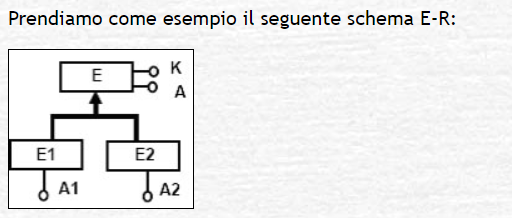
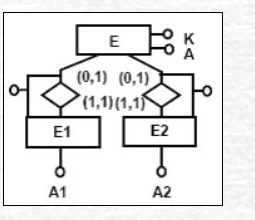
Riferimento: <http://www.dacrema.com/Informatica/gerarchie_elimin.htm>

Per la generalizzazione esistono tre modi di eliminare le gerarchie:

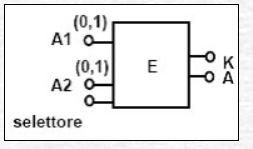


1) Mantenimento delle entità con associazioni



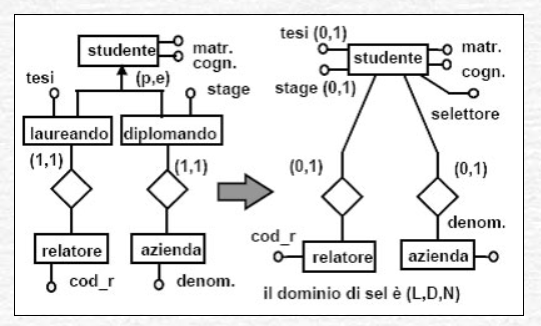
Tutte le entità vengono mantenute. Le entità figlie vengono messe in associazione con l’entità padre e sono identificate esternamente tramite l’associazione. La cardinalità (0,1) indica che per tale associazione l’ entità padre può avere zero o una entità figlio. Mentre (1,1) che l’ entità figlio può avere un solo padre.

2) Collasso verso l’alto

Esso riunisce tutte le entità figlie nell’entità padre.

E1 ed E2 vengono eliminate e le loro proprietà vengono aggiunte all’ entità padre . Gli attributi obbligatori per le entità figlie divengono opzionali per il padre , indicato dalla cardinalità (0,1), con la conseguenza che si *avrà una certa percentuale di valori nulli*. All’ entità ottenuta viene aggiunto un ulteriore attributo (selettore) che specifica se una istanza di E appartiene a una delle sottoentità.

Se la gerarchia era totale ed esclusiva il selettore ha N valori, quante sono le sottoentità. Se la gerarchia era parziale esclusiva il selettore ha N+1 valori; il valore in più serve per le istanze che non appartengono ad alcuna sottoentità. Se avessimo un overlapping occorrerebbero tanti selettori booleani quante sono le sottoentità. Vediamo di seguito un esempio di uno schema in cui è stata collassata la gerarchia verso l’ alto:



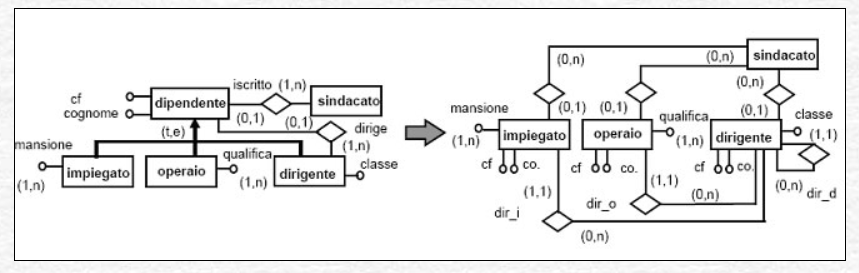
3) Collasso verso il basso

Si elimina l’entità padre trasferendone gli attributi su tutte le entità figlie. Una associazione del padre è replicata, tante volte quante sono le entità figlie *la soluzione è interessante in presenza di molti attributi di specializzazione* (con il collasso verso l’alto si avrebbe un eccesso di valori nulli).

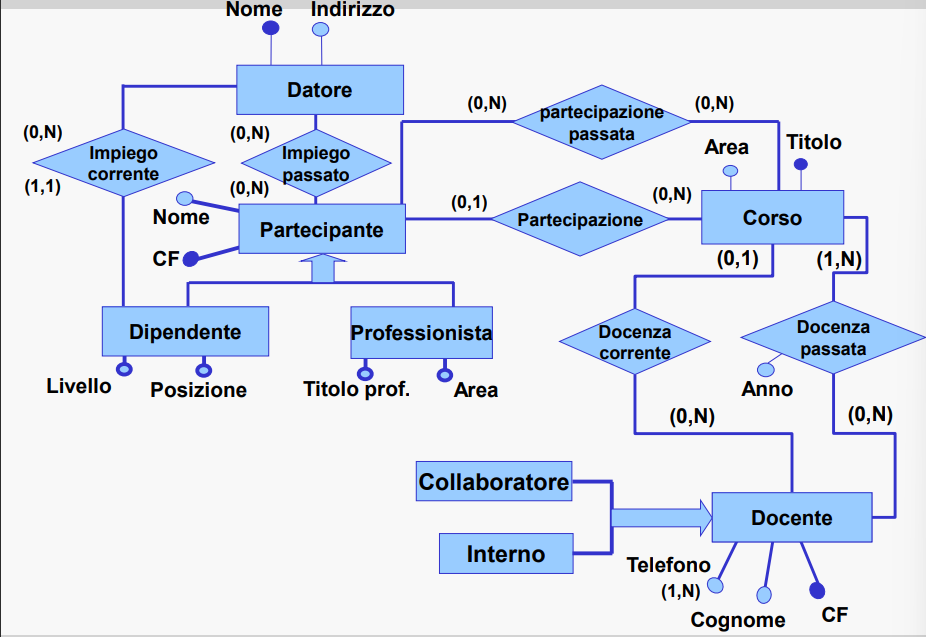
Immagine che contiene testo, orologio

Descrizione generata automaticamente

Si noti che se la copertura è parziale non si può fare il collasso verso il basso. Dove mettere gli E che non sono né E1, né E2 ? Mentre se la copertura è overlapping introduce ridondanza, infatti per una istanza presente sia in E1 che in E2 si rappresentano due volte gli attributi di E.

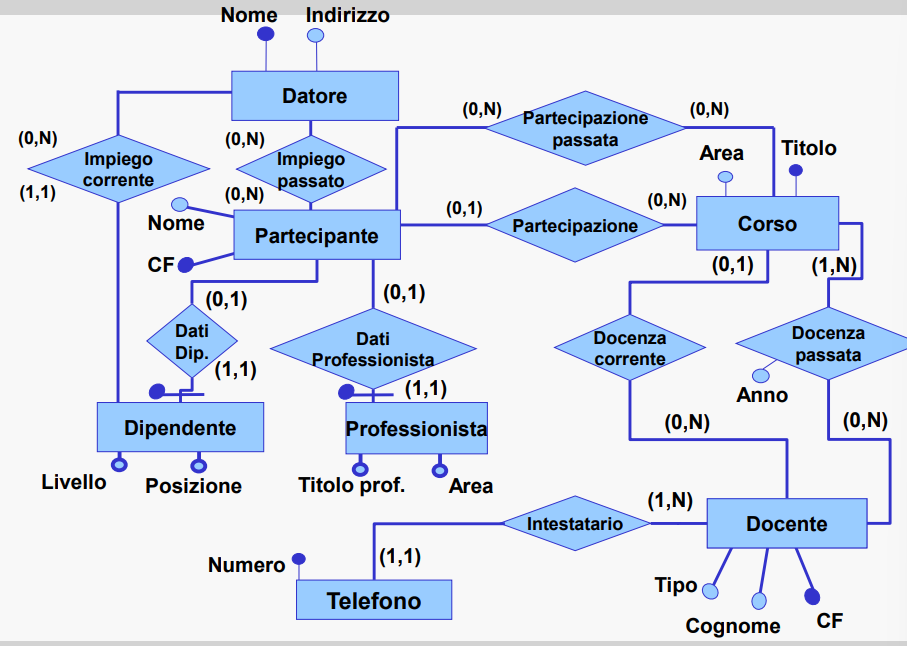
Vediamo di seguito un esempio di uno schema in cui è stata collassata la gerarchia verso il basso:

L’esempio visto in classe parte dallo schema:



E attua una minimizzazione dei valori nulli:

* mantenendo Docenza Corrente e Docenza Passata come relazioni
* usando il mantenimento delle entità con associazioni tra Partecipante e Dipendente/Professionista
* creando per l’attributo multivalore Telefono, un’entità apposita
* mantenendo Partecipazione e Partecipazione Passata come relazioni e similmente anche per Impiego (Corrente e Passato)



Creando come schema

• Partecipante(CF, Nome, Corso-Attuale)

