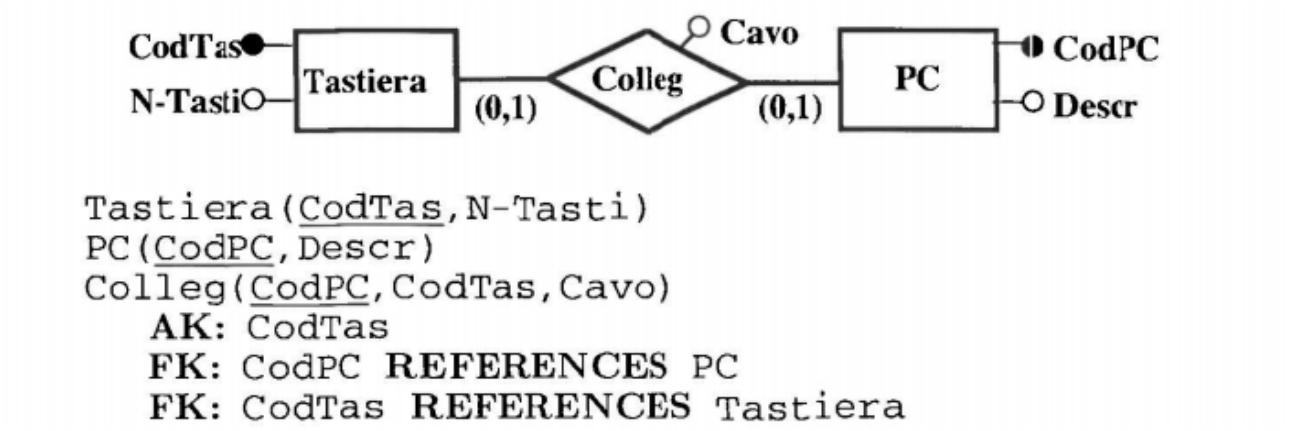


Figura 24 Diagramma E/R esempio

CODICE SU SQL SERVER:

```
create table Tastiera(
        CodTas integer PRIMARY KEY,
        NTasti integer
)

create table PC(
        CodPC integer PRIMARY KEY,
        descr varchar(100)
)
```

```
create table Colleg(
        CodPC integer PRIMARY KEY,
        CodTas integer not null,
        UNIQUE(CodTas),
        FOREIGN KEY(CodPC) REFERENCES PC(CodPC),
        FOREIGN KEY(CodTas) REFERENCES Tastiera(CodTas),
)
```

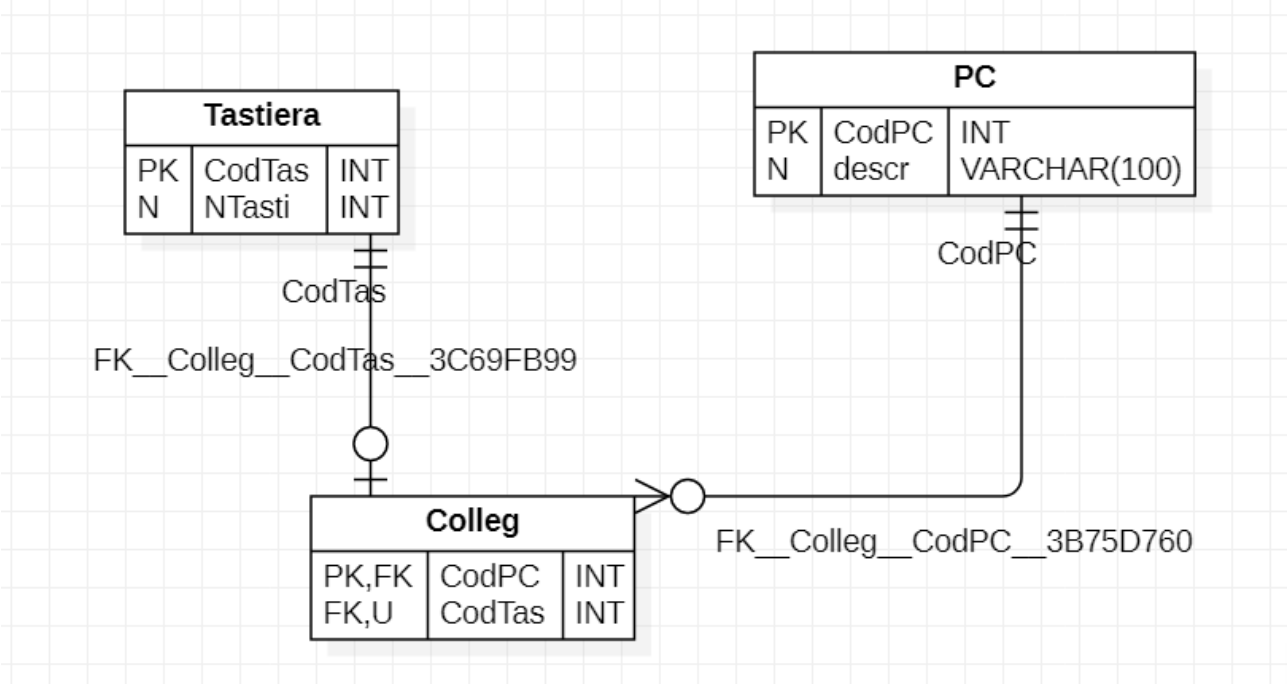


Figura 25 Reverse engineering effettuato su StarUML

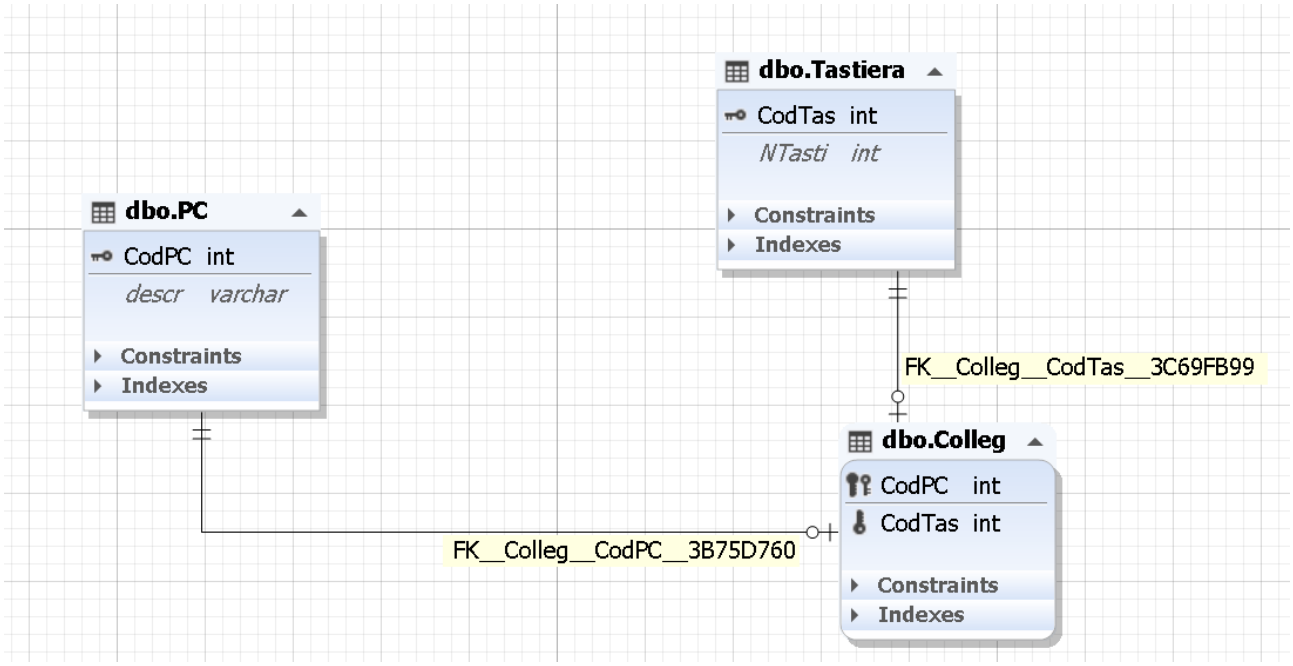
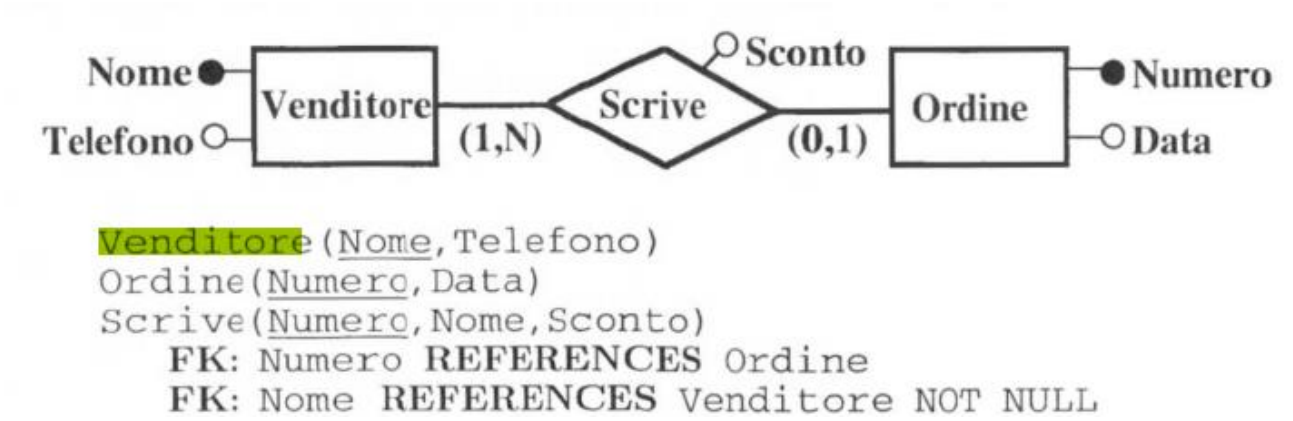


