



Basi Dati

Modello Entità Relazioni Esteso

a.a. 2021/2022
Prof.ssa G. Tortora

Perché migliorare l'ER?

- Il modello ER è nato negli anni '70. Da allora il mondo dei database ha compiuto progressi immensi.

Esempi:

- GIS,
 - Database Multimediali, con video, audio, ecc...
 - OLAP – Data Mining – Data Warehousing,
 - Motori di ricerca per Internet,
 - ...
- Tali database hanno requisiti molto complessi, non sempre rappresentabili con l'ER.

Limiti dell'ER

- Il modello ER, infatti, pone dei limiti alla modellazione di un miniworld:
 - Non possono essere definite relazioni tra un tipo di entità ed un tipo di relazione.
 - Non possono essere definite relazioni tra un tipo di entità ed una collezione di tipi di entità, in cui ogni tipo può partecipare alla relazione.
 - Non è possibile specificare vincoli tra tipi di relazioni (**es:** esclusione, coesistenza, ecc...).
 - Mancano i concetti di **generalizzazione** e **specializzazione**.

Il modello Enhanced Entity-Relationship



Il modello EER

- Il modello EER (ER esteso) include tutti i concetti di modellazioni propri dell'ER.

In più aggiunge i concetti di:

- Sottoclassi e Superclassi;
 - Specializzazione e Generalizzazione;
 - Categorie;
 - Ereditarietà degli attributi.
-
- È in pratica un modello ER con l'aggiunta di concetti di Object-Orientation.

Sottoclassi, Superclassi ed Ereditarietà



Sottoclasse / Superclasse

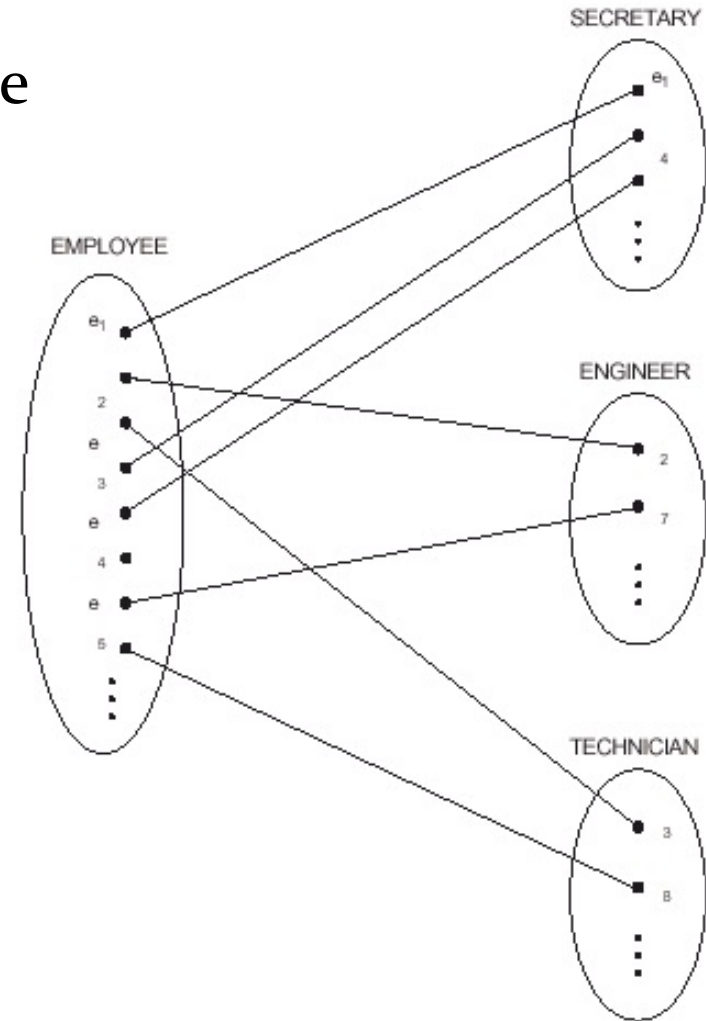
- Un tipo entità può avere dei **sottoraggruppamenti** delle sue entità in base a particolari caratteristiche.
In determinati contesti può essere interessante o necessario rappresentare tali raggruppamenti, ad esempio per memorizzare informazioni specifiche.
- **Esempio:** Le entità di “*employee*” potrebbero essere raggruppate in:
 - Segretarie (con il nuovo attributo “*velocità di digitazione*”)
 - Ingegnere (con il nuovo attributo “*tipo di ingegnere*”)
 - Tecnici (con il nuovo attributo “*specializzazione*”)
 - Impiegati Part-time (con il nuovo attributo “*paga oraria*”)
 - ...