• Partecipante.Corso-Attuale → Corso.Titolo

• Datore(Nome, Indirizzo)

• Corso(Titolo, Area, CF-Docente-Corrente)

• Corso.CF-Docente-Corrente → Docente.CF

• Docente(CF, Cognome,Tipo)

• Docente.Tipo ∈ { Collaboratore, Interno }

• Telefono(Numero, CF-Docente)

• Telefono.CF-Docente → Docente.CF

• Dipendente(CF, Livello, Posizione, Datore)

• Dipendente.CF → Partecipante.CF

• Dipendente.Datore → Datore.Nome

• Professionista(CF,Titolo-Prof, Area)

• Professionista.CF → Partecipante.CF

• ImpiegoPassato(CF-Partecipante,Nome-Datore)

• ImpiegoPassato.CF-Partecipante → Partecipante.CF

• ImpiegoPassato.Nome-Datore → Datore.Nome

• PartecipazionePassata(CF-Partecipante,Titolo-Corso)

• PartecipazionePassata.CF-Partecipante → Partecipante.CF

• PartecipazionePassata.Titolo-Corso → Corso.Titolo

• DocenzaPassata(CF-Docente,Titolo)

• DocenzaPassata.CF-Docente → CF.Docente

• DocenzaPassata.Titolo → Corso.Titolo

Esercizi concreti

Il sistema è condiviso tra diverse istituzioni, di ognuna delle quali è di interesse sapere il nome dell’istituzione, la via e il codice ISTAT del comune in cui ha la sede principale.

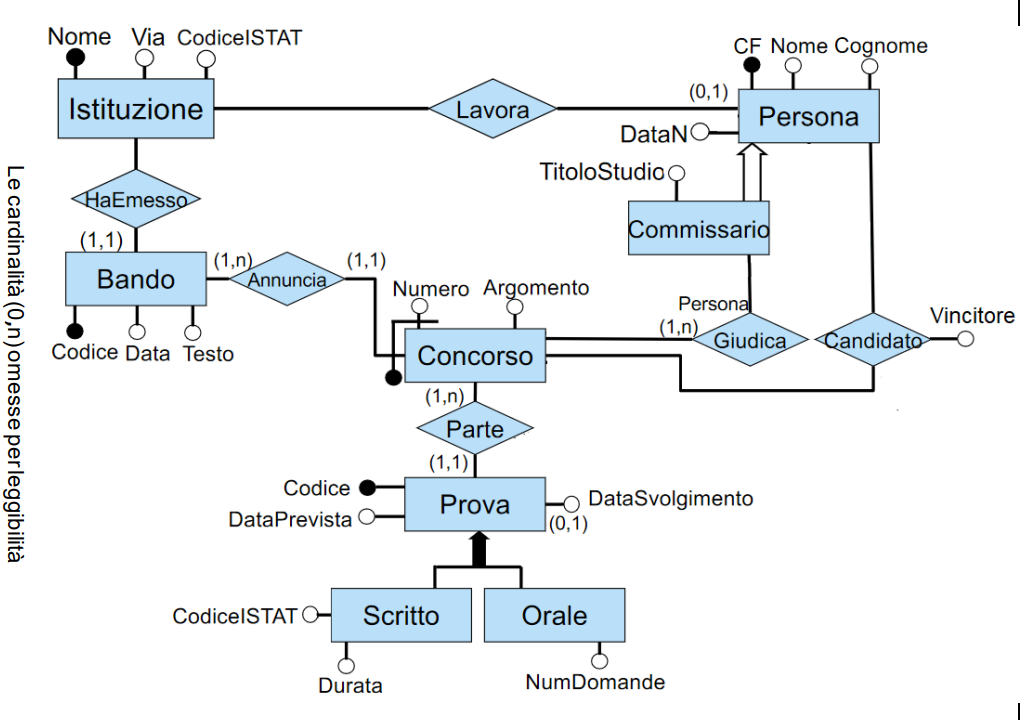
Per ogni istituzione, si vuole sapere le persone che lavorano presto tale istituzione ed i bandi di concorso che ha eventualmente emesso.

Di ogni persona interessa il codice fiscale (identificativo), il nome, il cognome, la data di nascita. Una persona può lavorare presso una istituzione (ma anche essere disoccupata). Alcune persone posso anche fungere da commissari giudicatori di concorso: per loro, si vuole sapere quale sia il titolo di studio.

Di ogni bando interessa il codice identificativo, la data di pubblicazione, il testo ed i concorsi (almeno uno) che sono annunciati nel bando stesso.

Di ogni concorso interessa il bando in cui è annunciato, il numero (unico nell’ambito del bando in cui è annunciato ma diversi bandi possono avere lo stesso numero concorso), l’argomento del concorso (giurisprudenza, ingegneria civile, amministrazione, ecc.) e le prove di cui è composto (almeno una), indentificata da un codice identificativo. Per ogni prova, si vuole conosce la data di svolgimento prevista e successivamente la data di svolgimento effettiva, quando questa avviene.

Le prove sono o scritte o orali e di ogni prova scritta interessa anche la durata e il codice ISTAT del comune dove si prevede si svolga, mentre di ogni prova orale interessa anche il numero di domande previste.

Di ogni concorso interessano anche le persone (almeno una) che hanno partecipato come commissari, le persone che hanno partecipato come candidati (se ci sono) e, tra queste ultime, quelle che sono risultate vincitrici (se ci sono).

*Occorre ottimizzare la seguente operazione: quando si accede alla durata di una prova o al numero di domande, si vuole sempre conoscere il concorso di cui fa parte*.

Quindi si opera eliminando l’entità *Prova* e duplicando i suoi attributi per *Scritto* ed *Orale*.

In merito a Commissario, si mantiene una relazione con cardinalità (0,1) tra Persona e Commissario (perché una persona può essere Commissario) e (1,1) tra Commissario e Persona (perché un commissario è sempre una persona).

Candidato rimane una relazione, essendoci due cardinalità (0,n) e (0,n), aggiungendo solo Vincitore come attributo (può essere nullo in un primo momento come attributo, dato che il testo indica che *può* esserci un vincitore; similmente, essendo che *possono* esserci dei candidati, nella stessa relazione candidato, potrà essere nullo anche l’attributo Persona)

Essendo inoltre che una Persona può essere disoccupata, può essere nullo il collegamento di chiave esterna con Istituzione.

In nome di tutto ciò, si struttura così lo schema logico (indicando con \* i valori nulli)

Istituzione (Nome, CodiceISTAT, Via)

Persona (CF, Nome, Cognome, Data\_nascita, Istituzione\*)

FK:

Persona.Istituzione 🡪 Istituzione.Nome

Commissario (CF, Titolo\_studio, Persona)

FK:

Commissario.CF 🡪 Persona.CF

Giudice (Commissario, Concorso)

FK:

Giudice.Commissario 🡪 Commissario.CF

Giudce.Concorso 🡪 Concorso.Numero

Candidato (Persona, Vincitore\*, Concorso)

FK

Cnndidato.Persona 🡪 Persona.CF

Candidato.Concorso 🡪 Concorso.Numero

Concorso (Numero, Argomento, Bando)

FK

Concorso.Numero 🡪 Bando.Codice

Bando (Codice, Data\_pubb, Testo, Istituzione)

FK

Bando.Istituzione 🡪 Istituzione.Nome

Prova\_scritta (Codice, DataPrevista, DataSvolgimento, CodiceISTAT, Durata, Concorso)

FK

Prova\_scritta.Concorso 🡪 Concorso.Numero

Prova\_orale (Codice, DataPrevista, DataSvolgimento, NumDomande, Concorso)

FK

Prova\_orale.Concorso 🡪 Concorso.Numero

*Si richiede di progettare lo schema concettuale Entità-Relazione di un’applicazione relativa ai festival musicali organizzati da agenzie.*

Di ogni agenzia interessa il nome (identificativo), la provincia della sede (con codice ISTAT identificativo e regione), la persona che la dirige (con anno di inizio direzione) e gli avvisi di festival che ha eventualmente pubblicato.

Festival sono annunciati tramite avvisi. Di ogni avviso interessa l’agenzia che lo ha pubblicato, il codice (unico nell’ambito dell’agenzia che lo ha pubblicato), la data di pubblicazione ed il testo definitivo.

Gli avvisi si suddividono in annullati e validi. Dei primi interessa il motivo dell’annullamento e dei secondi interessa il festival che è annunciato nell’avviso.