Figura 26 Reverse engineering effettuato da DB Forge Studio

È possibile vedere che vi è una differenza nella rappresentazione della relazione tra la tabella PC e la tabella Collegamento. Il tool DB Forge effettua correttamente il lavoro di reverse engineering riportando su entrambe le relazioni la cardinalità (0,1).

StarUML, invece, effettua un lavoro errato, riportando una cardinalità (0,n) tra la tabella PC e la tabella Collegamento. Tale errore non è isolato. L'errore viene causato spesso da relazioni binarie uno-a-uno, la cui traduzione comporta spesso l'introduzione di una foreign key in cui la colonna (o le colonne) che costituisce la primary key nella tabella referenziante, rappresenta anche la primary key nella tabella referenziata.

Si riporta un esempio simile:



```
create table Venditore(
        Nome varchar(10) PRIMARY KEY,
        telefono integer
)

create table Ordine(
        Numero integer PRIMARY KEY,
        Giorno datetime
)

create table Scrive(
        Numero integer PRIMARY KEY,
        Nome varchar(10) not null,
        Sconto integer,
        FOREIGN KEY(Numero) REFERENCES Ordine(Numero),
        FOREIGN KEY(Nome) REFERENCES Venditore(Nome),
)
```

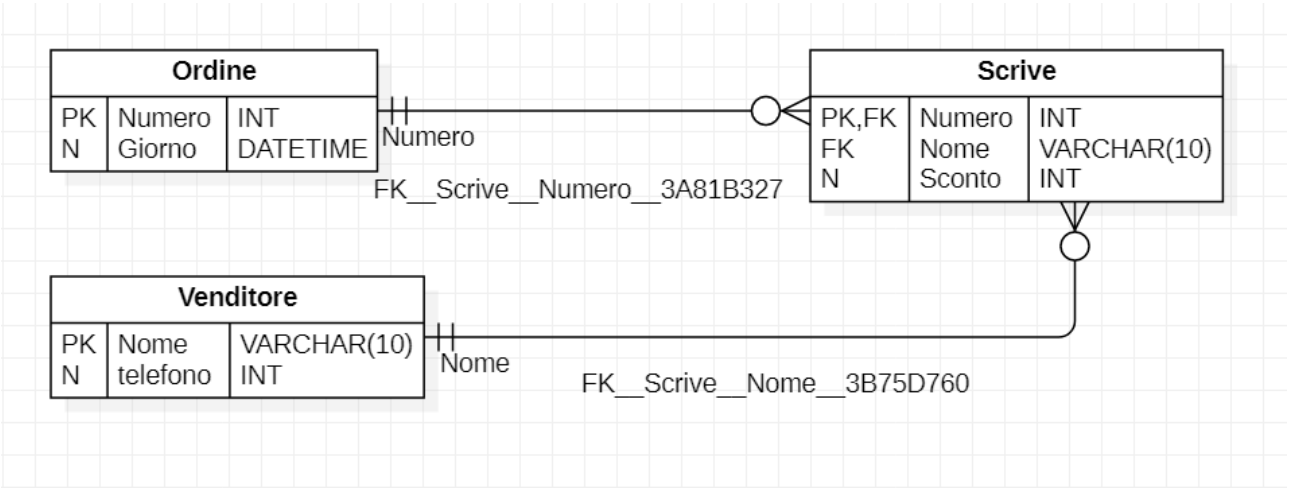


Figura 27 Reverse engineering effettuato su StarUML

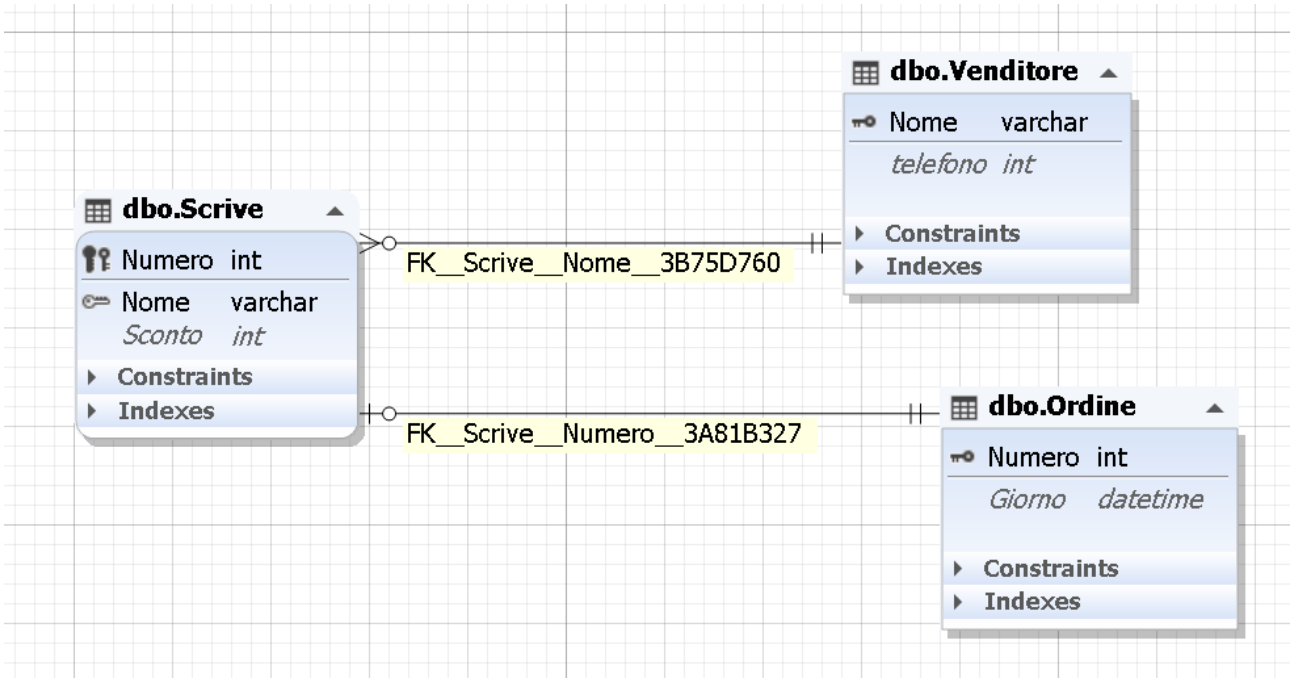


Figura 28 Reverse engineering effettuato su DB Forge Studio

L'errore, in questo caso, viene commesso nella rappresentazione della relazione tra la tabella Ordine e la tabella Scrive. L'attributo "Numero" (int) costituisce la primary key della tabella "Ordine". Nella tabella "Scrive" è presente la foreign key verso la tabella "Ordine" e tale attributo rappresenta l'intera primary key della tabella "Scrive".

3 SQL SERVER: FORWARD AND REVERSE ENGINEERING

3.1 REVERSE ENGINEERING

SQL Server mette a disposizione lo strumento **Database Diagram Tool** che effettua la realizzazione del diagramma a partire da un database esistente. Tale diagramma presenta le tabelle del database e le relazioni tra esse.