Sottoclasse / Superclasse (2)

- I raggruppamenti appena individuati sono detti **sottoclassi** di Employee, ed Employee è detta **superclasse** di ciascuna sottoclasse.
- Questo legame è detto **relazione classe/sottoclasse** o relazione **IS-A** (o IS-AN)
 - *Es:* Segretaria IS-AN Employee.

Sottoclasse / Superclasse (3)

- Da notare che un'entità della sottoclasse rappresenta la stessa entità della superclasse nel mondo reale.
 - La segretaria “*Joan Logano*” è anche l'impiegata “*Joan Logano*”.
 - L'entità della sottoclasse è la stessa della superclasse, ma in un **ruolo** diverso.
- Ciò implica che un'entità non può esistere nel db solo come membro di una sottoclasse, ma deve essere anche membro della superclasse.
 - Tale entità potrebbe però inclusa in più sottoclassi.



Ereditarietà degli attributi

- Poiché un membro di una sottoclasse è membro anche della superclasse, deve avere valori per tutti gli attributi della superclasse.
- Diciamo che un'entità membro di una sottoclasse **eredita** tutti gli attributi dell'entità superclasse. Eredita inoltre anche tutti i tipi di relazione in cui partecipa la superclasse.

Specializzazione e Generalizzazione



Specializzazione

- È il processo di definire un **insieme di sottoclassi** di un tipo entità, detto superclasse della specializzazione, sulla base di particolari caratteristiche.
- ***Esempio:***
L'insieme di sottoclassi *{Segretarie, Ingegneri, Tecnici}* è una specializzazione della superclasse “*Employee*” che distingue le entità in base al tipo di lavoro svolto.

Specializzazione (2)

- Possono esistere più specializzazioni dello stesso tipo di entità, in base a differenti caratteristiche.
 - *Es:* Possiamo suddividere gli impiegati in:
 - Impiegati a tempo fisso-Impiegati a part-time,
 - Mansione svolta,
 - Manager.
- Nell'EER le sottoclassi che definiscono una specializzazione sono unite con delle linee ad un cerchio connesso alla superclasse.

Il simbolo di sottoinsieme, posto sulla linea che congiunge la sottoclasse con il cerchio, descrive la direzione della relazione.

Specializzazione - Esempio

Tre specializzazioni di 'Employee':
{Segretarie, Ingegneri, Tecnici}
{Manager}
{Impiegati fissi, Impiegati Part-Time}