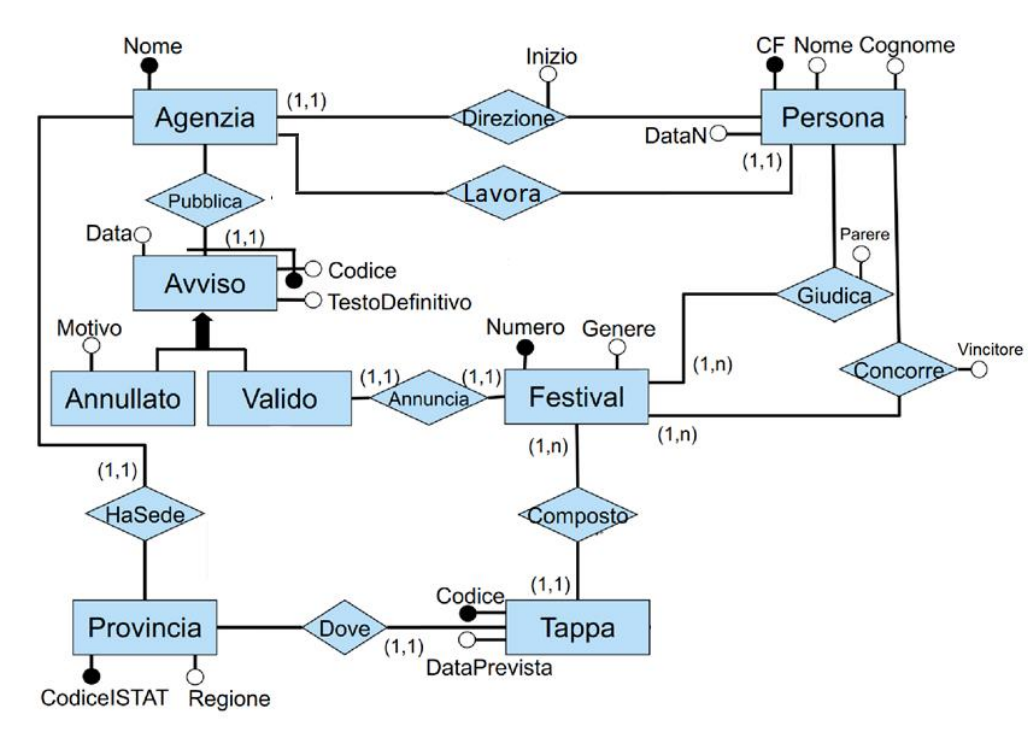
Di ogni festival interessa l’avviso in cui è annunciato, il numero (identificativo), il genere di musica coperto (ad esempio, classica, pop, rock, ecc.) e le tappe di cui è composto (almeno una), ciascuna con codice identificativo(univoco in assoluto), data di svolgimento prevista e la provincia in cui svolgerà.

Di ogni festival interessa sapere le tappe (almeno una), le persone che hanno partecipato come giurati (almeno una), le persone che hanno partecipato come concorrenti (almeno una) e, tra queste ultime, quelle che sono risultate vincitrici (se ci sono).

Di ogni persona interessa il codice fiscale (identificativo), il nome, il cognome e la data di nascita.

*Alcune note*:

* Provincia va messa come entità essendo attributo multivalore e, in particolare, perché indicata dal testo come contenitore di più campi (Codice Istat e Regione)
* La solita entità Persona collega i Giudici (di cui salva un parere anche se non richiesto dal testo), i concorrenti (mettendo come al solito dentro il campo vincitore), e il campo Direzione (Lavora direi che fa parte di una linea poi tagliata dal testo; quindi, salvare se per il giurato interessa anche l’agenzia di cui fa parte)



*A partire dal Diagramma ER dell’Esercizio 1, produrre uno schema relazionale del database nel riquadro sottostante, minimizzando i valori nulli delle relazioni. Indicare i vincoli di chiave e gli attributi che ammettono valori nulli. Illustrare come ristrutturare l’ER per essere traducibile in uno schema relazionale*.

Per minimizzare i valori nulli, pensiamo alle generalizzazioni, in questo caso su Avviso.

A tale scopo, si inserisce due relazioni, avendo cardinalità (0,1) da parte di Avviso e cardinalità (1,1) da parte di Annullato e da parte di Valido, tale che quest’ultimo sia direttamente collegato a Festival.

Togliendo Lavora perché parte di una vecchia richiesta del testo, tutto il resto (Direzione/Giudici/Concorso) vengono pensate come relazioni.

Agenzia (Nome, Dirigente, Provincia)

FK:

Agenza.Dirigente 🡪 Direzione.Persona

Agenzia.Provincia 🡪 Provincia.CodiceISTAT

Direzione (Data\_inizio, Dirigente)

FK:

Direzione.Dirigente 🡪 Persona.CF

Persona (CF, Nome, Cognome, DataN)

Giudica (Giudice, Parere, Festival)

FK:

Giudica.Giudice 🡪 Persona.CF

Giudica.Festival 🡪 Festival.Numero

Concorre (Vincitore, Festival)

FK:

Concorre.Vincitore 🡪 Persona.CF

Concorre.Festival 🡪 Festival.Numero

Festival (Numero, Genere, Avviso\_valido)

FK:

Festival.Avviso\_valido 🡪 Avviso\_valido.Avviso

Avviso (Codice, TestoDefinitivo, Data, Agenzia)

FK:

Avviso.Agenzia 🡪 Agenzia.Nome

Avviso\_annullato (Avviso, Motivo)

FK:

Avviso\_annullato.Avviso 🡪 Avviso.Codice

Avviso\_valido (Avviso, Festival)

FK:

Avviso\_valido.Avviso 🡪 Avviso.Codice

Avviso\_valido.Festival 🡪 Festival.Numero

Provincia (CodiceISTAT, Regione)

Tappa (Codice, Data\_prevista, Festival)

FK:

Tappa.Festival 🡪 Festival.Numero

*Si richiede di produrre lo schema concettuale Entità-Relazione di un’applicazione relativa alla gestione delle edicole per la vendita di giornali, Disegnare il diagramma ER nel riquadro della pagina che segue.*

Si richiede di progettare lo schema ER concettuale di un’applicazione relativa alle edicole per la vendita di giornali.

Di ogni edicola interessa il comune in cui essa è registrata, il codice, che è unico nell’ambito del comune in cui l’edicola stessa è registrata, la categoria, l’anno di inizio attività (non sempre disponibile), e i contratti che l’edicola ha con i distributori per l’approvvigionamento dei quotidiani.

Ogni contratto riguarda un’edicola, un quotidiano ed un distributore, ed è caratterizzato dal costo mensile a carico dell’edicola. Infine, per ogni edicola, interessa conoscere le varie persone che sono state proprietarie dell’edicola nei diversi anni, tenendo conto del fatto che in ogni anno un’edicola ha al massimo un proprietario.

Di ogni persona interessa il codice fiscale (id), l’anno di nascita, il comune di nascita, ed il comune di residenza. Di ogni distributore di quotidiani interessa la partita IVA (id), il fatturato ed il comune in cui è situata la direzione.

Di ogni quotidiano interessa il nome (identificativo), e l’anno di inizio pubblicazione. Di ogni comune interessa la provincia di appartenenza, il nome (unico nella provincia), ed il numero di abitanti. Dei comuni che sono capoluogo di provincia interessa l’attuale sindaco (con l’anno di elezione), ed il numero di assessori comunali.

Di ogni provincia interessa il nome (identificativo) e la regione di appartenenza. Alcune province sono “autonome” e di esse interessa anche l’anno di istituzione

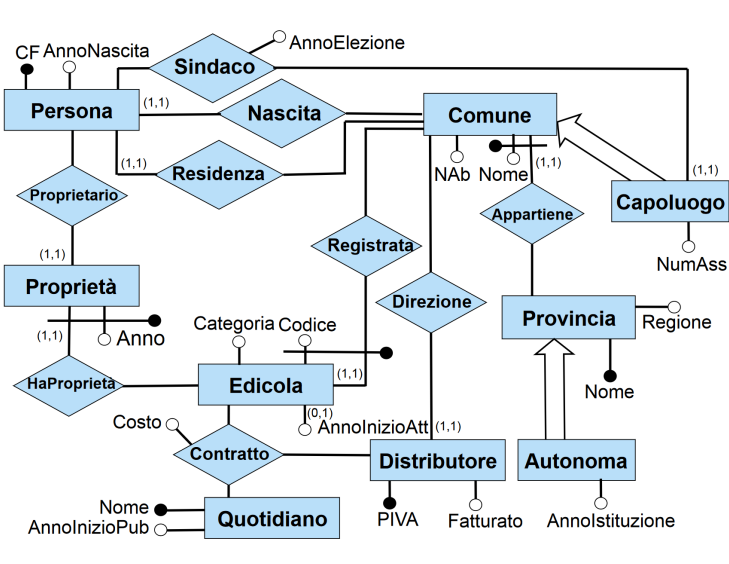


Immagine che contiene testo, quotidiano, screenshot

Descrizione generata automaticamente

Il sistema deve memorizzare le informazioni di ogni pista della catena. Di ogni pista, è di interesse sapere (1) la città in cui si trova, (2) l’anno di apertura e (3) il fatturato. Ogni città non può avere più di una pista. Inoltre, ogni pista è gestita da una persona, però una pista potrebbe temporaneamente non avere una persona che la gestisce. Si noti che una persona non può prendere in gestione più di una pista della catena.