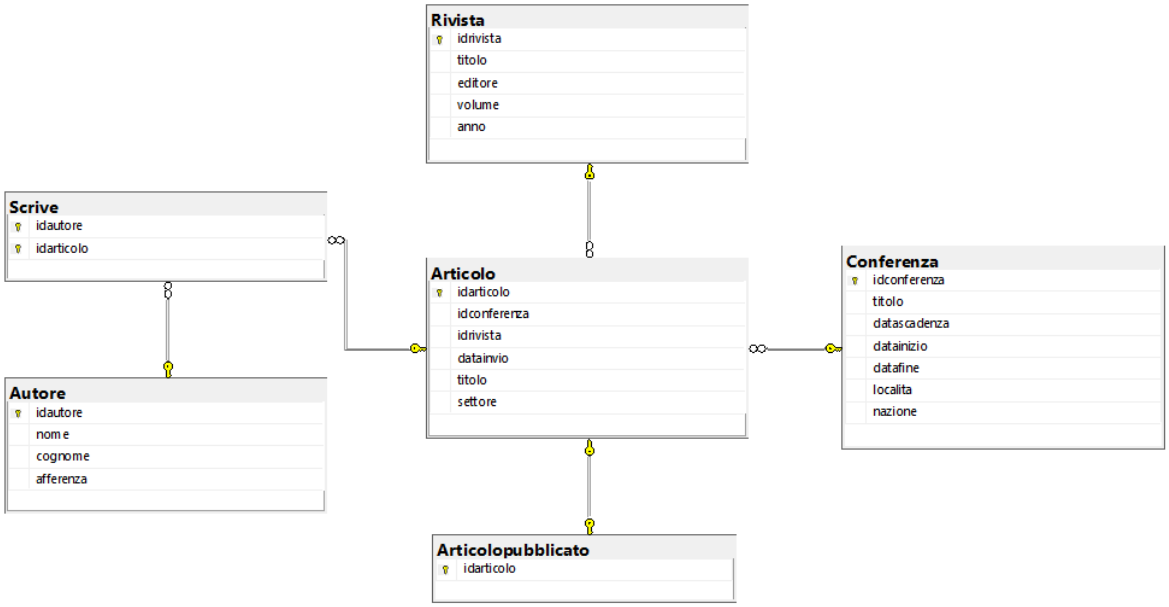


Figura 29 Diagramma realizzato da Database Diagram Tool

3.2 FORWARD ENGINEERING

Attraverso il **Database Diagram Tool** è possibile generare database, tabelle e relazioni tra tabelle senza scrivere righe di script ma utilizzando la progettazione grafica offerta dal tool. In pochi passaggi è possibile infatti creare il database (fig. 30) e inserire all'interno le opportune tabelle (fig. 31 e fig. 32). È possibile indicare la chiave primaria ed eventuali colonne che possono assumere valori NULL.