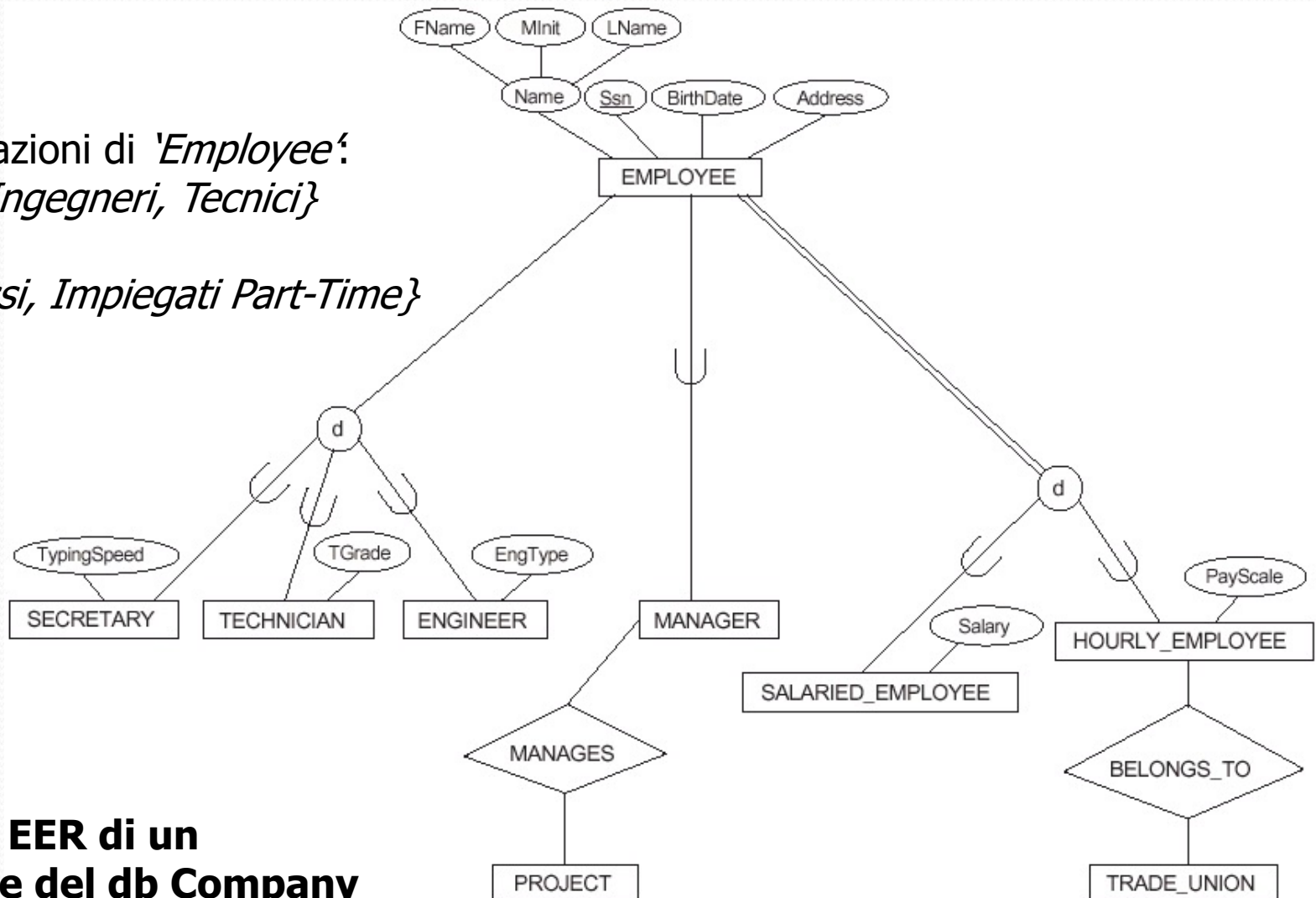


Diagramma EER di un sottoinsieme del db Company

Motivazioni per l'uso di sottoclassi/specializzazioni

- Certi attributi si possono applicare solo ad alcune entità del tipo entità superclasse.
Si definisce quindi una sottoclasse per raggruppare le entità a cui si applicano tali attributi.
 - L'attributo "*Specializzazione*" si applica solo agli impiegati "*Tecnici*".
- In alcuni tipi di relazioni possono partecipare solo entità appartenenti ad una sottoclasse.
 - La relazione "*Gestisce progetto*" si applica solo agli impiegati "*Manager*".