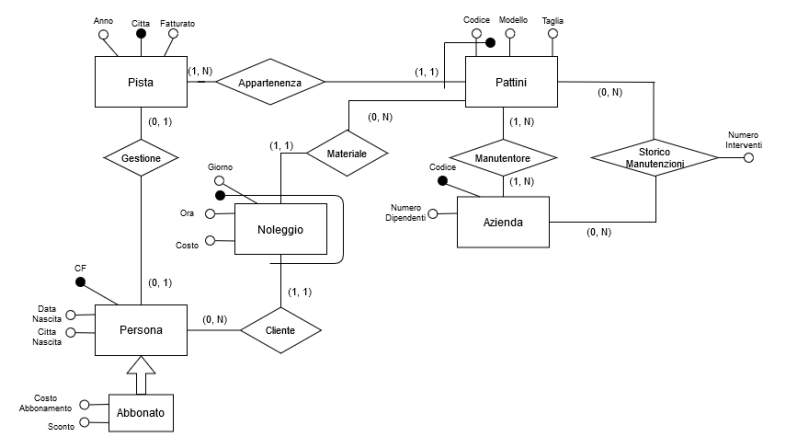
Il sistema memorizza un’anagrafe delle persone: di ogni persona, è di interesse sapere il codice fiscale (identificativo), la data di nascita e la città di nascita.

Quando le persone desiderano essere clienti di una pista, noleggiano un paio di pattini. Ogni paio di pattini è associato ad esattamente una pista.

Di ogni paio di pattini interessa il codice (unico all’interno della pista), il modello, la taglia. Inoltre, per ogni paio di pattini, è di interesse sapere quali aziende sono in grado di farne manutenzione. Si vuole anche memorizzare lo storico delle manutenzioni: per ogni paio di pattini, è di interesse sapere quante volte (intero) il paio è andato in manutenzione in ogni azienda in grado di manutenere. Di ogni azienda di riparazione pattini interessa il codice identificativo e il numero di dipendenti.

Come detto, ogni paio di pattini è noleggiato da persone per usarle nella pista associata. In particolare, si vuole sapere per ogni noleggio: (1) quale paio è stato preso, (2) il giorno e l’ora del noleggio e (3) il costo complessivo del noleggio. Si noti che ogni paio di pattini può essere noleggiato più volte dalla stessa persona, ma solo in giorni diversi.

Alcune persone sono clienti abbonati. Per tali clienti, è di interesse sapere il costo dell’abbonamento annuale (unico per cliente) e quale sconto (in percentuale) ottengono nei noleggi in virtù del loro abbonamento.



Si noti che:

* si parla di aziende in grado di fare manutenzione; ciò presuppone che ci sia almeno una azienda che fa manutenzione e, nel corso del tempo, avrò diverse aziende che fanno manutenzione; quindi è necessaria la doppia relazione
* la possibilità di scrivere Cliente e Gestore come generalizzazione (totale/parziale) di Persona ha poco senso; si noti che, a prescindere, volendo minimizzare i valori nulli si dovrebbero creare delle relazioni e i clienti non hanno alcun dato che non sia già presente in Persona.

Abbonato avrebbe dei dati, ma appunto la ristrutturazione considera l’introduzione di una relazione con cardinalità (0,1) ed (1,1) tra Persona ed Abbonato.

Immagine che contiene testo

Descrizione generata automaticamenteInfatti, il prof dice questo:

Considerato inoltre che una persona gestisce una sola pista ed una pista ha un solo gestore, allora questa sarà normalmente collegata come chiave esterna sotto forma di relazione.

Lo schema logico è:

PERSONA(CF, DataNascita, CittaNascita)

PISTA(Citta, Fatturato, Anno, Gestore)

• Chiave Esterna: PISTA.Gestore 🡪 Persona.CF

• Chiave Aggiuntiva: Gestore

PATTINI(Codice, CittaPista, Modello, Taglia)

• Chiave Esterna: PATTINI.CittaPista 🡪 PISTA.Citta

NOLEGGIO(CodicePattini, CittaPista, Persona, Giorno, Ora, Costo)

• Chiave Esterna: NOLEGGIO.(CodicePattini, CittaPista) 🡪 PATTINI.(Codice,CittaPista)

• Chiave Esterna: NOLEGGIO.Persona 🡪 Persona.CF A

AZIENDA(Codice, NumeroDipendenti)

MANUTENTORE(CodicePattini, CittaPista, CodiceAzienda)

• Chiave Esterna: MANUTENTORE.(CodicePattini, CittaPista) 🡪 PATTINI.(Codice,CittaPista)

• Chiave Esterna: MANUTENTORE.CodiceAzienda 🡪 AZIENDA.Codice STORICO(CodicePattini, CittaPista, CodiceAzienda, NumeroInterventi)

• Chiave Esterna: STORICO.(CodicePattini, CittaPista) 🡪 PATTINI.(Codice,CittaPista)

• Chiave Esterna: STORICO.CodiceAzienda 🡪 AZIENDA.Codice

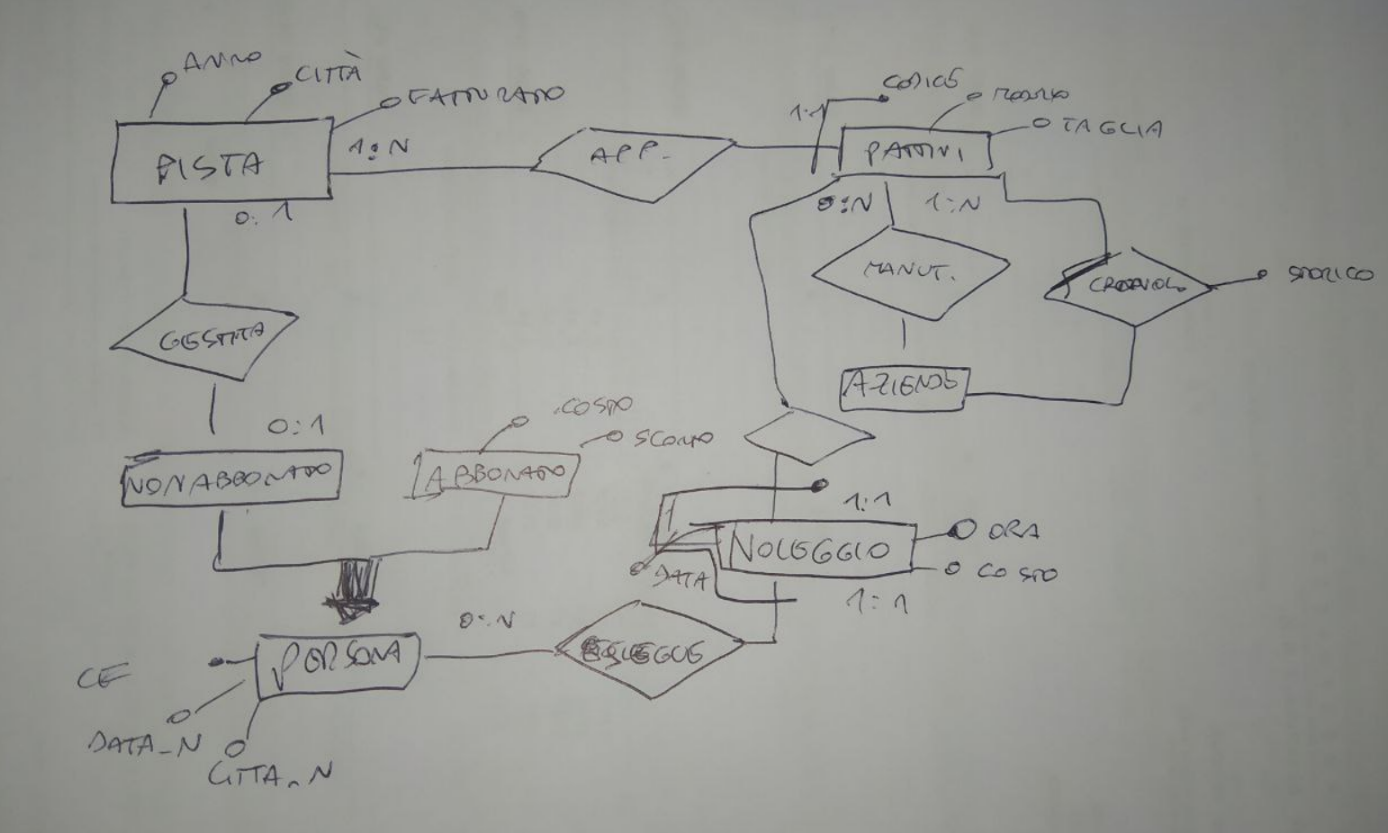
ABBONATO(CF, Costo, Sconto)

• Chiave Esterna: ABBONATO.CF 🡪 PERSONA.CF

Un’altra possibile soluzione discute questo fatto:

* con questa soluzione si ammette che un abbonato possa gestire una pista; il testo non cita esplicitamente questo fatto. Per questo, la persona ammette che sia abbonato che non abbonato possano gestire. Un cliente concettualmente non aggiunge alcuna informazione utile che non sia già in persona; rimane relazione

Più corretto quindi eseguire una generalizzazione totale tra Persone e Abbonato e Non Abbonato, ammettendo che un Non Abbonato gestisca e che il Gestore sia una persona non abbonata e che ha facoltà di gestire.



