Progettazione Basi di Dati

Sistemi Informativi (Ingegneria Gestionale)

Dott. Michele Zanella

AA 2018-2019

1 Ristrutturazione Schema E/R

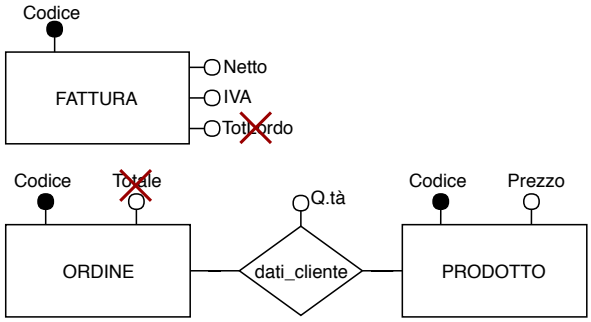
Si suddivide in diverse fasi:

1. Analisi delle ridondanze
2. Ristrutturazione delle gerarchie
3. Ristrutturazione attributi
4. Partizionamento/Merging delle Entità e Relazioni

1.1 Analisi delle ridondanze

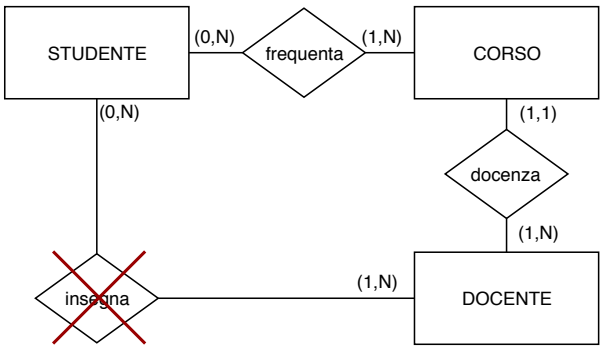
Ridondanza: la stessa informazione viene presentata due (o più) volte in modi differenti
La presenza di una ridondanza può essere vantaggiosa in termini di minori accessi necessari per derivare un'informazione. Tuttavia, può essere svantaggiosa in termini di spazio di archiviazione e la necessità di effettuare operazioni addizionali per mantenere le informazioni derivate consistenti.

- *Ridondanze negli attributi* L'obiettivo è eliminare attributi ridondanti, ad esempio quelli calcolabili. Nell'esempio *TotLordo* di FATTURA può essere calcolata utilizzando gli attributi *Netto* e *IVA*.



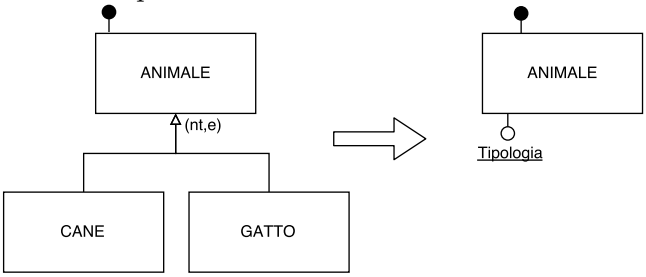
- *Ridondanze schematiche*

  1. Verificare la presenza di cicli: in caso di assenza di cicli non c'è ridondanza. **Attenzione:** la presenza di cicli è condizione necessaria ma non sufficiente per il verificarsi della ridondanza.
  2. Verificare se, partendo da un'entità $E1$, è possibile ritornare alla stessa entità attraverso un percorso ciclico univoco. Un percorso univoco è tale quando coinvolge almeno nela direzione considerata tutte relazioni uno a molti.
  3. Verificare ridondanze logiche
     Nel caso in cui ci sia un ciclo e il metodo del punto precedente non è applicabile, per ogni associazione del ciclo verificare se è ridondante: si sceglie una associazione, la si elimina e si verifica se si ottiene una perdita di informazioni. Nell'esempio sotto la relazione *insegna* è ridondante poichè se un docente insegna in un corso e tale corso è frequentato dagli studenti è possibile ricavare a quali studenti il docente insegna senza necessità di mantenere l'associazione.