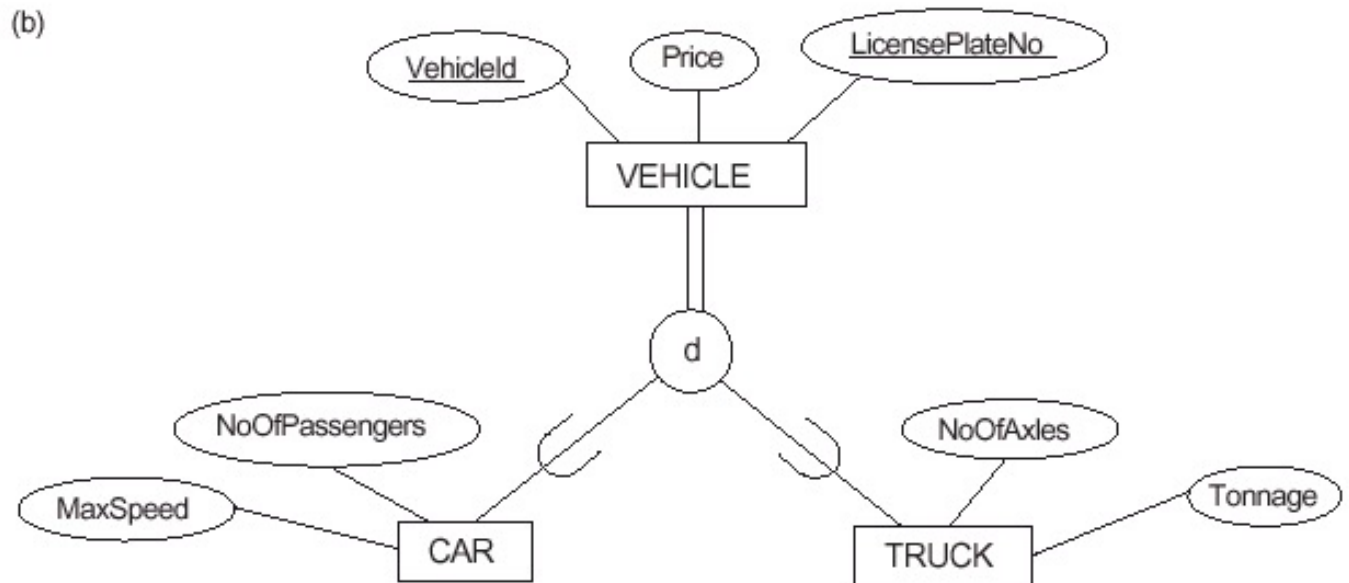
Generalizzazione

- Processo inverso di astrazione, in cui vengono sopprese le differenze tra i vari tipi di entità, sono identificate le caratteristiche comuni e si generalizzano le entità in una singola superclasse di cui i tipi di entità originali sono **sottoclassi speciali**.

Generalizzazione - *Esempio*



a) Due tipi di entità, 'CAR' e 'TRUCK'.



b) La loro generalizzazione nel tipo di entità 'VEHICLE'.

Vincoli e caratteristiche di specializzazione e generalizzazione



Vincoli di specializzazione e generalizzazione

- È possibile avere più specializzazioni definite sullo stesso tipo di entità (*superclasse*).