Il negozio di fumetti vuole costruire un database relativo ai fumetti in vendita nel negozio, tenendo traccia dei fumetti venduti e dei clienti che hanno effettuato acquisti.

Ogni fumetto è identificato dal nome della serie di cui fa parte e dal numero che lo contraddistingue all’interno della serie (ad es. Topolino n.21243). Per tutti i fumetti si vuole inoltre conoscere l’anno di pubblicazione.

Per alcune tipologie di fumetti sono di particolare interesse informazioni addizionali. Per i manga (i fumetti giapponesi) interessa conoscere l’autore, mentre per i fumetti western è interesse conoscere la casa editrice.

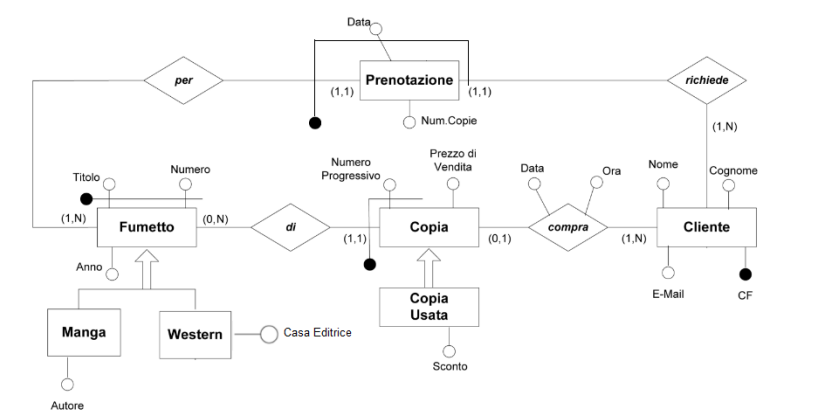
Il negozio vuole tener traccia delle varie copie disponibili per ciascun fumetto; ciascuna copia è identificata un numero progressivo e da un prezzo diverse. Copie diverse dello stesso fumetto possono avere prezzi di vendita diverse (per esempio, in funzione del momento di stampa della copia).

Ciascun cliente (di cui interessa conoscere il codice fiscale, il nome, il cognome e l’e-mail) può fare le seguenti cose:

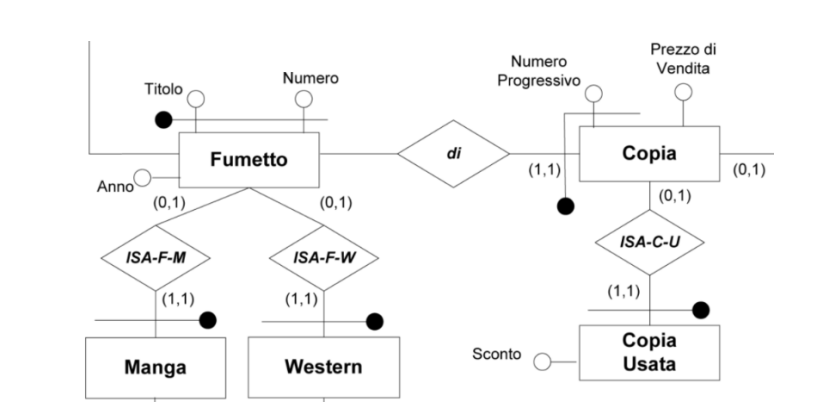
• Acquistare copie di fumetti: in tal caso dell'acquisto si memorizza la data e l'ora dello stesso

• Prenotare fumetti di interesse: un cliente può prenotare un certo numero di copie di un particolare fumetto, memorizzando data della prenotazione. Lo stesso cliente può prenotare lo stesso fumetto più volte, ma solo in date diverse.

Si noti che il negozio può vendere anche copie usate di un fumetto, e in questo caso è di interesse conoscere lo sconto applicato rispetto alla versione venduta a prezzo pieno.



La ristrutturazione dello schema logico le vede come al solito come delle “is-a” e come tale aggiunge relazioni di collegamento intermedie:



E lo schema logico completo è il seguente:

FUMETTO (Titolo, Numero, Anno)

MANGA (Titolo, Numero, Autore)

• Chiave Esterna: MANGA.(Titolo,Numero) 🡪 FUMETTO.(Titolo,Numero)

WESTERN (Titolo, Numero, CaseEditrice)

• Chiave Esterna: WESTERN.(Titolo,Numero) 🡪 FUMETTO.(Titolo,Numero)

CLIENTE(CF, Nome, Cognome, E-mail)

COPIA(FumettoTitolo, FumettoNumero, NumeroProgressivo, PrezzoVendita)

• Chiave Esterna: COPIA.(FumettoTitolo, FumettoNumero) 🡪 FUMETTO.(Titolo,Numero)

COPIA-USATA(FumettoTitolo, FumettoNumero, NumeroProgressivo, Sconto)

• Chiave Esterna: COPIA-USATA.(FumettoTitolo, FumettoNumero,NumeroProgressivo) 🡪 COPIA.(FumettoTitolo, FumettoNumero,NumeroProgressivo)

COMPRA(FumettoTitolo, FumettoNumero, NumeroProgressivo, CF-Cliente, Data, Ora)

• Chiave Esterna: COMPRA.(FumettoTitolo, FumettoNumero,NumeroProgressivo) 🡪 COPIA.(FumettoTitolo, FumettoNumero,NumeroProgressivo)

• Chiave Esterna: COPIA.CF\_Cliente 🡪 CLIENTE.CF

PRENOTAZIONE(FumettoTitolo, FumettoNumero, CF-Cliente, NumCopie)

• Chiave Esterna: PRENOTAZIONE.(FumettoTitolo, FumettoNumero) 🡪 FUMETTO.(Titolo,Numero)

• Chiave Esterna: PRENOTAZIONE.CF-Cliente 🡪 CLIENTE.CF

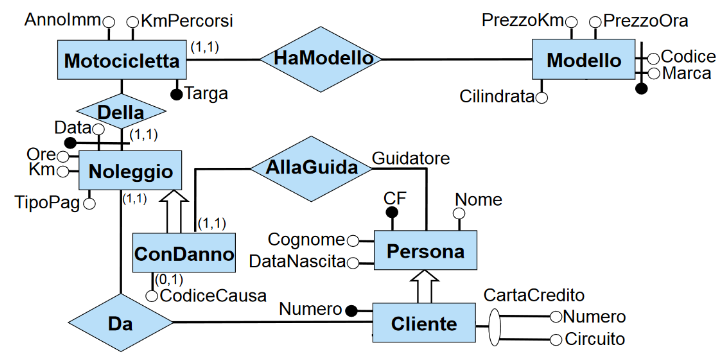
Si noti che, se COMPRA fosse fusa con COPIA, il numero di attributi con valori potenzialmente nulli sarebbero alti. Infatti, COPIA avrebbe tre attributi CF-Cliente, Data e Ora che avrebbero valori nulli per tutte le copie non ancora vendute.

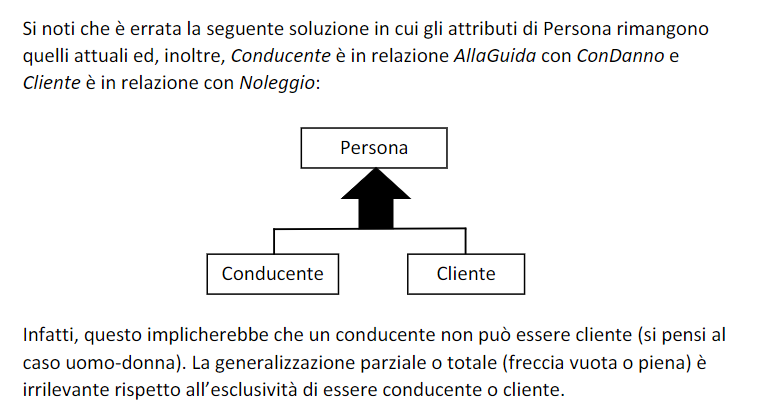
Il negozio possiede tante motociclette. Di ogni motocicletta, interessa la targa (identificativo), il modello, l’anno di immatricolazione ed i chilometri percorsi. Di ogni modello interessala marca (ad esempio, “Honda”), il codice (unico nell’ambito della marca), la cilindrata, il prezzo del noleggio all’ora ed il prezzo del noleggio al chilometro.