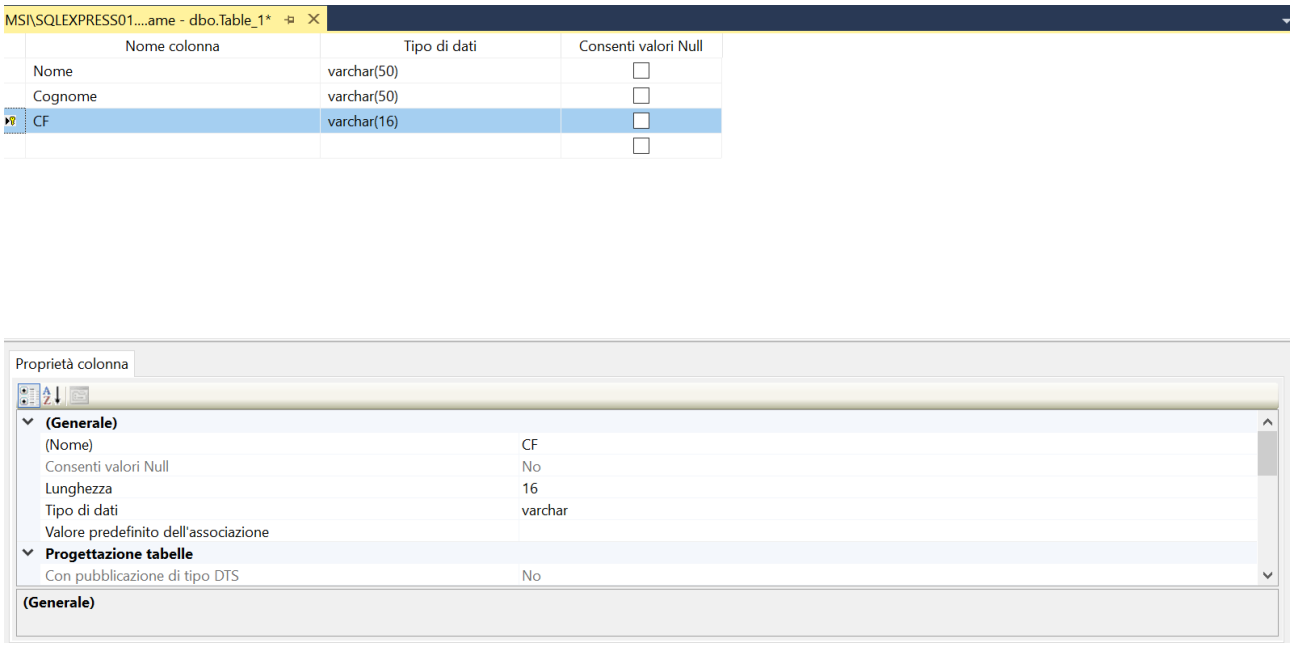


Figura 30 Creazione Database

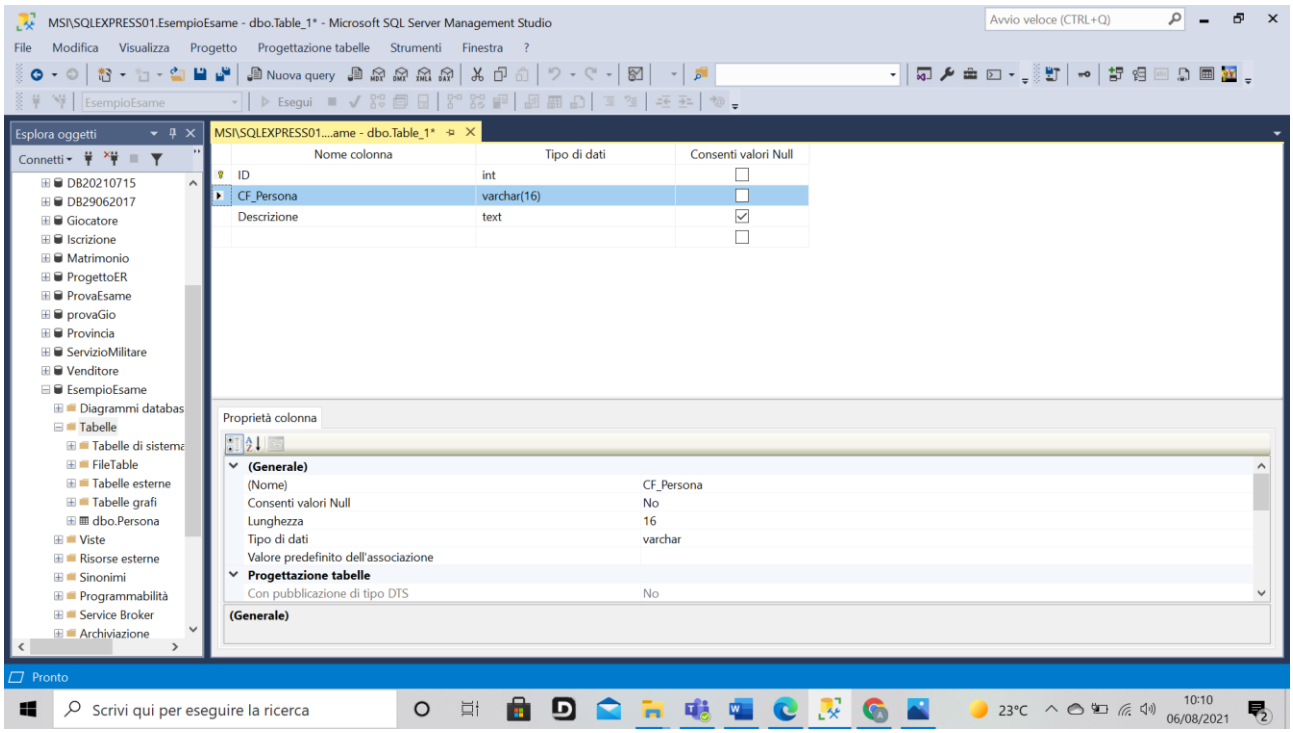


Figura 31 Creazione tabella Persona

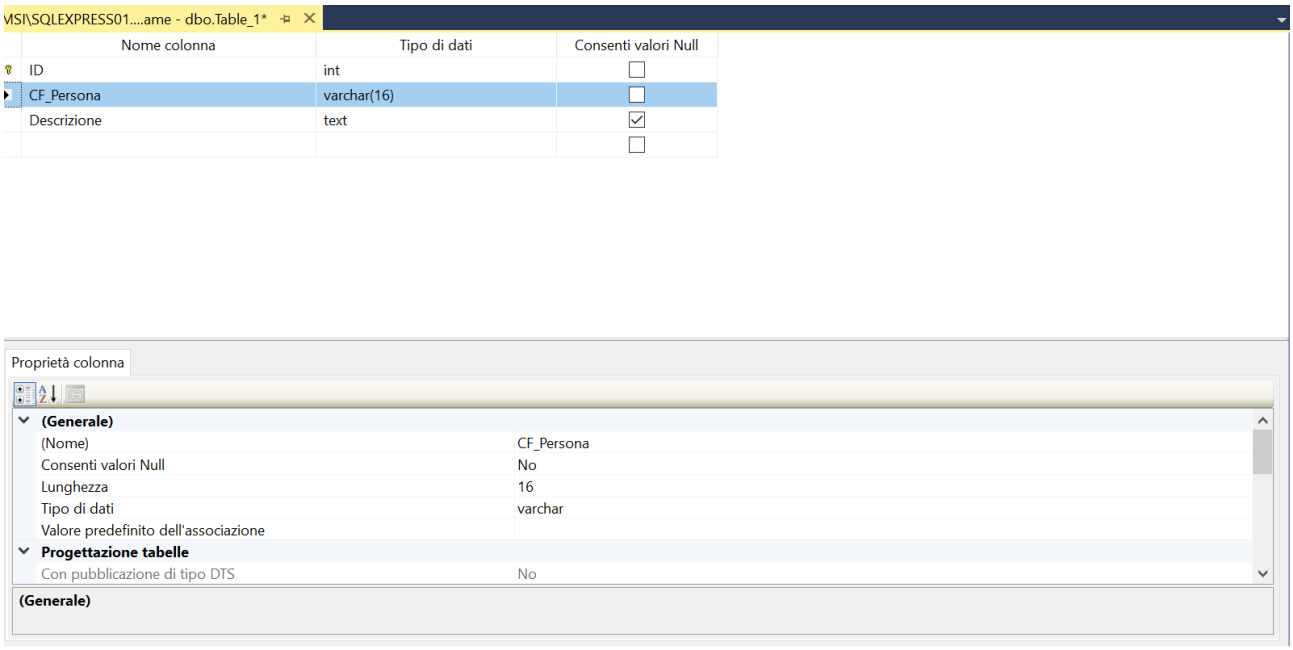


Figura 32 Creazione tabella Prenotazione

L'inserimento delle foreign key avviene attraverso la sezione "Relazioni" presente su ogni colonna creata. Va inserita la tabella di chiave primaria e la tabella di chiave esterna con le rispettive colonne che sono coinvolte nella foreign key.

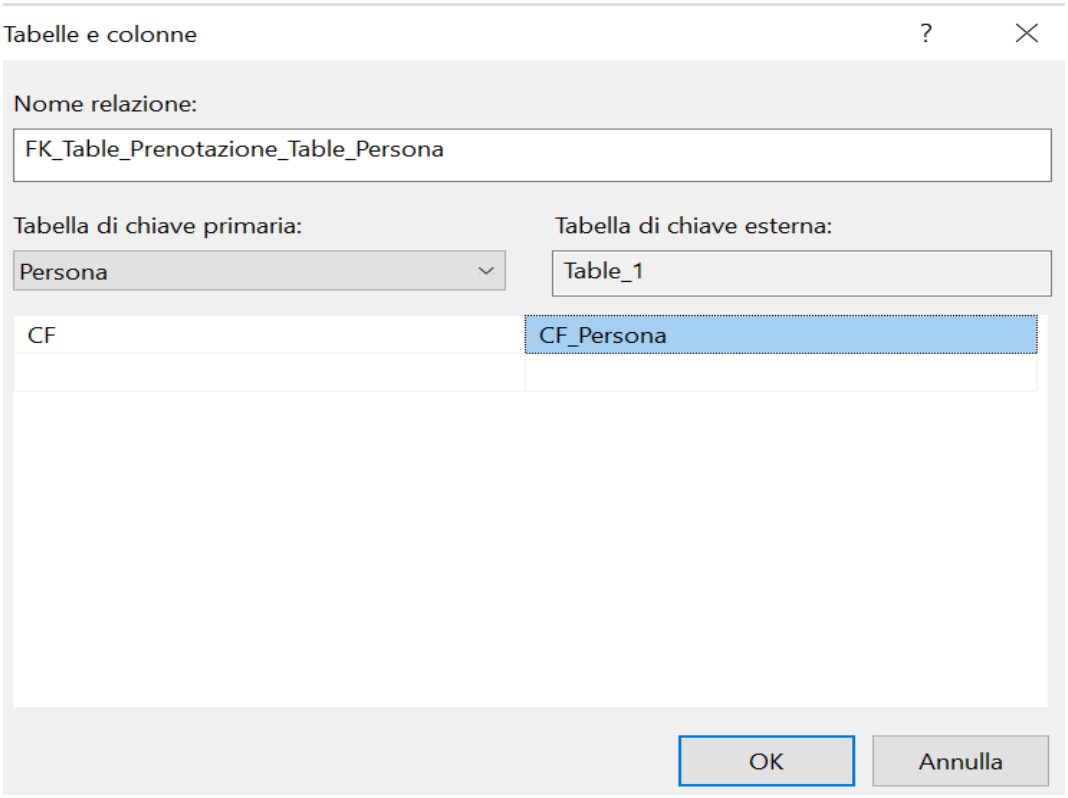


Figura 33 Inserimento foreign key

Il database creato attraverso questi passaggi presenta questo diagramma di database (fig. 34).

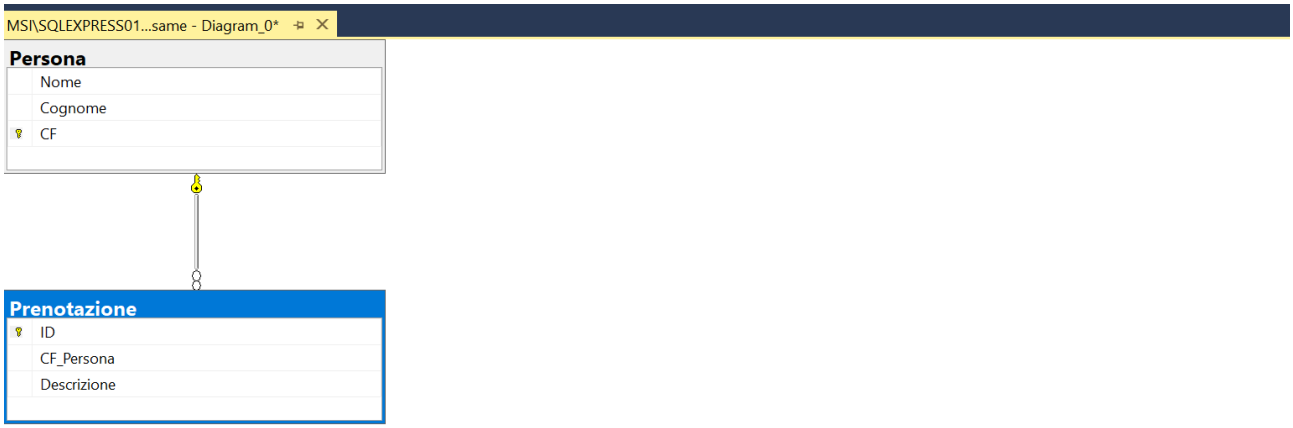


Figura 34 Database diagram creato con forward engineering

Sebbene sia possibile creare database, tabelle e relazioni senza scrivere script, SQL Server non presenta un potente tool di modellazione grafica di database.

4 MySQL WORKBENCH: FORWARD AND REVERSE ENGINEERING

MySQL WORKBENCH si presenta come uno strumento di accesso alla progettazione e modellazione di database Visual per database relazionali di server MySQL. L'obiettivo del workbench è quello di fornire un'interfaccia ordinata e semplice per lavorare con i database. Permette infatti la creazione di modelli di dati fisici e la modifica di database esistenti con tecniche di forward/reverse engineering. Gli strumenti di modellazione e progettazione supportano la creazione di oggetti come tabelle, viste, stored procedure, trigger, ecc.

4.1 MODELLO EER

Il modello che viene utilizzato dal workbench viene chiamato Modello EER (Enhanced entity-relationship model). Tale modello ingloba tutti i concetti del modello ER.