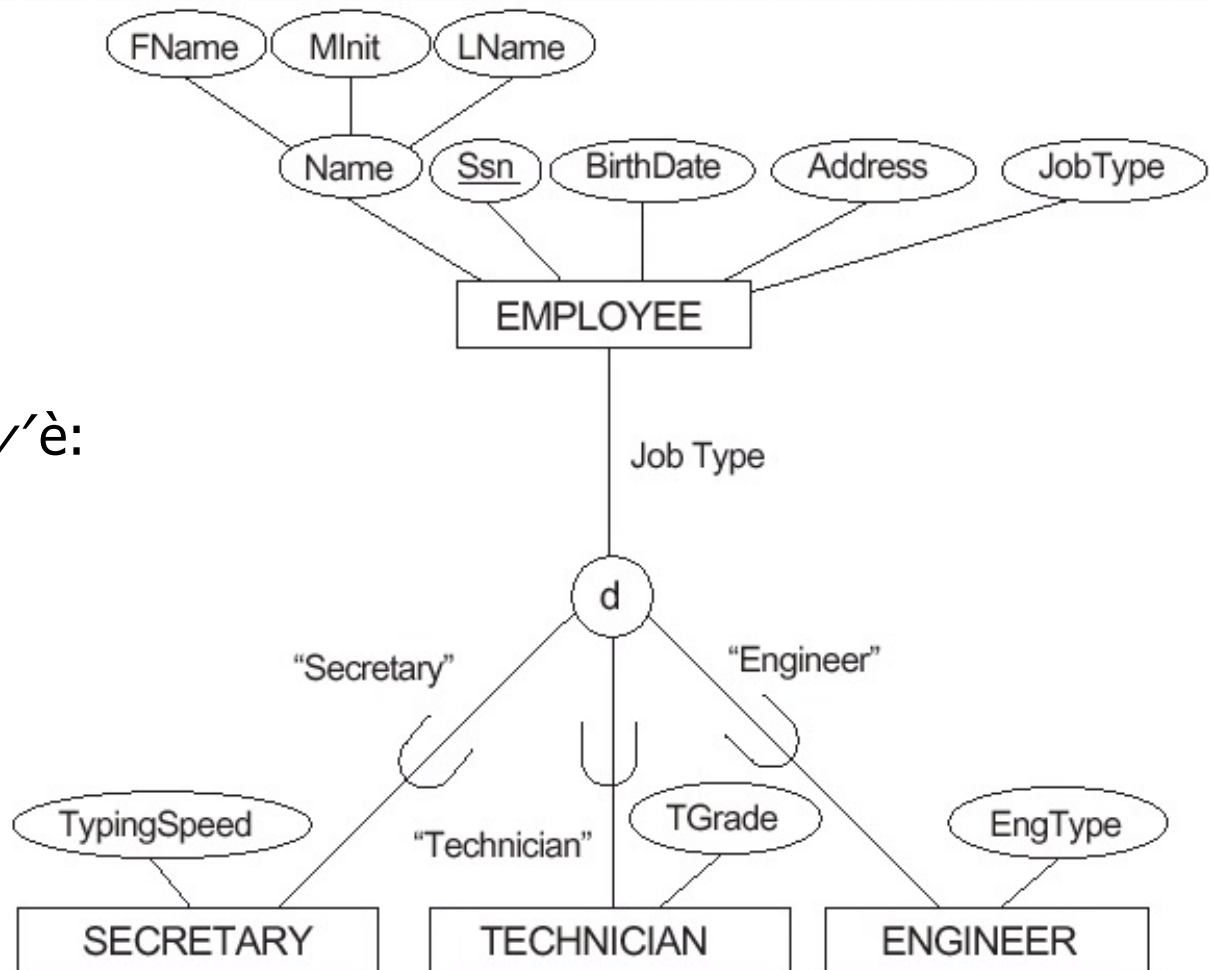
In tal caso, le entità possono comparire in sottoclassi per ogni specializzazione.

- In alcune specializzazioni possiamo determinare con esattezza le entità che diventeranno membri di ciascuna sottoclasse, ponendo una condizione sul valore di qualche attributo della superclasse.
 - Tali sottoclassi sono dette **sottoclassi predicate-defined** (o **condition-defined**).

Esempio di predicate-defined

La condizione di appartenenza alla sottoclasse 'Secretary' è:
JobType="Secretary"

Predicato che definisce la sottoclasse



Una specializzazione definita sull'attributo 'Job Type' di 'Employee'

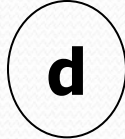
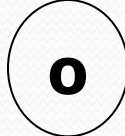
Vincoli di specializzazione e generalizzazione (2)

- Una sottoclasse predicate-defined è visualizzata riportando il nome della condizione vicino la linea.
- Se tutte le sottoclassi in una specializzazione hanno la condizione di appartenenza sullo stesso attributo, la specializzazione è detta **attribute-defined**.

Sottoclassi user-defined

- Se non abbiamo alcuna condizione per determinare l'appartenenza ad una sottoclasse, la sottoclasse è detta **user-defined**.
- In tal caso, l'appartenenza è specificata individualmente per ogni entità dall'utente e non da una condizione valutabile automaticamente.

