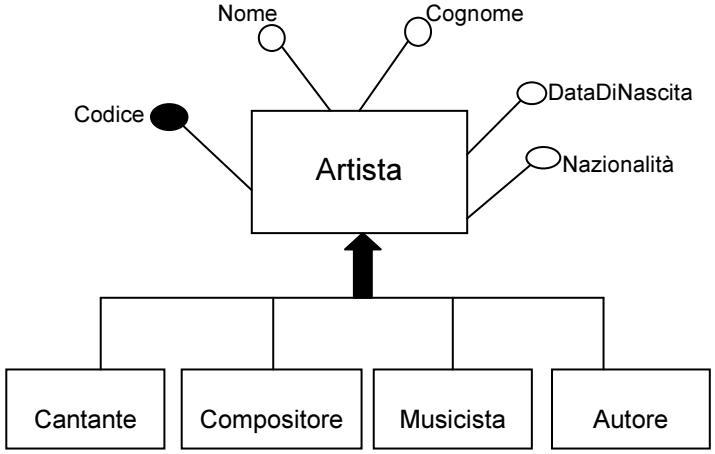
PROGETTAZIONE LOGICA

L'obiettivo della fase di progettazione Logica è progettare lo Schema Logico della Base di Dati partendo da quanto prodotto nella fase di progettazione Concettuale.
Si ricorda che, se come modello Concettuale dei Dati si è utilizzato il modello Entità-Relazione (E-R), i prodotti della fase di progettazione Concettuale sono lo Schema o Diagramma E-R della Base di Dati e le Regole Aziendali. Se adottiamo come modello Logico dei Dati il modello Logico Relazionale il risultato della fase di progettazione Logica sarà lo Schema Logico Relazionale della Base di Dati. Di conseguenza, la fase di progettazione Logica dovrà "tradurre" opportunamente il Diagramma E-R e le Regole Aziendali prodotte durante la fase di progettazione Concettuale nello Schema Logico Relazionale della Base di Dati.
Un algoritmo semplificato per effettuare tale "traduzione", applicabile ai costrutti del modello E-R visti a lezione, si compone di 4 passi:

1. **Rappresentazione delle Entità**. Ad ogni Entità presente nel Diagramma E-R corrisponderà una Relazione (d'ora in avanti Tabella per evitare confusione con il termine *relazione* del modello E-R) nello Schema Logico Relazionale. Ogni attributo dell'Entità sarà un attributo della Tabella, l'attributo o l'insieme di attributi identificatori dell'Entità costituiranno la chiave primaria della Tabella. Il dominio di ciascun attributo sarà opportunamente scelto in base alle Regole Aziendali ed ai requisiti della Base di Dati. In tale fase si ignorano le Entità *figlie* delle Generalizzazioni che verranno appositamente trattate al passo 1-bis.

   1-bis. Rappresentazione delle Generalizzazioni. Premesso che non è possibile dare delle regole generali per la rappresentazione delle Generalizzazioni e che ogni caso andrebbe valutato dal progettista e trattato in modo specifico si possono, tuttavia, dare delle linee guida applicabili in alcuni casi particolari e molto semplici:

   a. Le Entità *figlie* non possiedono attributi propri. In questo caso si aggiungono all'Entità *padre* tanti attributi di tipo *bit* per quante sono le Entità *figlie*. Tali attributi assumeranno valore 1 se l'Entità *padre* è anche del tipo dell'Entità *figlia* corrispondente.
   Consideriamo l'esempio seguente:



   Applicando la regola sopra definita avremo:

   TABLE **Artista (** *Codice* longint, Cog*nome* varchar(20), *Nome* varchar(20), *DataDiNascita* date, *Nazionalità* varchar(20), *Cantante* bit, *Compositore* bit, *Musicista* bit, *Autore* bit, primary key(*Codice*) **);**