Componenti addizionali:

- Specializzazione / generalizzazione
- Categorie (UNION)
- Ereditarietà di attributi e relazioni
- Aggregazione

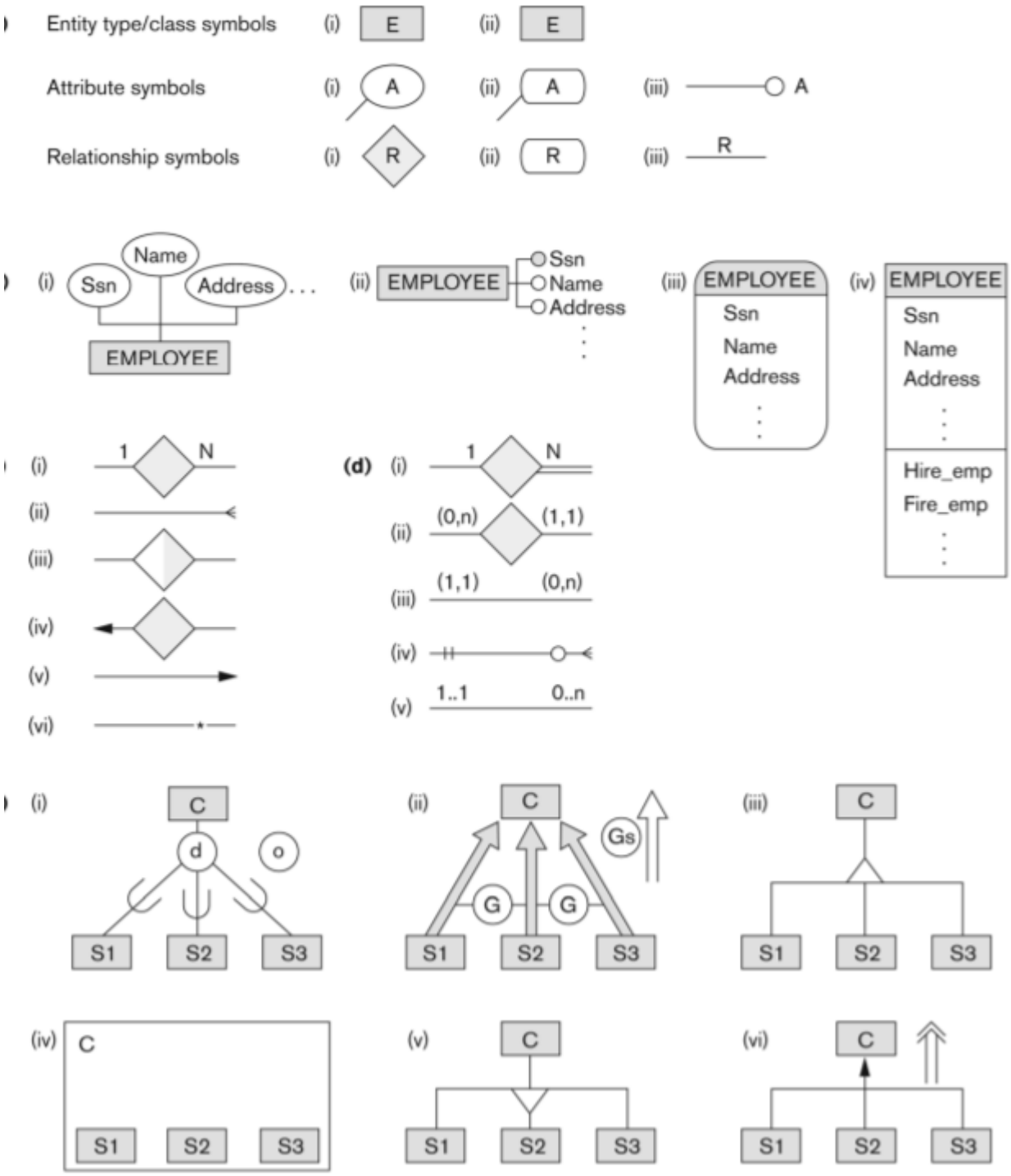


Figura 35 Notazione del modello EER

ESEMPIO DI AGGREGAZIONE

L'aggregazione è un processo che rappresenta la relazione tra un intero oggetto e i suoi componenti. Riesce ad astrarre la relazione tra oggetti rendendola un unico oggetto.

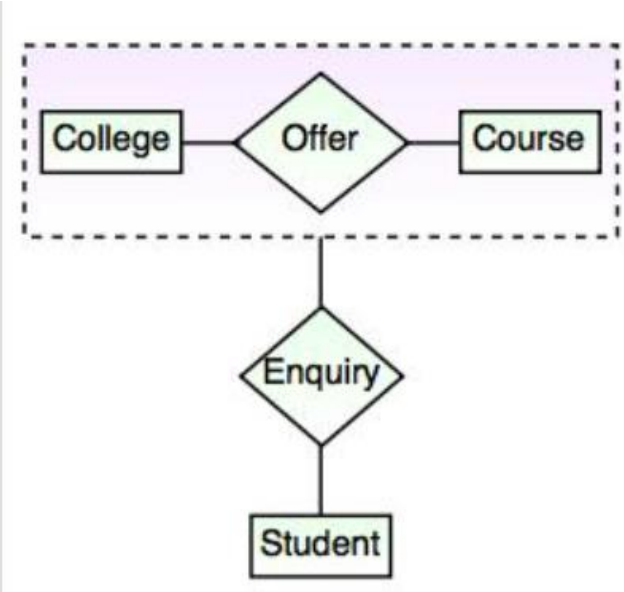


Figura 36 Esempio di aggregazione

La relazione tra College e Course si sta comportando come un'unica entità nei confronti della relazione con Student.

4.1.1 TIPI DI RELAZIONI PRESENTI

Sono quattro le differenti opzioni di relazioni presenti nel workbench. Bisogna tenere a mente nel leggere il modello che la cardinalità è opposta rispetto al tradizionale modello ER con la notazione di Chen.

Figura 37 Tipi di relazioni presenti

4.1.2 RELAZIONI IDENTIFICANTI E RELAZIONI NON IDENTIFICANTI

Nel momento in cui viene instaurata una relazione identificante o una relazione non identificante, nella tabella dipendente compare una chiave esterna che coincide con la chiave primaria della tabella riferita.

- Relazione identificante: la chiave della tabella primaria partecipa alla chiave primaria della tabella dipendente.

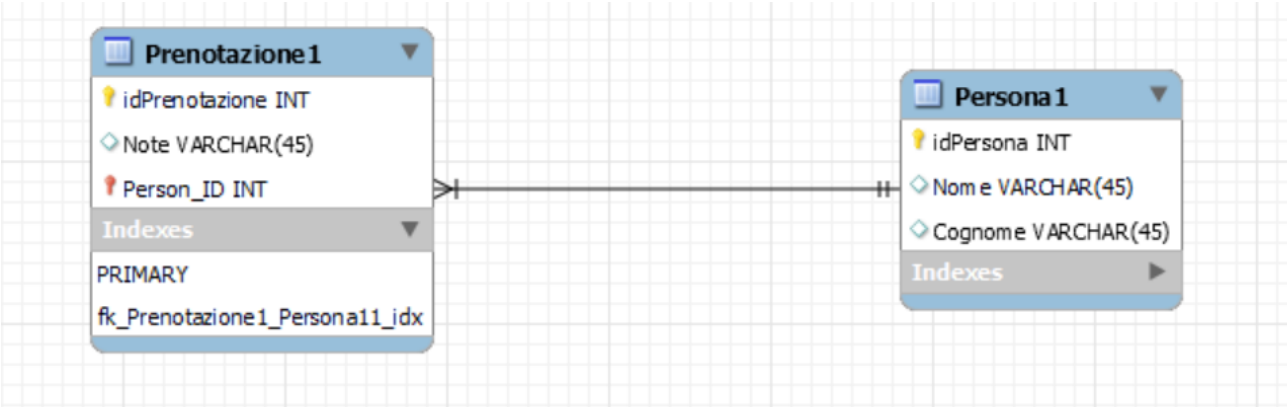


Figura 38 Person_ID partecipa alla chiave primaria di Prenotazione

- Relazione non identificante: la chiave della tabella primaria non partecipa alla chiave primaria della tabella dipendente.