Vincolo di disgiunzione

- Specifica che le sottoclassi di una specializzazione devono essere disgiunte.
 - Un'entità può essere membro di al più una sottoclasse.
 - Si denota con: 
- Se non vale tale vincolo, le entità possono avere un overlap.
 - La stessa entità può essere membro di più sottoclassi.
 - Si denota con: 

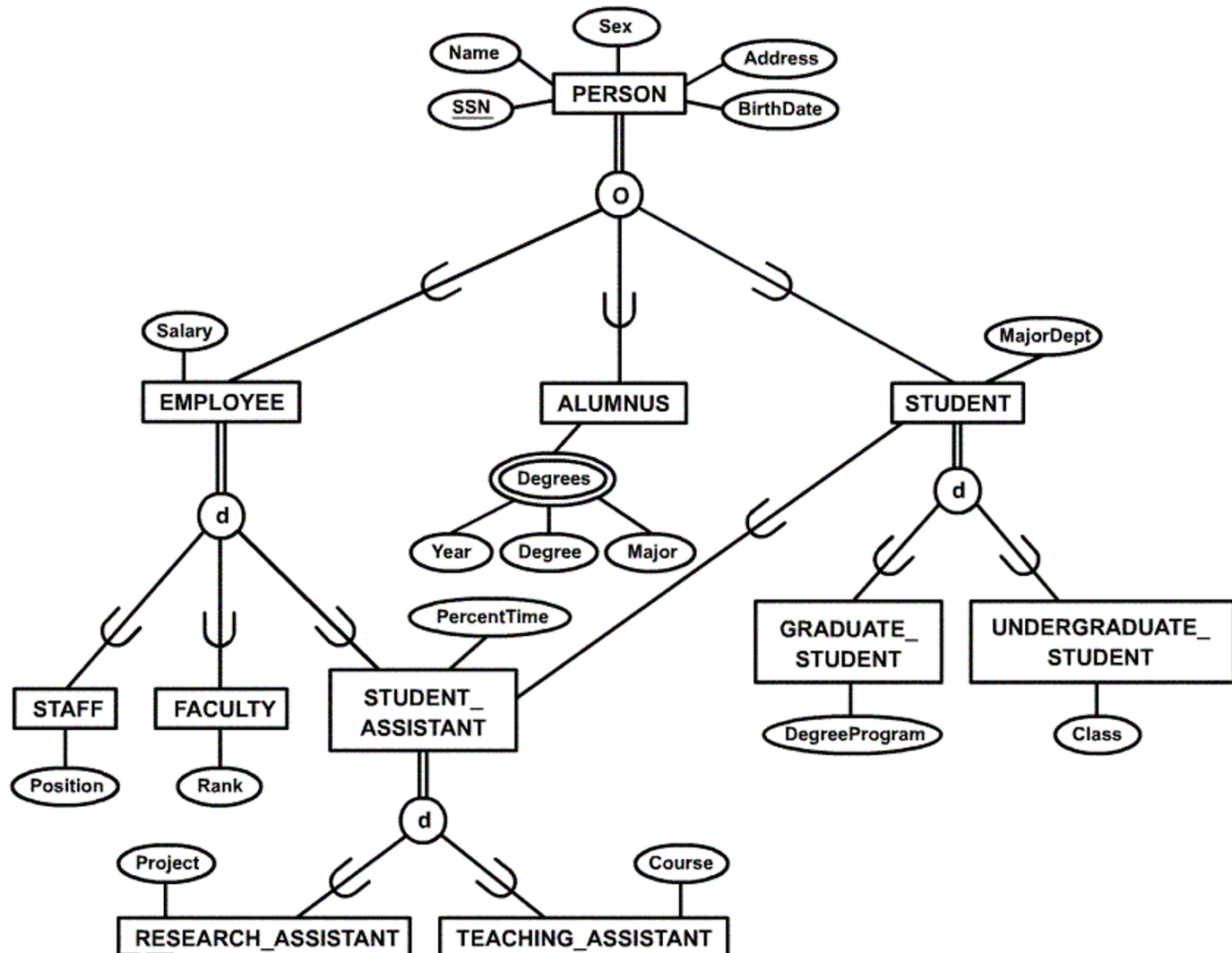
Vincolo di completezza (parziale o totale)

- Un vincolo di specializzazione totale specifica che ogni entità deve appartenere ad una sottoclasse.
 - Si denota con una linea doppia che collega la superclasse al cerchio.
- Una specializzazione parziale permette a qualche entità di non appartenere a nessuna sottoclasse.
 - Si denota con una linea singola che collega la superclasse al cerchio,