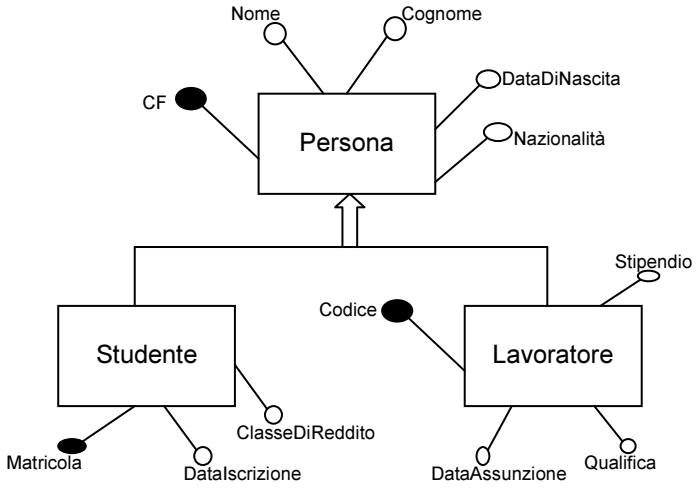
   Ad esempio, un Artista potrebbe avere il valore 1 per gli attributi *Cantante* ed *Autore* ed il valore 0 per gli attributi *Musicista* e *Compositore*. Se invece si vuole impedire che più attributi abbiano il valore 1 (generalizzazione esclusiva) si possono: (i) inserire uno o più

vincoli di tupla; (ii) sostituire gli attributi di tipo bit con un solo attributo di tipo smallint ed associare ad ogni Entità *figlia* un particolare valore intero, ad esempio:

TABLE **Artista (** *Codice* longint, Cog*nome* varchar(20), *Nome* varchar(20), *DataDiNascita* date, *Nazionalità* varchar(20), *TipoArtista* smallint, primary key(*Codice*) **);**

con la convenzione che *TipoArtista* avrà valore1 se l'Artista è un Cantante, valore 2 se è un Compositore, valore 3 se è un Musicista e valore 4 se è Autore. Ovviamente sarà necessario un vincolo sui dati che impedisca all'attributo di assumere un valore più piccolo di 1 e più grande di 4 (check (TipoArtista>=1 and TipoArtista<=4)) se la generalizzazione è totale[1] (non esistono altre tipologie di artisti). Se, invece, la generalizzazione è parziale, cioè esistono altre tipologie di artisti oltre a quelle specificate dalle Entità figlie (ad esempio Attore), è ammissibile per *TipoArtista* il valore 0, che indica appunto che all'artista non è assegnabile nessuno dei tipi rappresentati dalle Entità *figlie*, in tal caso il vincolo diventa: check (TipoArtista>=0 and TipoArtista<=4).
Un caso particolare si ha quando le Entità *figlie* sono due e la generalizzazione è esclusiva e totale, in questo caso basta aggiungere un solo attributo di tipo bit il cui nome può essere quello di una delle due Entità *figlie*. Il suo valore sarà pari a 1 se l'Entità padre è del tipo dell'Entità *figlia* corrispondente al nome dell'attributo, 0 se l'Entità *padre* è del tipo dell'altra Entità *figlia*.

b. Le Entità *figlie* possiedono attributi propri. In questo caso a ciascuna Entità *figlia* corrisponderà una Tabella nella quale saranno riportati tutti gli attributi propri dell'Entità *figlia* più un attributo che si riferisce alla chiave primaria dell'Entità *padre*.
Consideriamo l'esempio seguente:



Applicando la regola sopra definita avremo:

TABLE **Persona (** *CF* char(16), Cog*nome* varchar(20), *Nome* varchar(20), *DataDiNascita* date, *Nazionalità* varchar(20), primary key(*CF*) **);**

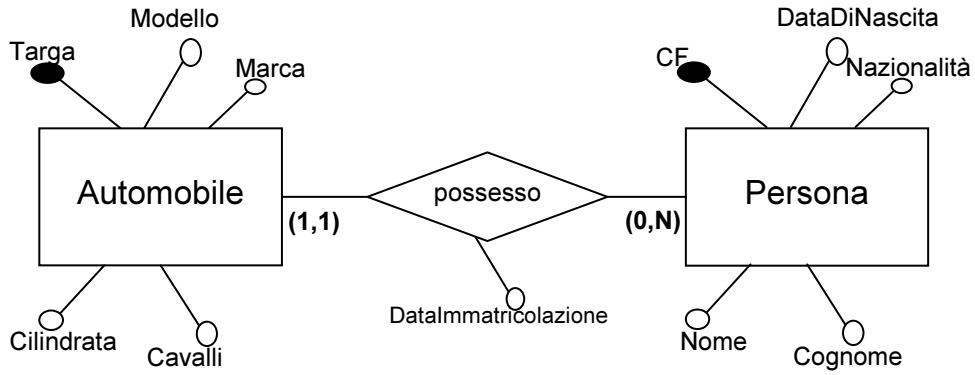
TABLE **Studente (** *Matricola* longint, *DataIscrizione* date, *ClasseDiReddito* smallint, *CodiceFiscale* char(16), primary key(*Matricola*), foreign key(*CodiceFiscale*) references Persona(*CF*) **);**