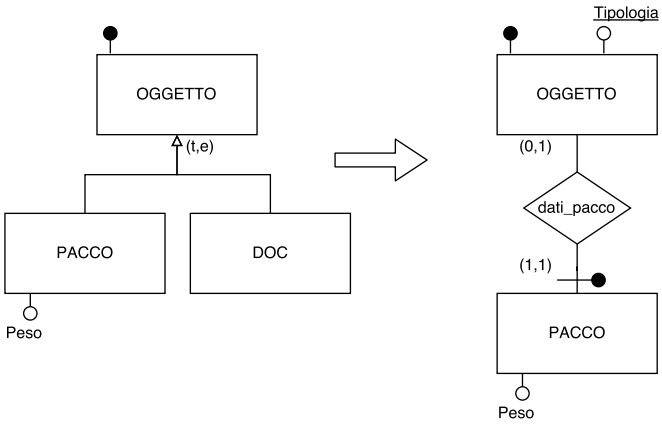
1.2 Ristrutturazione delle gerarchie

- Gerarchia parziale ed esclusiva.



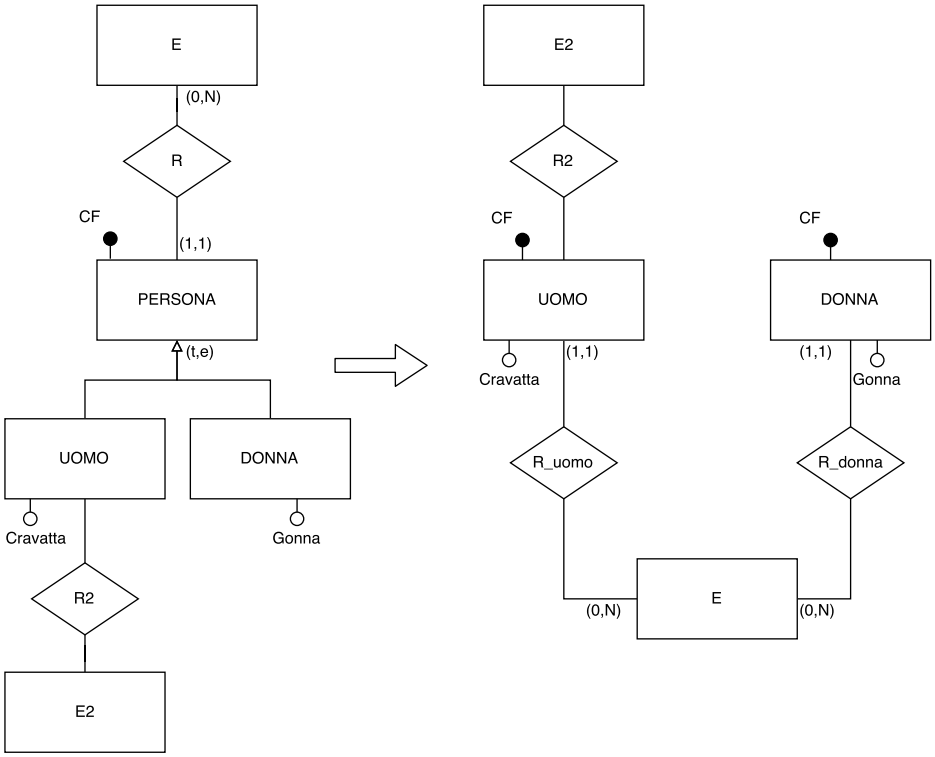
  Le due entità figlie (specializzazioni) vengono accorpate nell'entità padre ANIMALE e viene aggiunto un attributo per specificarne la *Tipologia*.

- Gerarchia totale ed esclusiva con attributo in una delle specializzazioni.



Si utilizza una strategia mista, l'entità DOC viene accorpata nell'entità OGGETTO in quanto priva di propri attributi. Viene aggiunto l'attributo *Tipologia* nell'entità OGGETTO per poterne distinguere i vari tipi.
L'entità PACCO viene mantenuta in quanto contiene almeno un attributo proprio. Si definisce una nuova associazione uno a uno tra OGGETTO e PACCO denominata *dati_pacco*. Infine, l'entità PACCO siccome contiene informazioni relative a specifici oggetti viene resa debole rispetto ad OGGETTO e con essa ne condivide l'identificatore primario.
La partecipazione sarà opzionale per l'entità OGGETTO in quanto non tutti gli oggetti sono dei pacchi e quindi hanno i relativi dati, mentre è obbligatoria per l'entità PACCO in quanto debole.