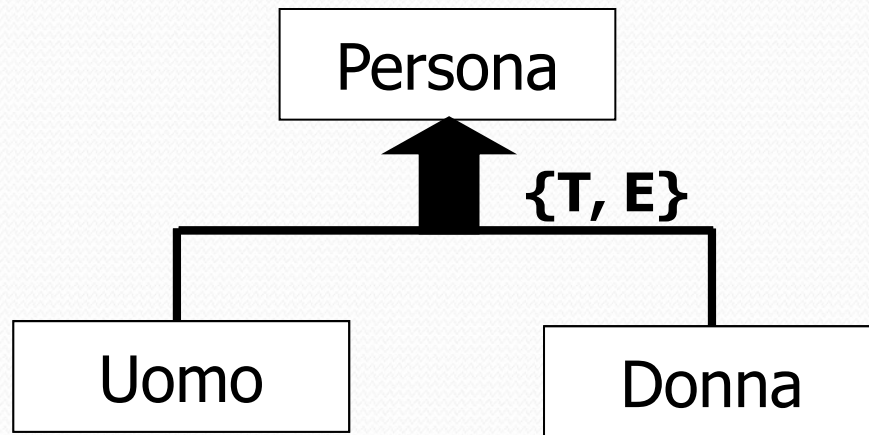
Combinazioni di vincoli

- I due vincoli sono indipendenti, pertanto sono ammesse tutte e quattro le combinazioni:
 - Disgiunto, totale;
 - Disgiunto, parziale;
 - Con Overlap, totale;
 - Con Overlap, parziale.
- Una superclasse di generalizzazione è in genere ‘*Totale*’, poiché la superclasse è combinazione delle sottoclassi.

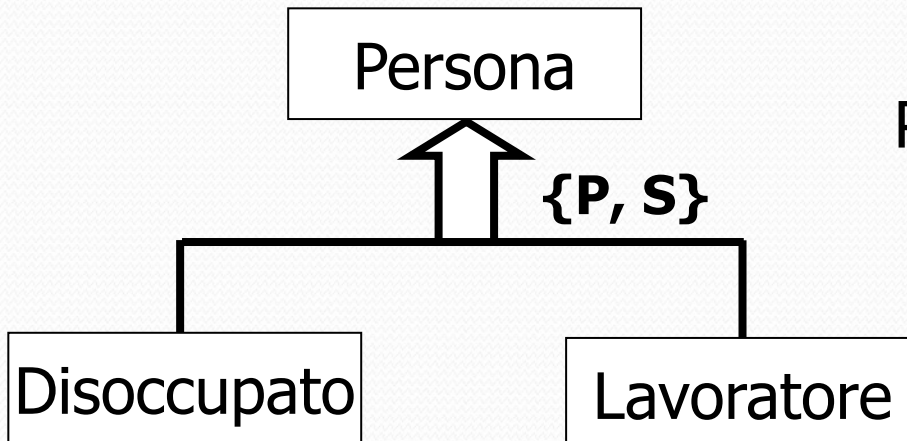
Esempio con i vincoli



Specializzazione/Generalizzazione: notazione grafica (Atzeni)