TABLE **Lavoratore (** *Codice* longint, *DataAssunzione* date, *Qualifica* varchar(20), *Stipendio* currency, *CodiceFiscale* char(16), primary key(*Codice*), foreign key(*CodiceFiscale*) references Persona(*CF*) **);**

c. Alcune Entità *figlie* possiedono attributi propri ed altre non possiedono attributi propri. In questo caso si combinano i due approcci precedentemente discussi.

[1] Una generalizzazione totale è rappresentata disegnando la freccia con tratto pieno

2. **Rappresentazione delle Associazioni uno a molti**. Le Associazione uno a molti presenti nel Diagramma E-R si rappresentano aggiungendo alla Tabella rappresentante l'Entità raffigurata al *lato uno* dell'associazione uno o più attributi che si riferiscono alla chiave primaria della Tabella rappresentante l'Entità raffigurata al *lato enne* dell'associazione. Si aggiungono inoltre, sempre alla Tabella rappresentante l'Entità raffigurata al *lato uno* dell'associazione, eventuali attributi propri dell'Associazione.
   Consideriamo l'esempio seguente:

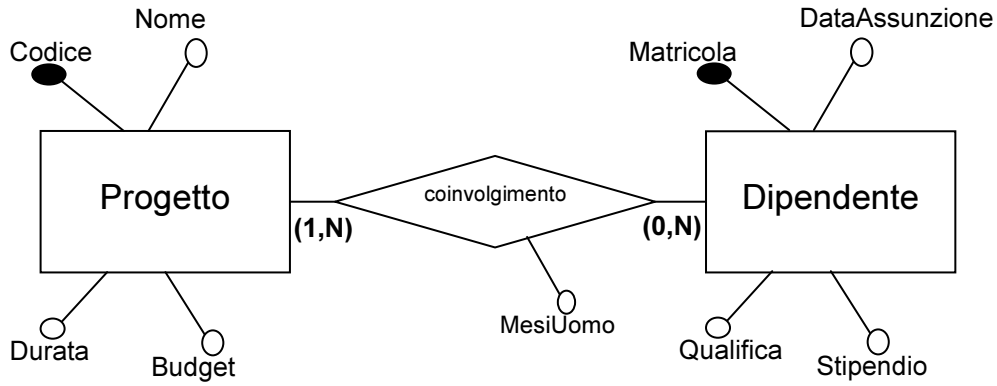


   L'Associazione uno a molti *possesso* lega le automobili alle persone specificando che una persona può possedere al minimo nessun automobile ed al massimo N automobili e che una automobile ha uno ed un solo proprietario. Applicando la regola sopra definita avremo:

   TABLE **Automobile (** *Targa* char(7), *Modello* varchar (20), *Marca* varchar (20), *Cilindrata* int, *Cavalli int, CodiceProprietario* char(16), *DataImmatricolazione* date, primary key(*Targa*), foreign key(*CodiceProprietario*) references Persona(*CF*) **);**

   TABLE **Persona (** *CF* char(16), Cog*nome* varchar(20), *Nome* varchar(20), *DataDiNascita* date, *Nazionalità* varchar(20), primary key(*CF*) **);**

3. **Rappresentazione delle Associazioni molti a molti**. Le Associazioni molti a molti presenti nel Diagramma E-R si rappresentano mediante una Tabella avente come nome quello dell'Associazione e come attributi un insieme di attributi che si riferiscono alle chiavi primarie delle Tabelle rappresentanti le Entità correlate dall'Associazione più gli (eventuali) attributi propri dell'Associazione. La chiave primaria della "nuova" Tabella sarà costituita dall'insieme di attributi che si riferiscono alle chiavi primarie delle Tabelle rappresentanti le Entità correlate dall'Associazione.
   Consideriamo l'esempio seguente:



   L'Associazione molti a molti *coinvolgimento* lega i dipendenti ai progetti specificando che un dipendente può essere coinvolto al minimo in nessun progetto ed al massimo in N progetti e che un progetto può vedere coinvolti al minimo un dipendente ed al massimo N dipendenti. Applicando la regola sopra definita avremo:

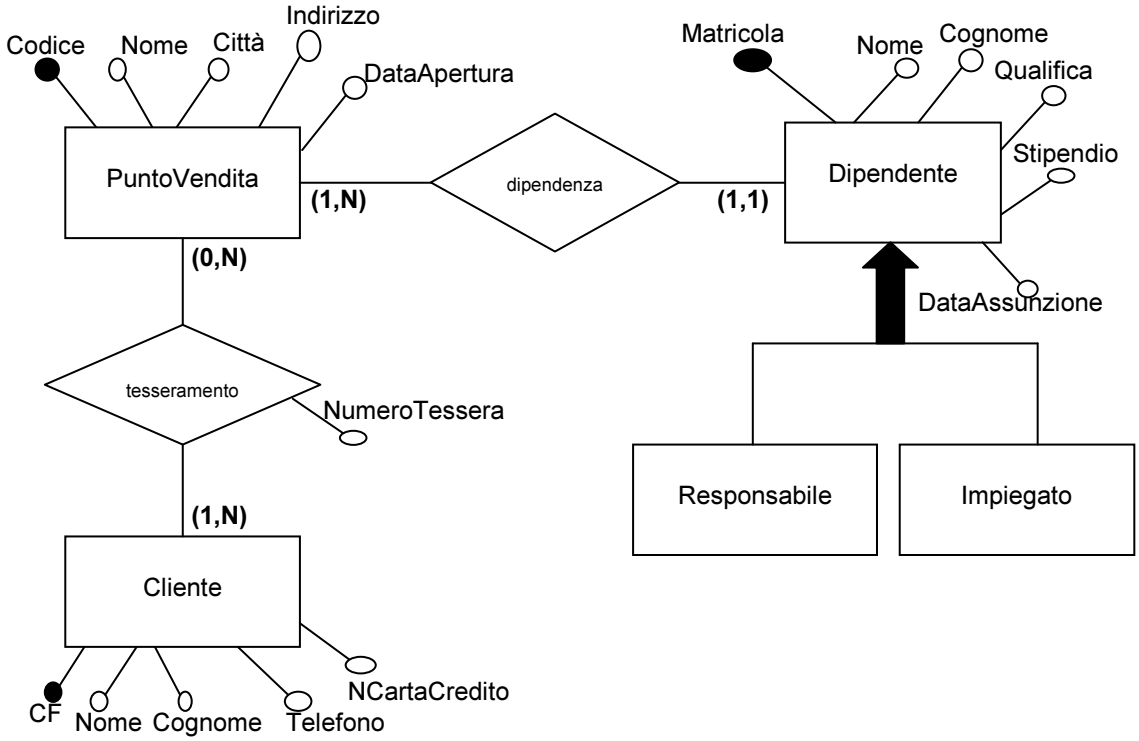
TABLE **Dipendente (** *Matricola* longint, *Qualifica* varchar(20), *DataAssunzione* date, *Stipendio* currency, primary key(*Matricola*) **);**

TABLE **Progetto (** *Codice* char(4), *Nome* varchar(50), *durata* smallint, *budget* currency, primary key(*Codice*) **);**

TABLE **Coinvolgimento (** *CodiceDipendente* longint, *CodiceProgetto* char(4), *MesiUomo* smallint, primary key(*CodiceDipendente, CodiceProgetto*), foreign key(*CodiceDipendente*) references Dipendente(*Matricola*), foreign key(*CodiceProgetto*) references Progetto(Codice) **);**

4. **Rappresentazione dei Vincoli sui Dati.** Si arricchiscono le Tabelle sinora definite dei *vincoli sui valori* degli attributi e dei *vincoli di tupla* ricavati in base alle Regole Aziendali ed ai requisiti della Base di Dati.