Di ogni noleggio, interessa il cliente che l’ha effettuato, la motocicletta noleggiata, la data in cui è avvenuto (ogni noleggio è infatti relativo ad un giorno), la durata in ore, la tipologia di pagamento (a chilometro oppure ad ora) ed i chilometri percorsi. Si noti che una stessa motocicletta non può essere noleggiata più volte nello stesso giorno.

Dei clienti, Il negozio vuole conoscere il numero di telefono. Ogni cliente possiede una carta di credito, di cui interessa il numero ed il circuito.

Per quei noleggi per i quali si è verificato un danno alla motocicletta noleggiata, interessa il codice indicante la causa del danno (se nota) e la persona-conducente che era alla guida della motocicletta al momento del verificarsi del danno. Di ogni conducente, l’applicazione deve memorizzare i seguenti dati: codice fiscale (identificativo), nome, cognome e data di nascita. Si noti che il conducente può coincidere con il cliente del noleggio oppure essere un’altra persona.





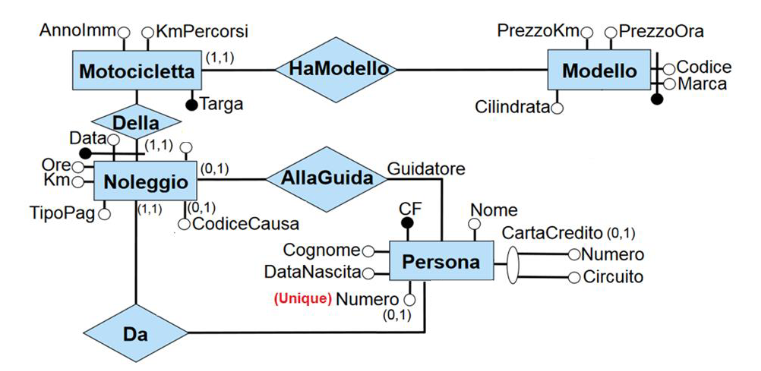
Ristrutturazione:

Dato che il testo indica questo:

*Nella progettazione logica, si vuole assicurare il minimo numero minimo di tabelle necessarie, anche a costo di valori nulli. Questo vincolo è al fine di minimizzare il numero di join.*

Siamo in presenza di una ristrutturazione semplice:

* Tutti gli attributi di Con Danno entrano a far parte di Noleggio
* Tutti gli attributi di Cliente entrano a far parte di persona

SI porta quindi tutto su:

Il vincolo *Unique* per numero serve a rappresentare che, se una persona ha un numero telefonico, allora è l’unica con quel numero. Questo viene derivato dal fatto che l’entità Cliente nel diagramma ER concettuale usa tale attributo come identificatore. Si noti che una persona potrebbe non avere un numero di telefono perché solo le persone clienti lo hanno

*Da Esercitazione sull’ER:*

La serra *PianteBellissime* vuole dotarsi di un sistema informativo per gestire la manutenzione e la vendita delle piante. La serra è specializzata in piante di alto fusto, ma vende anche piante fiorite.

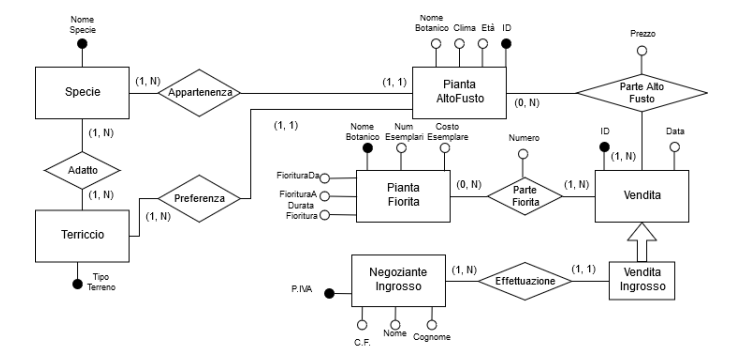
Per le piante ad alto fusto, si vogliono memorizzare i singoli esemplari in magazzino. Ogni esemplare è contraddistinto da un codice individuale, la specie, il nome botanico, l’età in anni, il clima ed il tipo di terriccio ideali.

Per le piante fiorite, non è di intesse memorizzare i singoli esemplari, ma solo le quantità a disposizione in forma aggregata. Di conseguenza, I tipi di piante fiorite non hanno un codice individuale, sono raggruppate in insiemi identificati tramite il nome botanico; di ogni insieme sono noti il periodo e la durata della fioritura, il numero di esemplari in magazzino, nonché il costo del singolo esemplare (uguale per tutti gli esemplari).

Le piante di alto fusto, se non vendute, devono essere sottoposte ad un trattamento annuale di svasamento con terricci diversi a seconda del tipo. A tale scopo, occorre associare ogni specie di pianta ad uno o più terricci.

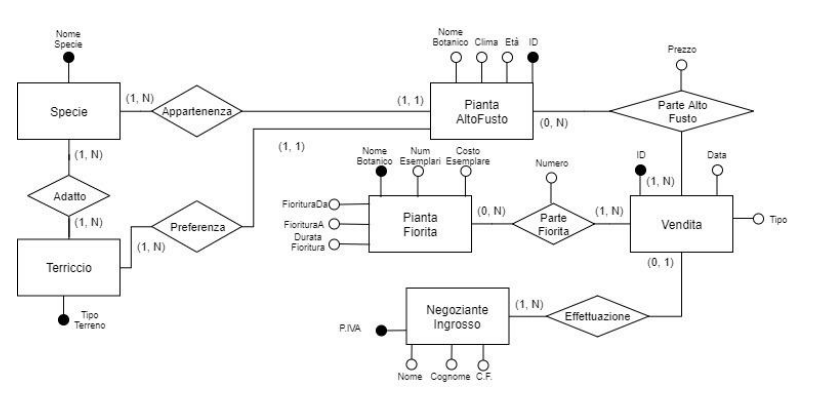
La serra ha una clientela di due tipi: singoli individui che acquistano al dettaglio e negozianti che acquistano all’ingrosso. Soltanto per questi ultimi è necessario conoscere nome, cognome, codice fiscale e partita iva.

Di ogni vendita interessa memorizzare la data, l’identificativo, le piante acquistate. Ogni vendita contiene uno o più tipi di piante fiorite e piante ad alto fusto. Per ogni vendita, ogni tipo di pianta fiorita è venduto in una certa quantità; viceversa, ogni vendita contiene solo un esemplare di un tipo di pianta ad alto fusto, di cui si vuole conoscere il prezzo di vendita, che può cambiare ad ogni vendita. Solo per le vendite all’ingrosso, si interessa conoscere l’acquirente.

In merito all’ER alcune osservazioni:

* Le specie, citate da parte del testo, richiedono solo un piccolo ragionamento; ogni specie contiene più esemplari e questi ultimi fanno parte di una sola specie. A questi scopi viene istituita l’entità Specie
* L’informazione forma aggregata non specifica altre informazioni utili, semplicemente le piante fiorite e ad alto fusto sono due cose diverse già descritte dalle entità
* Il trattamento a cui devono essere sottoposte le piante dipende dal tipo di terriccio; pertanto, quest’ultimo attributo è la chiave di Terriccio, di cui fanno parte N specie delle piante ad alto fusto
* Il fatto che ci sia una generalizzazione per Vendita all’ingrosso viene suggerito dal testo perché dice che “*Solo per le vendite all’ingrosso, interessa conoscere l’acquirente*”
* Si citano due tipi di clientela, al che uno penserebbe alla generalizzazione totale da Acquirente al dettaglio e Commerciante all’Ingrosso verso Cliente. Il testo però cita espressamente che solo per i commercianti all’ingrosso si intendono salvare dati e, nel successivo pezzo di testo, specifica pure che solo per la vendita all’ingrosso vogliamo salvare dati.

Per tutti questi motivi, Vendita si prefigura come entità, Vendita Ingrosso rappresenta una generalizzazione parziale, perché *eventualmente* salviamo i dati utili

*Ristrutturazione*

Dato che vogliamo minimizzare i valori nulli, idealmente a noi interessa solo capire se la vendita sia all’ingrosso o meno. La generalizzazione può quindi essere risolta togliendo Vendita Ingrosso ed aggiungendo un campo Tipo, che qualifica in che modo la vendita avvenga.

Mantenendo solamente una relazione senza aggiungere il campo Tipo, si risolverebbe la generalizzazione, ma si salverebbero anche i dati di tutte quelle vendite che non sono all’ingrosso.