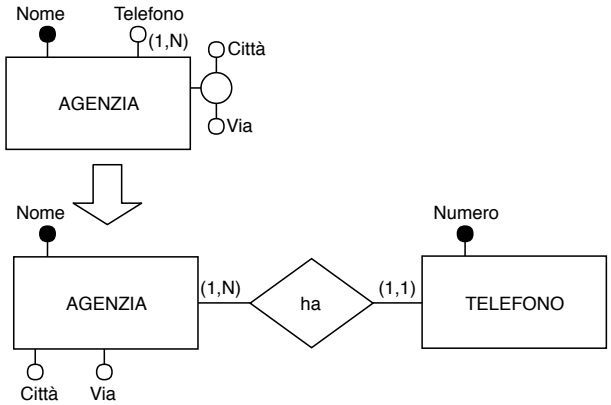
- Gerarchia totale ed esclusiva con attributi propri delle specializzazioni, associazione nell'entità padre e associazione in una delle specializzazioni.



Si mantengono solo le specializzazioni: tutti gli attributi dell'entità PERSONA vengono aggiunti nelle entità UOMO e DONNA.
L'associazione R tra PERSONA ed E viene replicata per ogni entità figlia: vengono aggiunte le associazioni *R_uomo* tra UOMO ed R ed *R_donna* tra DONNA ed R.
L'associazione *R2* tra UOMO ed E2 viene mantenuta immutata.

1.3 Ristrutturazione attributi

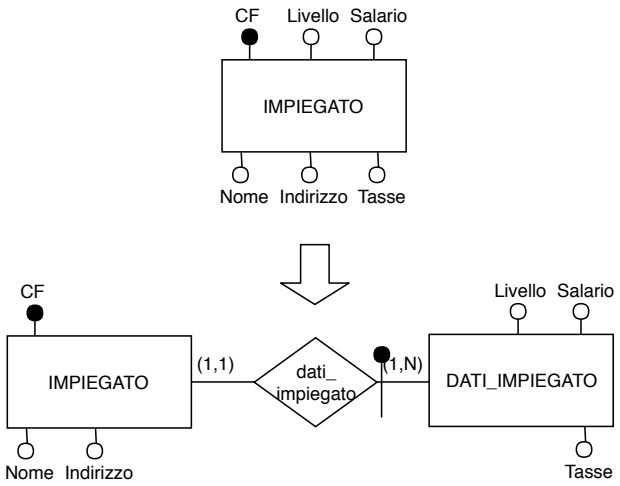
- Eliminare attributi multivalore
- Eliminare attributi composti



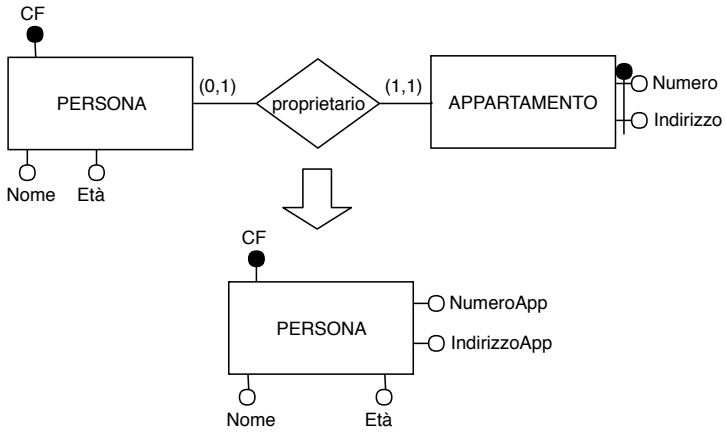
1.4 Partizionamento/Merging

Si utilizza per aumentare l'efficienza delle operazioni.