Al fine di chiarire ulteriormente i passi sopra elencati progettiamo lo Schema Logico Relazione corrispondente al seguente Diagramma E-R:



PASSO 1: Rappresentazione delle Entità.
Nel Diagramma E-R, ignorando le Generalizzazioni, sono presenti 3 Entità: Dipendente, Punto Vendita e Cliente. In base agli attributi E-R indicati nel Diagramma si ottengono le seguenti Tabelle:

TABLE **Dipendente (** *Matricola* longint, *Nome* varchar (20), *Cognome* varchar (20), *Qualifica* varchar(20), *DataAssunzione* date, *Stipendio* currency, primary key(*Matricola*) **);**

TABLE **PuntoVendita (** *Codice* char(5), *Nome* varchar(50), *Città* varchar (50), *Indirizzo* varchar(70), *DataApertura* date, primary key(*Codice*) **);**

TABLE **Cliente (** *CF* char(16), *Nome* varchar(20), *Cognome* varchar(20), *Telefono* varchar(15), *NCartaCredito* char(16), primary key(*CF*) **);**

PASSO 1-bis: Rappresentazione delle Generalizzazioni.
Abbiamo una sola generalizzazione che coinvolge l'Entità Dipendente. Le Entità *figlie* (Impiegato e Responsabile) non hanno attributi propri, quindi, ricadiamo nel caso a. del passo 1-bis; inoltre, la generalizzazione è totale (non esistono altre tipologie di dipendenti) ed esclusiva poiché un dipendente o è un semplice impiegato oppure è un responsabile. Possiamo quindi aggiungere, come indicato al termine del caso a. , un unico attributo di tipo bit che chiameremo *Responsabile*, se tale attributo avrà valore 1 allora ci troveremo di fronte ad un dipendente con incarico di responsabile del punto vendita, se, invece, tale attributo avrà valore 0 allora ci troveremo di fronte ad un dipendente con mansioni di semplice impiegato.

TABLE **Dipendente (** *Matricola* longint, *Nome* varchar (20), *Cognome* varchar (20), *Qualifica* varchar(20), *DataAssunzione* date, *Stipendio* currency, *Responsabile* bit, primary key(*Matricola*) **);**

TABLE **PuntoVendita (** *Codice* char(5), *Nome* varchar(50), *Città* varchar (50), *Indirizzo* varchar(70), *DataApertura* date, primary key(*Codice*) **);**

TABLE **Cliente (** *CF* char(16), *Nome* varchar(20), *Cognome* varchar(20), *Telefono* varchar(15), *NCartaCredito* char(16), primary key(*CF*) **);**

PASSO 2: Rappresentazione delle Associazioni uno a molti.
Abbiamo una sola associazione una a molti, l'associazione *dipendenza* che lega i dipendenti ai punti vendita specificando che un punto vendita può avere al minimo un dipendente ed al massimo N dipendenti e che un dipendente lavora in uno ed un solo punto vendita. Seguendo le regole relative al passo 2 e tenendo presente che l'Entità *lato uno* è il Dipendente mentre L'Entità *lato enne* è il Punto Vendita si ottiene:

TABLE **Dipendente (** *Matricola* longint, *Nome* varchar (20), *Cognome* varchar (20), *Qualifica* varchar(20), *DataAssunzione* date, *Stipendio* currency, *Responsabile* bit, *CodicePuntoVendita* char (5), primary key(*Matricola*), foreign key(*CodicePuntoVendita*) references PuntoVendita(*Codice*) **);**

TABLE **PuntoVendita (** *Codice* char(5), *Nome* varchar(50), *Città* varchar (50), *Indirizzo* varchar(70), *DataApertura* date, primary key(*Codice*) **);**

TABLE **Cliente (** *CF* char(16), *Nome* varchar(20), *Cognome* varchar(20), *Telefono* varchar(15), *NCartaCredito* char(16), primary key(*CF*) **);**

PASSO3: Rappresentazione delle Associazioni moli a molti.
Abbiamo una sola associazione molti a molti, l'associazione *tesseramento* che lega i punti vendita ed i clienti specificando che un punto vendita può tesserare al minimo nessun cliente ed al massimo N clienti e che un cliente può essere tesserato al minimo con un punto vendita ed al massimo con N punti vendita. L'associazione *tesseramento* possiede, inoltre, un attributo proprio indicante il numero della tessera del cliente. Seguendo le regole relative al passo 3 si ottiene:

TABLE **Dipendente (** *Matricola* longint, *Nome* varchar (20), *Cognome* varchar (20), *Qualifica* varchar(20), *DataAssunzione* date, *Stipendio* currency, *Responsabile* bit, *CodicePuntoVendita* char (5), primary key(*Matricola*), foreign key(*CodicePuntoVendita*) references PuntoVendita(*Codice*) **);**

TABLE **PuntoVendita (** *Codice* char(5), *Nome* varchar(50), *Città* varchar (50), *Indirizzo* varchar(70), *DataApertura* date, primary key(*Codice*) **);**

TABLE **Cliente (** *CF* char(16), *Nome* varchar(20), *Cognome* varchar(20), *Telefono* varchar(15), *NCartaCredito* char(16), primary key(*CF*) **);**

TABLE **Tesseramento (** *CodiceCliente* char(16), *CodicePuntoVendita* char(5), *NumeroTessera* longint,

primary key(*CodiceCliente, CodicePuntoVendita*),
foreign key(*CodiceCliente*) references Cliente(*CF*),
foreign key(*CodicePuntoVendita*) references PuntiVendita(*Codice*) **);**

PASSO 4: Rappresentazione dei Vincoli sui Dati.
Pur non avendo a disposizione le Regole Aziendali a corredo del Diagramma E-R possiamo sulla base di ovvie considerazioni inserire i seguenti vincoli sui dati:

TABLE **Dipendente (** *Matricola* longint, *Nome* varchar (20), *Cognome* varchar (20) not null, *Qualifica* varchar(20), *DataAssunzione* date not null, *Stipendio* currency not null, *Responsabile* bit default 0, *CodicePuntoVendita* char (5), primary key(*Matricola*), foreign key(*CodicePuntoVendita*) references PuntoVendita(*Codice*), check(*DataAssunzione*>10/06/2002) **);**

TABLE **PuntoVendita (** *Codice* char(5), *Nome* varchar(50), *Città* varchar (50) not null, *Indirizzo* varchar(70) not null, *DataApertura* date, primary key(*Codice*), check(*DataApertura*>20/06/2002) **);**

TABLE **Cliente (** *CF* char(16), *Nome* varchar(20), *Cognome* varchar(20), *Telefono* varchar(15) not null, *NCartaCredito* char(16), primary key(*CF*) **);**

TABLE **Tesseramento (** *CodiceCliente* char(16), *CodicePuntoVendita* char(5), *NumeroTessera* longint not null,
primary key(*CodiceCliente, CodicePuntoVendita*),
foreign key(*CodiceCliente*) references Cliente(*CF*),
foreign key(*CodicePuntoVendita*) references PuntiVendita(*Codice*),
unique(*NumeroTessera*),
check(*NumeroTessera*>0) **);**

Lo Schema Logico Relazionale della Base di Dati risultato dalla fase di progettazione Logica sarà quindi:

DATABASE *PuntiVendita*
{

TABLE **Dipendente (** *Matricola* longint, *Nome* varchar (20), *Cognome* varchar (20) not null, *Qualifica* varchar(20), *DataAssunzione* date not null, *Stipendio* currency not null, *Responsabile* bit default 0, *CodicePuntoVendita* char (5), primary key(*Matricola*), foreign key(*CodicePuntoVendita*) references PuntoVendita(*Codice*), check(*DataAssunzione*>10/06/2002) **);**

TABLE **PuntoVendita (** *Codice* char(5), *Nome* varchar(50), *Città* varchar (50) not null, *Indirizzo* varchar(70) not null, *DataApertura* date, primary key(*Codice*), check(*DataApertura*>20/06/2002) **);**

TABLE **Cliente (** *CF* char(16), *Nome* varchar(20), *Cognome* varchar(20), *Telefono* varchar(15) not null, *NCartaCredito* char(16), primary key(*CF*) **);**

TABLE **Tesseramento (** *CodiceCliente* char(16), *CodicePuntoVendita* char(5), *NumeroTessera* longint not null,
primary key(*CodiceCliente, CodicePuntoVendita*),
foreign key(*CodiceCliente*) references Cliente(*CF*),
foreign key(*CodicePuntoVendita*) references PuntiVendita(*Codice*),
unique(*NumeroTessera*),
check(*NumeroTessera*>0) **);**

}