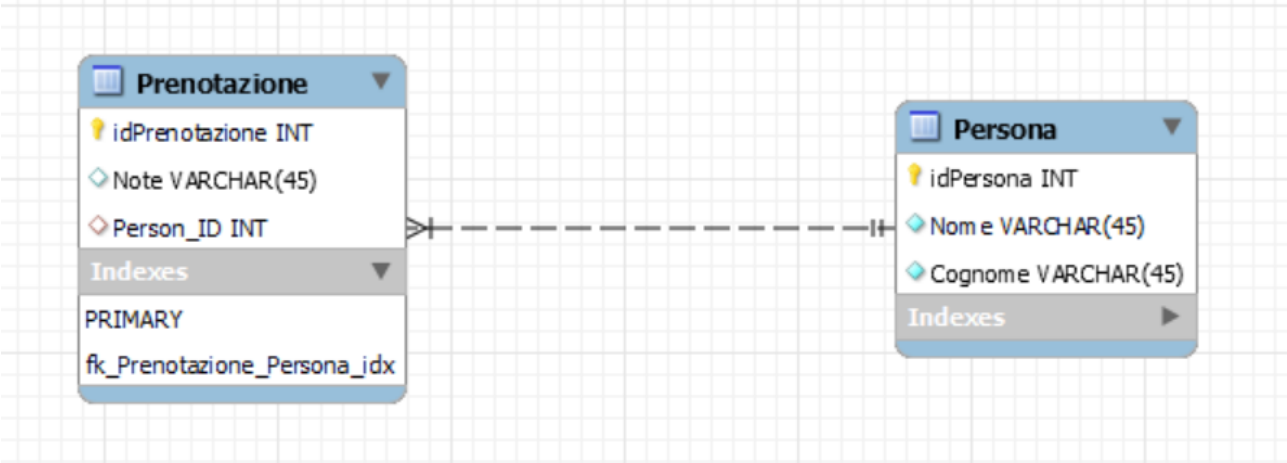


Figura 39 Person_ID non partecipa alla chiave primaria di Prenotazione

4.2 LIMITI NELLA PROGETTAZIONE GRAFICA DI MySQL WORKBENCH

4.2.1 COMPOSITE ALTERNATE KEY NON RAPPRESENTABILE DIRETTAMENTE

Se una tabella ha più di una chiave candidata, una di esse diventerà la chiave primaria e le altre saranno chiavi alternative. Sia le chiavi primarie che quelle alternative possono essere composte da più di un attributo.

Per la rappresentazione grafica di primary key composte da più di un attributo non vi è alcun ostacolo. Il problema sorge per le chiavi alternative. Un modo di indicare le chiavi alternative è quello di inserire in coppia il vincolo di "NOT NULL" con quello di "UNIQUE". Tuttavia la rappresentazione grafica non riesce a rappresentare una chiave alternativa che comprenda i vari attributi.

La soluzione è quella di inserire a mano nello script il vincolo di "UNIQUE" che raccoglie i vari attributi, inserendo così la desiderata chiave alternativa.

Se prendiamo in considerazione la tabella "Esame" da implementare in questo modo:

ESAME (Cod_Studente, Data, Voto, Cod_Corso)

AK: Cod_Studente, Cod_Corso

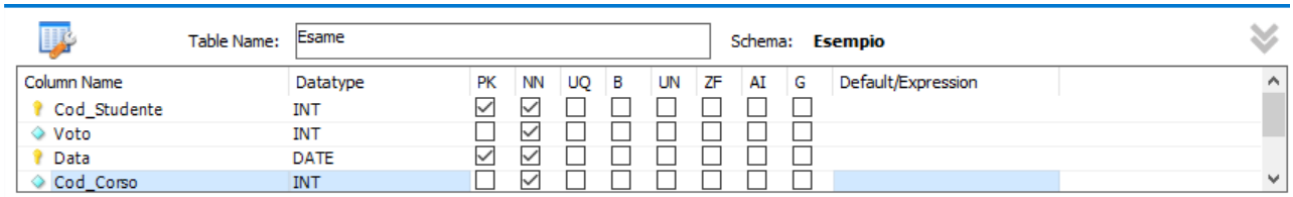


Figura 40 Progettazione grafica della tabella

In fig. 40 è possibile notare l'assenza di modalità di inserimento di chiavi alternative composte da più attributi. Se andassimo a spuntare i vincoli di "NOT NULL" e "UNIQUE" per gli attributi che partecipano alla chiave alternativa, il workbench andrebbe a creare tante chiavi alternative composte da un attributo. L'unico modo per ovviare a tale problema consiste nel modificare lo script di creazione della tabella che si ottiene nella sezione "Forward Engineering" del workbench.