Dato che i sistemi tradizionali per la gestione delle basi di dati non consentono di rappresentare costrutti quali generalizzazioni, gerarchie, attributi composti o identificatori esterni, risulta necessario ristrutturare lo schema precedente trasformandoli in elementi rappresentabili. In questo caso, nella trasformazione, bisogna solamente rimuovere la generalizzazione Vendita-Vendita Ingrosso.

Per far ciò aggiungiamo un attributo “Tipo” alla classe Vendita e aggiungiamo il seguente vincolo di integrità:

• L’attributo Tipo della classe Vendita può assumere solamente i valori {All’ingrosso, Non all’ingrosso}.

*Schema logico*:

Terriccio (TipoTerreno)

Specie (NomeSpecie)

Adatto (Specie, Terriccio)

• Adatto.Terriccio -> Terriccio.TipoTerreno

• Adatto.Specie -> Specie.NomeSpecie

PiantaAltoFusto (ID, NomeBotanico, Età, Clima, TipoTerreno, NomeSpecie)

• PiantaAltoFusto.NomeSpecie -> Specie.NomeSpecie

ParteAltoFusto (Vendita, PiantaAltoFusto, Prezzo)

• ParteAltoFusto.Vendita -> Vendita.ID

• ParteAltoFusto.PiantaAltoFusto -> PiantaAltoFusto.ID

Vendita (ID, Data, Tipo, PIVANegoziante) (PIVANegoziante è chiave esterna a Negoziante all’ingrosso partita IVA)

• PIVA può essere NULL

ParteFiorita (Vendita, PiantaFiorita, Numero)

• ParteFiorita.Vendita -> Vendita.ID

• ParteFiorita. PiantaFiorita -> PiantaFiorita.NomeBotanico

PiantaFiorita (NomeBotanico, NumEsemplari, CostoEsemplare, FornituraDa, FornituraA, DurataFioritura)

Negoziante Ingrosso (PIVA, CF, Nome, Cognome)

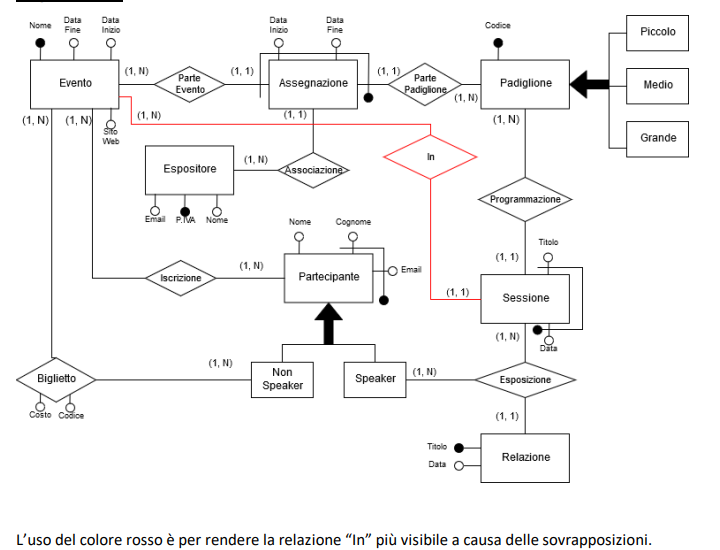
Un’azienda che gestisce gli eventi di uno spazio di fiera vuole progettare una base di dati per la memorizzazione delle informazioni di suo interesse.

L’azienda ha il compito di gestire tutti gli eventi che sono organizzati nello spazio di fiera. Questi eventi sono caratterizzati da un nome, dalla durata dell’evento e, se disponibile, dal sito web dell’evento.

L’evento si svolge nei padiglioni dello spazio di fiera. È di interesse memorizzare quali padiglioni sono occupati in quali giorni. Inoltre, i padiglioni, oltre ad essere identificati mediante un codice alfanumerico, sono classificabili in padiglioni grandi, medi e piccoli. Ogni padiglione ha associato un espositore, di cui interessa il nome, la partita iva, ed il riferimento ad una persona di contatto (e-mail).

Il padiglione ha un programma delle giornate in cui si svolge l’evento che risulta organizzato in sessioni. A ciascuna sessione è associato un titolo, la giornata in cui si svolge e un insieme di speaker, di cui interessa nome, cognome, e-mail, oltre che titolo e durata della relazione che dovranno esporre.

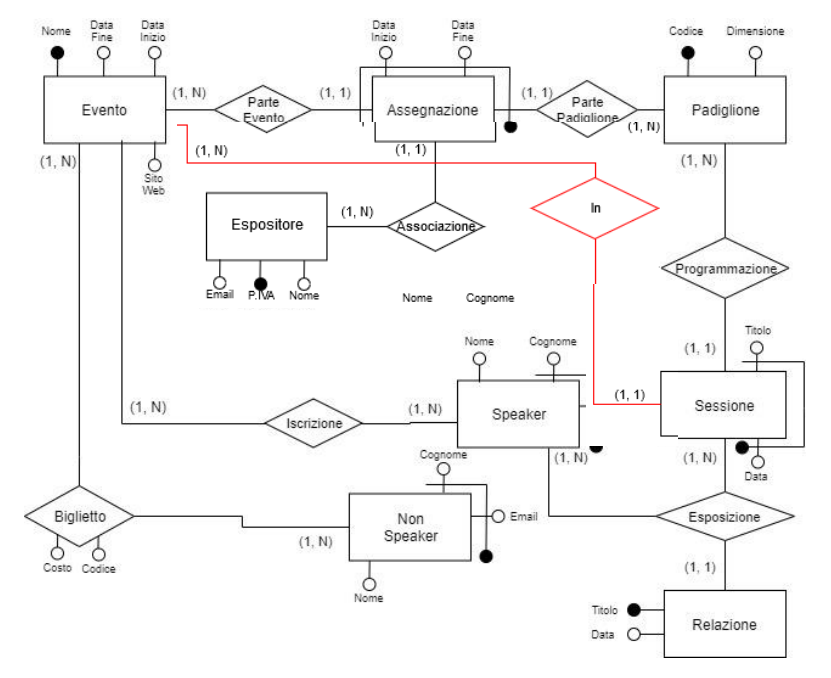
Si noti che uno speaker può effettuare più di una relazione all’interno dello stesso evento. I partecipanti all’evento devono iscriversi fornendo i loro dati anagrafici (nome, cognome, e-mail). Anche gli speaker devono iscriversi all’evento. Tutti i partecipanti tranne gli speaker devono poi pagare un biglietto di iscrizione caratterizzato da un codice identificativo ed un costo.



Il diagramma ER possiede 2 generalizzazioni:

• Rimuoviamo le classi Piccolo, Medio, Grande e aggiungiamo un nuovo attributo Dimensione alla classe Padiglione. Aggiungiamo quindi il vincolo che tale attributo può assumere solamente i valori {Piccolo, Medio, Grande}.

• Le classi Speaker e Non-Speaker ereditano gli attributi e le relazioni della classe Partecipante. Tuttavia, Non-Speaker era già in relazione con Evento, quindi nessuna relazione viene aggiunta.

*Schema logico:*

Evento (Nome, DataFine, DataInizio, SitoWeb)

Assegnazione (DataInizio, DataFine, Padiglione, Nome-Evento, PIVA\_Espositore,)

• Assegnazione.Padiglione -> Padiglione.Codice

• Assegnazione.PIVA\_Espositore -> Espositore.PIVA

• Assegnazione.Nome-Evento -> Evento.Nome

Espositore (PIVA, Nome, E-mail)

Padiglione (Codice, Dimensione)

Sessione (Titolo, Data, Padiglione, Evento)

• Sessione.Padiglione -> Padiglione.Codice

• Sessione.Evento -> Evento.Nome

Speaker (Cognome, E-mail, Nome)

Iscrizione (Evento, SpeakerCognome, SpeakerEmail)

• Iscrizione.Evento -> Evento.Nome

• Iscrizione.SpeakerCognome -> Speaker.Cognome

• Iscrizione.SpeakerEmail -> Speaker.E-mail

Biglietto (Evento, Non-SpeakerCognome, Non-SpeakerEmail, Costo, Codice)

• Biglietto.Evento -> Evento.Nome

• Biglietto.(Non-SpeakerCognome, Non-SpeakerEmail) -> Non-Speaker.(Cognome,Email)

Non-Speaker (Cognome, E-mail, Nome)

Relazione (Titolo, Data, Speaker, Sessione)

Esposizione (SpeakerCognome, SpeakerEmail, SessioneTitolo, SessioneData, Relazione)