- Partizionamento: separare alcuni informazioni di un'entità definendo un'entità debole



- Merging: accorpare due entità e relative informazioni in una stessa entità



2 Progettazione logica

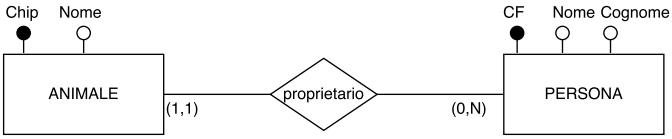
2.1 Traduzione delle entità

- Per ogni entità si crea una tabella
- La tabella risultante avrà come chiave primaria l'identificatore primario dell'entità originale

- La tabella conterrà gli attributi dell'entità originale più eventuali attributi risultanti dalla traduzione delle associazioni che coinvolgono la tabella come spiegato nella prossima sottosezione.

2.2 Traduzione delle associazioni in logico

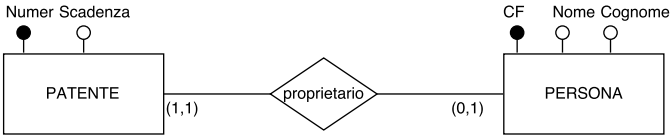
- Associazione uno a molti opzionale



Le due entità vengono tradotte direttamente in due tabelle.
L'associazione viene tradotta aggiungendo l'identificatore esterno della tabella PERSONA nella tabella ANIMALE.
ANIMALE(Chip, Nome, *CF_Persona*)
PERSONA(CF, Nome, Cognome)

Viene definito anche il relativo vincolo di intergrità referenziale:
Animale.CF_Persona → Persona.CF

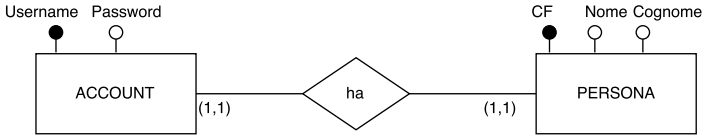
- Associazione uno a uno opzionale



Le due entità vengono tradotte direttamente in due tabelle.
L'associazione viene tradotta aggiungendo l'identificatore esterno della tabella PERSONA nella tabella PATENTE, in quanto è l'entità con partecipazione obbligatoria.
PATENTE(Numero, Scadenza, *CF_Persona*)
PERSONA(CF, Nome, Cognome)

Viene definito anche il relativo vincolo di intergrità referenziale:
Patente.CF_Persona → Persona.CF

- Associazione uno a uno obbligatoria



Le due entità vengono tradotte direttamente in due tabelle.
L'associazione viene tradotta aggiungendo l'identificatore esterno della tabella PERSONA nella tabella ACCOUNT, o viceversa, in ogni caso solo in una delle due per evitare ridodanze.
ACCOUNT(Username, Password, *CF_Persona*)

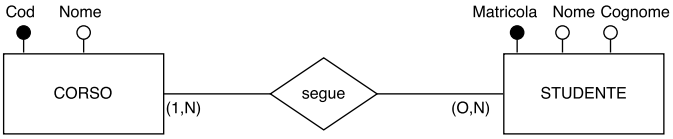
PERSONA(<u>CF</u>, Nome, Cognome)

Viene definito anche il relativo vincolo di intergrità referenziale:
Account.CF_Persona → Persona.CF

Oppure:
ACCOUNT(<u>Username</u>, Password)
PERSONA(<u>CF</u>, Nome, Cognome, *Username_Account*)

Viene definito anche il relativo vincolo di intergrità referenziale:
Persona.Username_Account → Account.Username

- Associazione molti a molti obbligatoria o opzionale



Le due entità vengono tradotte direttamente in due tabelle.
L'associazione viene tradotta direttamente con una tabella ponte denominata CORSO_STUDENTE oppure ISCRIZIONE. Questa tabella avrà la chiave composta dalle rispettive chiavi delle tabelle CORSO e STUDENTE che saranno anche gli identificatori esterni di tali tabelle.
CORSO(<u>Cod</u>, Nome)
STUDENTE(<u>Matricola</u>, Nome, Cognome)
ISCRIZIONE(<u>*Matricola_Studente*</u>, <u>*Cod_Corso*</u>)

Vengono definiti anche i relativi vincoli di intergrità referenziale:
Iscrizione.Matricola_Studente → Studente.Matricola
Iscrizione.Cod_Corso → Corso.Cod