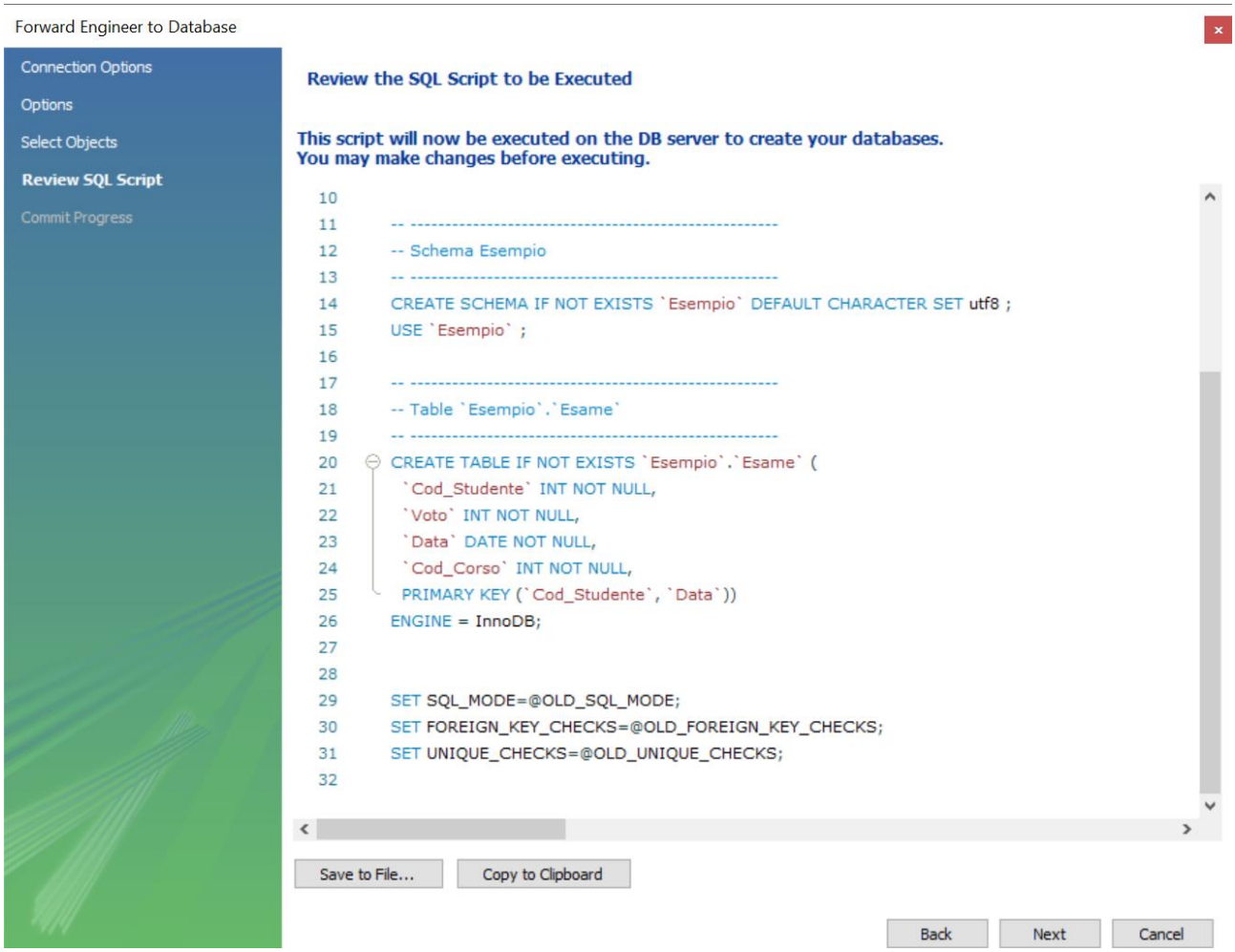


Figura 41 Script senza alternate key

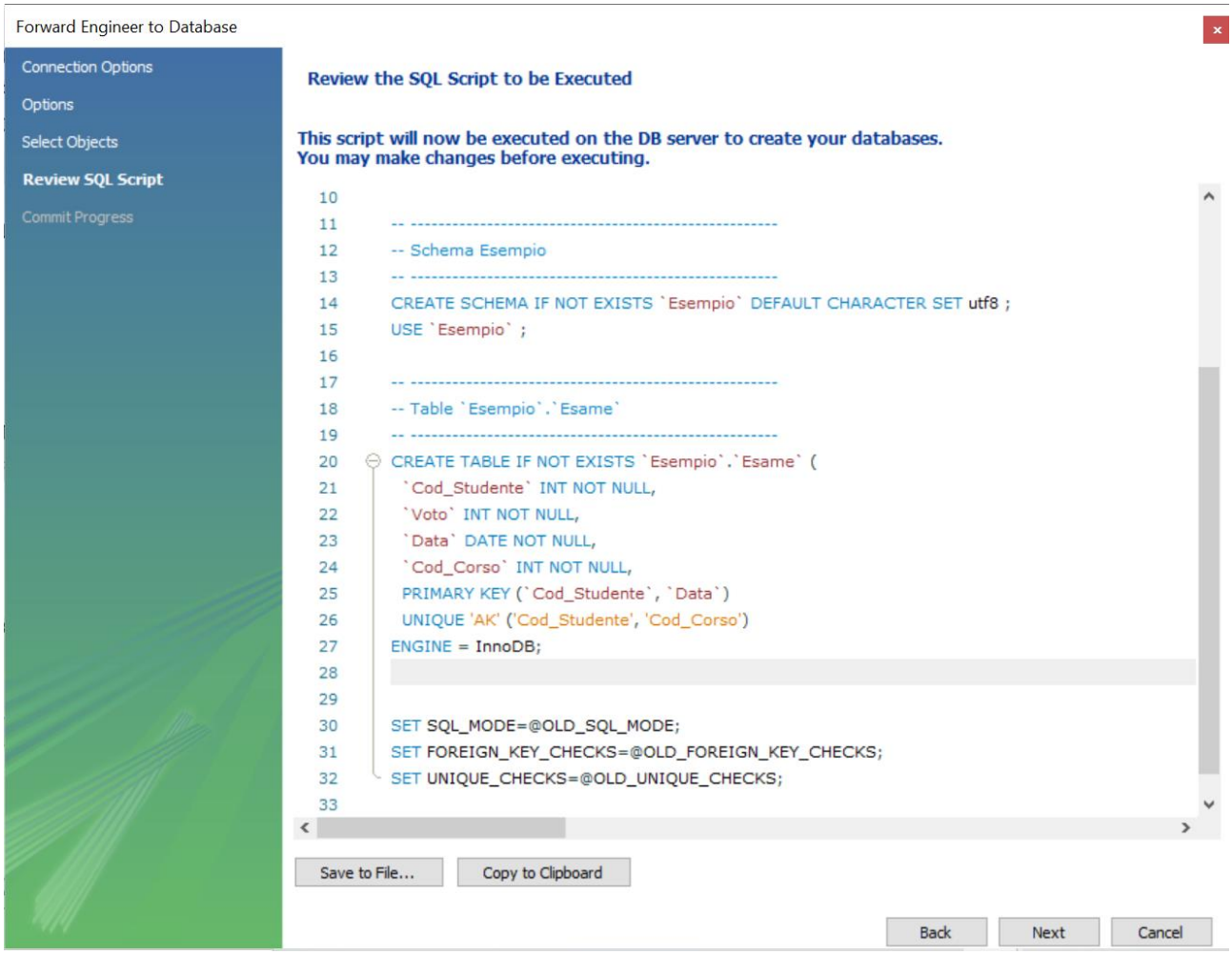


Figura 42 Script di creazione con alternate key

4.2.2 LE ASSOCIAZIONI TERNARIE NON SONO DIRETTAMENTE RAPPRESENTABILI

Come era stato già mostrato nel modello E/R con la notazione Crow's Foot, anche in MySQL Workbench non è possibile rappresentare graficamente le associazioni ternarie.

L'unica soluzione è quella di creare una tabella intermedia collegata alle prime due tabelle con un'associazione uno-a-molti e inserire in essa una chiave composta ed esterna. Infine collegare tale tabella intermedia con la terza tabella con un'associazione uno-a-molti.

4.3 REVERSE ENGINEERING

Dopo aver effettuato la connessione al database, è possibile ottenere il diagramma EER relativo al database esistente.

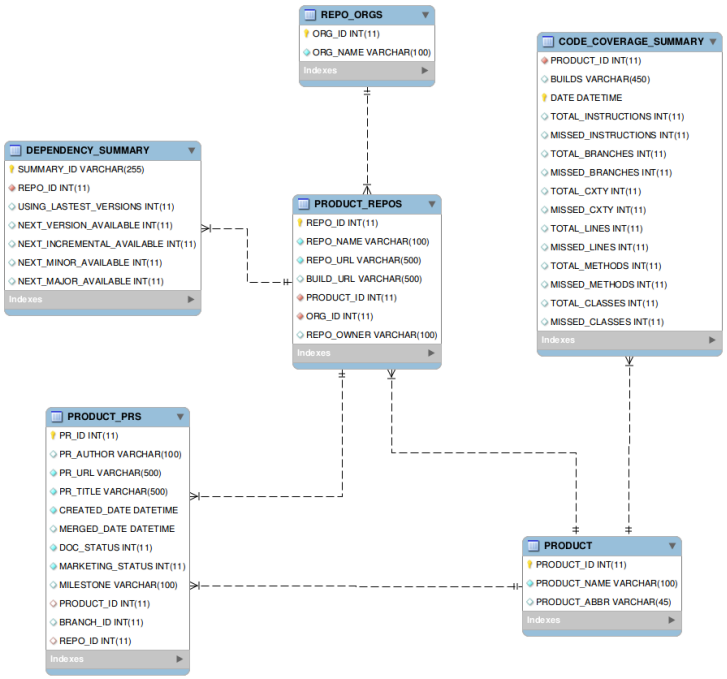


Figura 43Diagramma EER realizzato con il reverse engineering

Il lavoro di reverse engineering effettuato da MySQL Workbench e da SQL Server non presentano notevoli differenze.

5 MODELLO UML PER LA RAPPRESENTAZIONE DI BASI DI DATI

Il modello UML è un linguaggio grafico per la modellazione di sistemi software basati sul paradigma object-oriented. Il modello presenta una varietà di diagrammi permettendo una molteplicità di viste della stessa applicazione. Esso rappresenta un'alternativa al modello E/R per la rappresentazione concettuale dei dati. Viene fatto uso dei **diagrammi delle classi** che descrivono le relazioni tra classi di oggetti dell'applicazione. Oltre al diagramma della classi sono presenti altri diagrammi fondamentali: degli oggetti, di sequenza, degli stati, delle attività, dei casi d'uso.

Alcuni costrutti del tradizionale modello ER non sono previsti nel modello UML. Si può ricorrere a notazioni non standard in alcuni casi. Non sono presenti, ad esempio, gli identificatori interni ed esterni, inoltre non sono ammessi gli attributi composti.

Il diagramma delle classi prevede la presenza di **classi** e di **associazioni** tra esse. Le associazioni binarie vengono rappresentate con **linee** che collegano due classi. Non si possono assegnare attributi alle associazioni: devono essere usate classi di associazione.

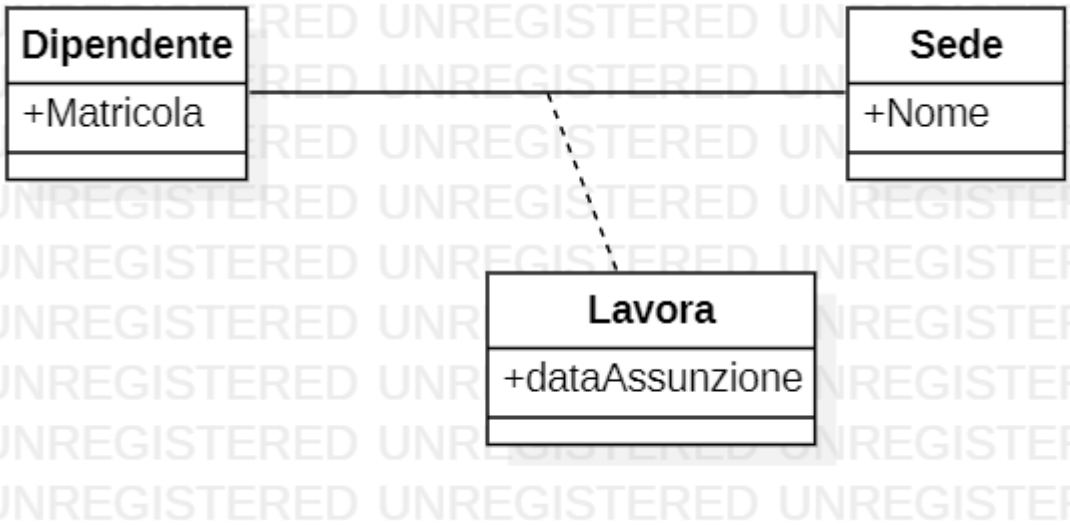


Figura 44 Esempio di classe di associazione

5.1 ASSOCIAZIONI N-ARIE

Le associazione n-arie prevedono la stessa notazione del tradizionale modello E/R, dunque la presenza del rombo che viene collegato alle varie entità. È sempre possibile, per le associazioni di grado superiore al secondo, di eliminare l'associazione tramite l'operazione di reificazione.