• Esposizione.SpeakerCognome -> Speaker.Cognome

• Esposizione.SpeakerEmail -> Speaker.E-mail

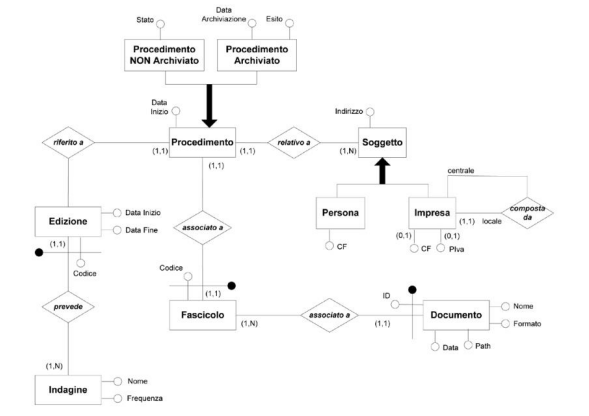
• Esposizione.SessioneTitolo -> Sessione.Titolo

• Esposizione.SessioneData -> Sessione.Data

• Esposizione.Relazione -> Relazione.Titolo

Si vuole realizzare una base di dati che gestisca procedimenti sanzionatori nel contesto di rilevazioni statistiche ufficiali di carattere nazionale. Per alcune rilevazioni statistiche ufficiali esiste, infatti, l’obbligo di risposta da parte dei soggetti contattati per la conduzione delle rilevazioni. Qualora il soggetto contattato non risponda al questionario inviatogli, dopo un prefissato intervallo di tempo, ha inizio un procedimento sanzionatorio che consta di due fasi principali: invio della diffida al soggetto non rispondente e, qualora tale soggetto continui ad essere inadempiente (cioè non risponda al questionario), invio della sanzione che il soggetto stesso dovrà pagare. I soggetti possono essere persone fisiche o imprese. Delle persone fisiche interessa memorizzare il codice fiscale, delle imprese il codice fiscale o la partita iva in maniera alternativa. Inoltre, è di interesse l’indirizzo cui il soggetto è contattabile. Si noti che le imprese possono prevedere delle unità locali, ovvero l’impresa si articola secondo una struttura che consiste di un’impresa centrale ed eventualmente di un insieme di imprese “periferiche”. Un procedimento viene avviato in relazione alla non risposta ad una specifica edizione di un’indagine. Ogni indagine è caratterizzata da un nome (es. Forze di lavoro), da una frequenza con cui le sue edizioni occorrono (es. trimestrale) e dalle specifiche edizioni che sono occorse (es. primo trimestre 2011).

Le edizioni, che hanno un codice univoco nell’ambito dell’indagine in cui sono svolte, hanno una data di inizio ed una data di fine che caratterizzano l’inizio e la fine della rilevazione sul campo dei dati oggetto dell’indagine. Nell’ambito di un procedimento è prodotto un insieme di documenti che costituisce il fascicolo del procedimento. Un fascicolo ha un codice che lo identifica nell’ambito del procedimento a cui è legato. I documenti, che dispongono di un ID univoco nell’ambito del fascicolo in cui sono redatti, sono rappresentati da un nome, un tipo, una data di produzione e dal path relativo al file cui sono associati. Del procedimento, oltre alle informazioni necessarie a desumere il suo avanzamento, interessa memorizzare la data di inizio e, qualora sia stato archiviato, l’esito della archiviazione (ad esempio archiviato perché il soggetto ha risposto) e la data di archiviazione.



Interessante da esaminare il caso dell’impresa locale e le relative periferiche; dato che non vogliamo memorizzare alcuna informazione in merito a queste singole, si noti che il testo specifica che l’impresa di per sé è locale, dunque costituita da un impresa centrale ed un insieme di periferiche.

La prima idea sarebbe di usare una generalizzazione per rappresentare questo discorso ma non si manterrebbe in alcun modo il riferimento che si ha tra l’impresa locale, di per sé impresa e periferiche; chiaro quindi come si nota una ricorsione, le imprese periferiche dipendono dall’impresa locale e similmente l’impresa locale *potrebbe* avere delle periferiche.

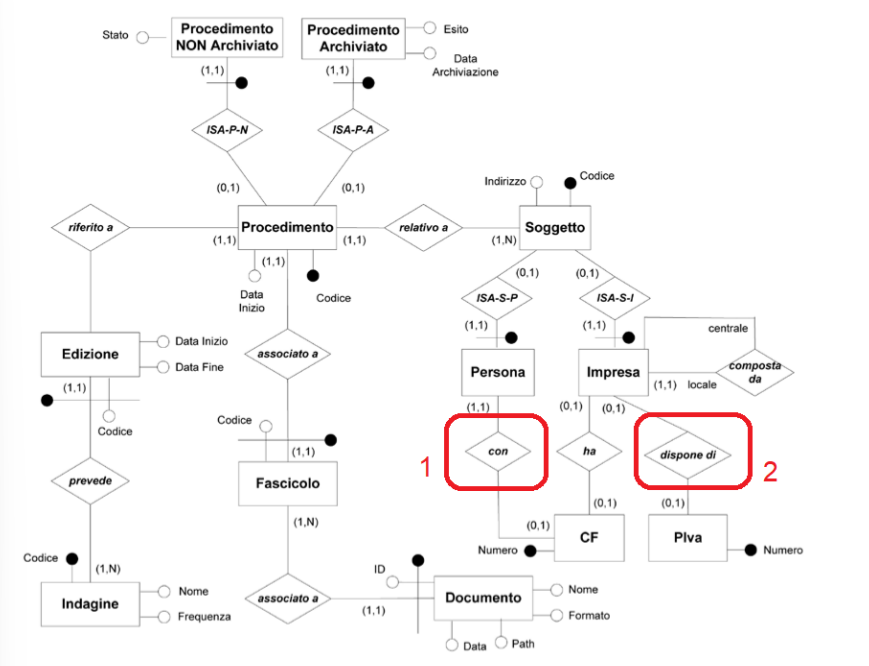
Sono comunque dati irrilevanti al mantenimento dell’informazione del procedimento avviato su di esse; dunque, si struttura la soluzione presentata sopra. Per identificare correttamente l’impresa centrale, viene scelto di aggiungere un campo codice centrale successivamente nello schema logico, distinguendo così tramite codice normale le imprese definite come periferiche.

*E ristrutturazione:*

Essendo che dobbiamo minimizzare la presenza di valori nulli, allora si sceglie di introdurre delle relazioni tra persona e CF e Impresa con PIVA/CF, mantenendo cardinalità (0,1) da parte di CF/Piva ed (1,1) da parte di Persona ed Impresa.

Similmente per Procedimento, cioè il discorso Archiviato/Non archiviato, mettendo 0,1 da parte di procedimento verso le due sottoentità e (1,1) da parte delle due sottoentità.

Le scelte effettuate nella fase di ristrutturazione hanno avuto come obiettivo quello di evitare la presenza di valori NULL nella base di dati. Questo ha comportato una traduzione "ridondante” delle generalizzazioni, in cui sia l’entità padre che quelle figlie sono state mantenute nello schema, e una traduzione degli attributi con cardinalità (0,1) in nuove entità specifiche. Si noti che quest’ultimo aspetto ha comportato la modifica di uno dei vincoli esterni (quello identificato nel rettangolo num. 2) e l'aggiunta di un nuovo vincolo esterno (quello identificato nel rettangolo num. 1), rispetto a quelli definiti in fase di Progettazione Concettuale. Inoltre, dato che, per ogni entità, è necessario individuare un campo principale, è stato introdotto un codice, i cui valori sono speciali ed hanno l'unico scopo di identificare le istanze dell'entità, per tutte quelle entità che naturalmente non ne dispongono.



*Schema Logico:*

Procedimento(Codice, Soggetto, Edizione, Indagine, DataInizio)

• Procedimento.Soggetto --> Soggetto.Codice)

• Procedimento.(Edizione,Indagine) --> Edizione.(Codice, Indagine) ProcedimentoNonArchiviato(Codice,Stato)

• ProcedimentoNonArchiviato.Codice -->Procedimento.Codice) ProcedimentoArchiviato(Codice,Esito,DataArchiviazione)

• ProcedimentoArchiviato.Codice -->Procedimento.Codice

Indagine(Codice,Nome,Frequenza)

Edizione(Codice,Indagine,DataInizio,DataFine)

• Edizione.Indagine --> Indagine.Codice

Fascicolo(Codice,Procedimento)

• Fascicolo.Procedimento --> Procedimento.Codice Documento(ID,Fascicolo,Data,Path,Nome,Formato)

• Documento.Fascicolo -->Fascicolo.Codice

Soggetto(Codice,Indirizzo)

Persona(Codice,CF)

• Persona.Codice -->Soggetto(Codice)

• Persona.CF -->CF.Numero

Impresa (Codice,CodiceCentrale)

• Impresa.Codice -->Soggetto.Codice

• Impresa.CodiceCentrale --> Impresa.Codice

CF(Numero)

PIva(Numero)

ha(CF,Impresa)

• ha.CF --> CF.Numero

• ha.Impresa --> Impresa.Codice

ImpresadisponeDi(PIva, Impresa)

• disponeDi.PIva --> PIva(Numero)

• disponeDi.Impresa --> Impresa.Codice