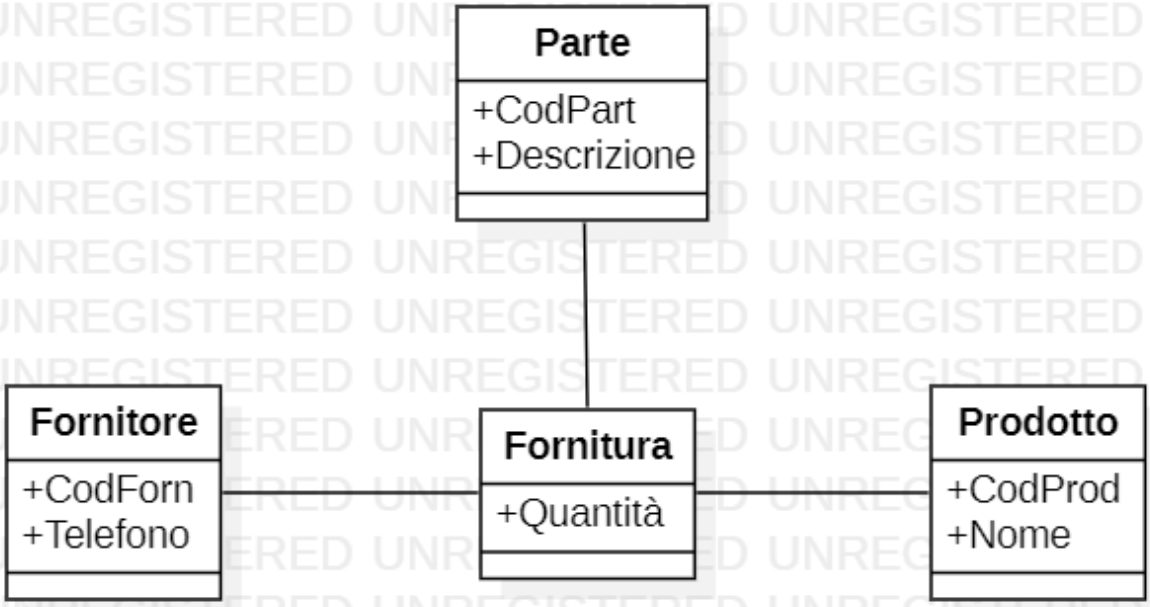


Figura 45 Esempio di reificazione nel modello UML

È possibile indicare con una freccia un verso privilegiato di navigabilità. Inoltre si possono rappresentare delle associazioni che sono aggregazioni di concetti. Il rombo bianco indica un'aggregazione, ovvero una relazione non forte: le classi parte hanno un significato anche senza la classe tutto. Il rombo nero indica invece una composizione, o relazione forte: le classi parte hanno un significato solo in presenza della classe tutto.

Figura 46 Esempio di composizione e di aggregazione

5.2 VINCOLI DI INTEGRITÀ : LE MOLTEPLICITÀ

Come nel modello E/R, è possibile indicare la cardinalità(molteplicità) di partecipazione delle classi alle associazioni. Ci sono diverse convenzioni:

- * sta per 0..N
- (x,y) diventa x..y
- N diventa *, dunque (1,N) diventa (1,*)
- 1 sta per (1,1)

Come nel caso della notazione Crow's Foot, la cardinalità di partecipazione della classe viene riportata accanto all'altra classe che partecipa alla relazione (si può vedere in fig. 47).

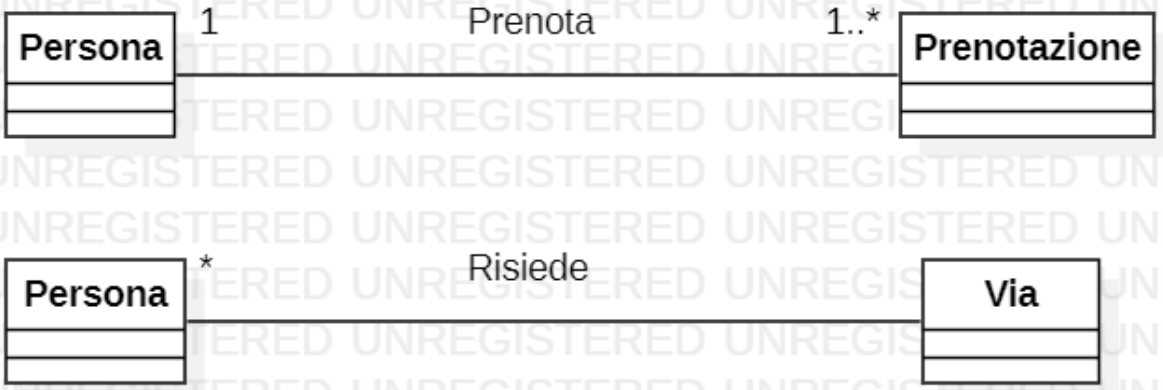


Figura 47 Tipi di cardinalità differenti

5.3 IDENTIFICATORI INTERNI

Dato che UML è un modello basato sull'identità degli oggetti, non esiste una notazione standard per gli identificatori delle classi. Una possibile soluzione è quella di utilizzare un **vincolo utente.** Questo vincolo è esprimibile con delle parentesi graffe accanto all'elemento oggetto del vincolo.

Un identificatore viene espresso con il vincolo utente {id}. Può essere presente un solo identificatore per classe e, in caso di identificatore composto, il vincolo viene espresso per ogni attributo coinvolto.

Figura 48 Identificatori interni nel modello UML

5.4 IDENTIFICATORI ESTERNI

Il modello UML prevede l'utilizzo di **stereotipi** che servono per modellare un concetto che non può essere modellato con i costrutti di base. Gli stereotipi vengono rappresentati da un nome racchiuso tra i simboli << e >>.

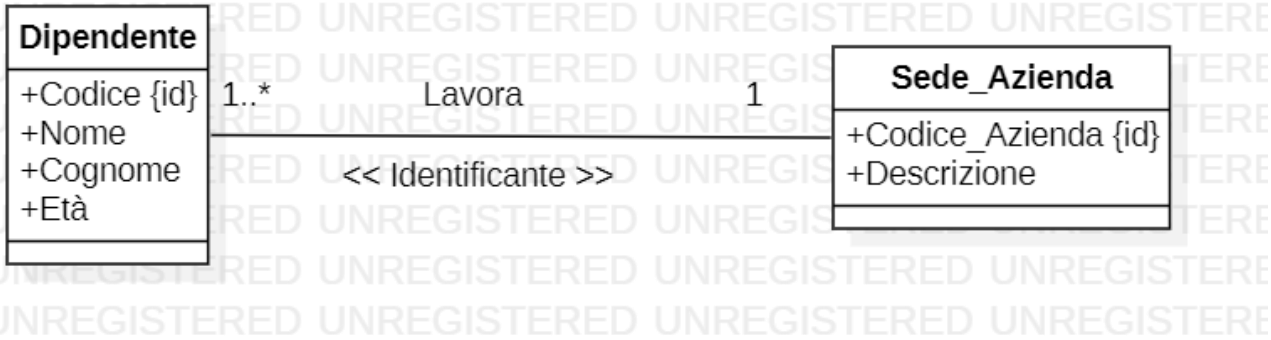


Figura 49 Identificatore esterno nel modello UML

5.5 UTILIZZO DELLE NOTE

È quasi sempre necessario ricorrere all'utilizzo delle note per documentare un diagramma UML. È un semplice commento testuale all'interno di un rettangolo collegato all'elemento cui fa riferimento attraverso una linea tratteggiata.

6 CONCLUSIONI

Il data modeling costituisce un mezzo per migliorare la comunicazione e rendere più efficiente ed efficace il lavoro da svolgere. Nelle aziende, infatti, sta aumentando esponenzialmente l'utilizzo di data modeling Tools. Questo lavoro ha cercato di mettere in luce le potenzialità delle notazioni esistenti (ovvero la loro intrinseca capacità espressiva) e dei tool di data modeling. Sono state messe in risalto le differenze tra le notazioni esistenti; inoltre è stata approfondita l'impossibilità di esprimere alcuni vincoli del database. Si è cercato anche di mettere in risalto il limite nell'utilizzo di alcuni tool, incapaci talvolta di compiere un lavoro corretto e coerente con i vincoli del database. Soprattutto per lavori inerenti a database di grandi dimensioni, è necessario utilizzare tool che non commettano piccoli errori difficili da scovare nei diagrammi prodotti e colpevoli di rendere fallace la documentazione.

7 BIBLIOGRAFIA

- *Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, Maurizio Vincini, "Progetto di Basi di Dati Relazionali. Lezioni ed esercizi", Pitagora Editrice Bologna, 2007.*

- *Lezione 08_Lab2019 Corso Basi di Dati*

- *https://users.dimi.uniud.it/~angelo.montanari/UMLperModConcet.pdf*

- *https://www.cs.purdue.edu/homes/bb/cs448_Spring2014/lecture-files/pdf/ch04-Enhanced%20Entity-Relationship%20(EER)%20Modeling.pdf*