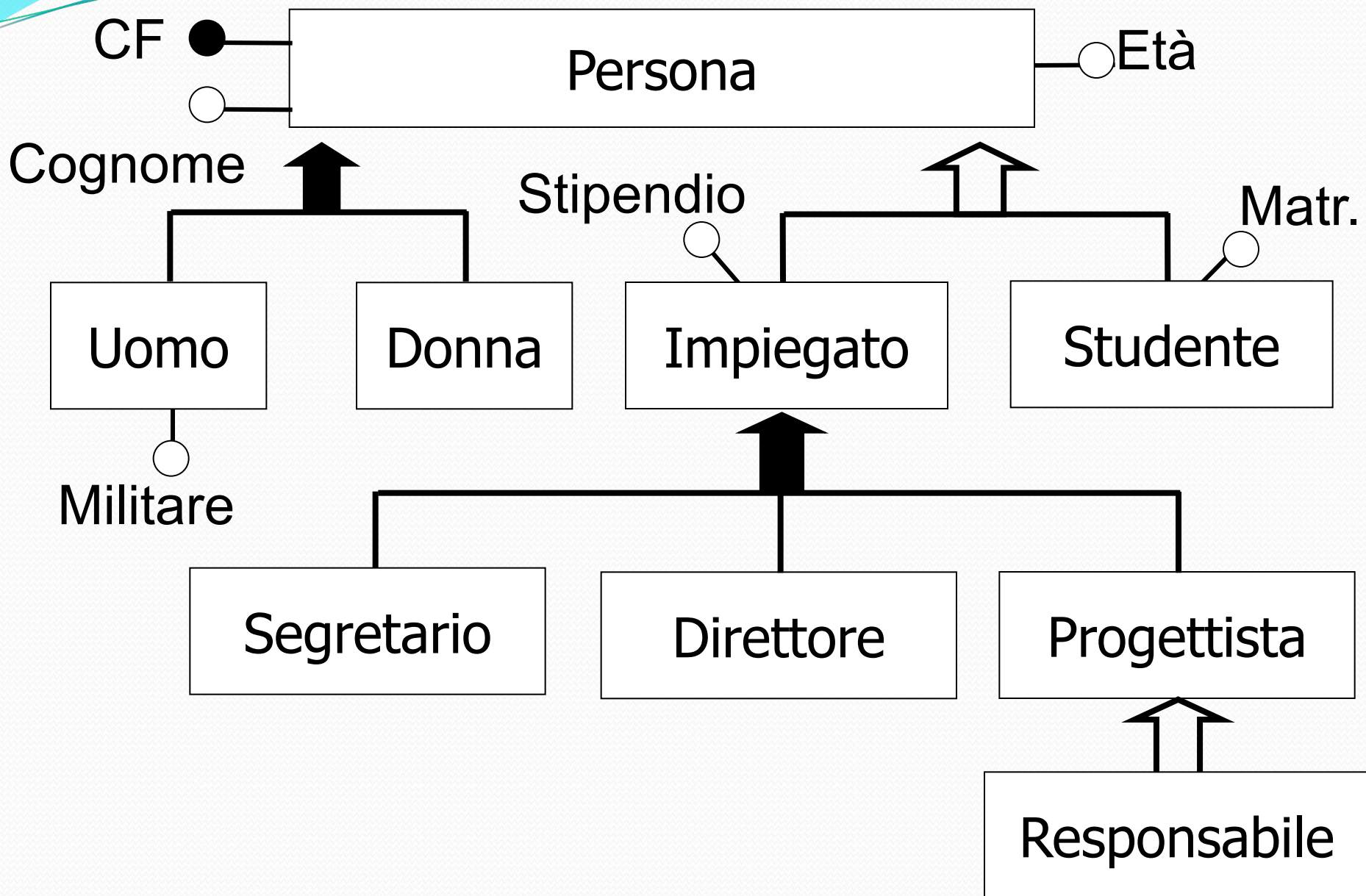
Totale ed esclusiva



Parziale e sovrapposta

Esercizio: Modellare un diagramma EER

- Le persone hanno CF, cognome ed età;
- gli uomini anche la posizione militare;
- gli impiegati hanno lo stipendio e possono essere segretari, direttori o progettisti (un progettista può essere anche responsabile di progetto);
- gli studenti (che non possono essere impiegati) un numero di matricola;
- esistono persone che non sono né impiegati né studenti (ma i dettagli non ci interessano)



Vincoli, con inserimenti e cancellazioni

- L'applicazione dei vincoli comporta delle regole nelle operazioni di inserimento e cancellazione.
 - Cancellare un'entità da una superclasse implica che sia cancellata automaticamente da tutte le sottoclassi.
 - Inserire un'entità in una superclasse implica che l'entità sia obbligatoriamente inserita nelle sottoclassi predicate-defined per cui il predicato è valido.
 - Inserire un'entità in una superclasse di una specializzazione totale implica che l'entità sia inserita obbligatoriamente in almeno una sottoclasse.

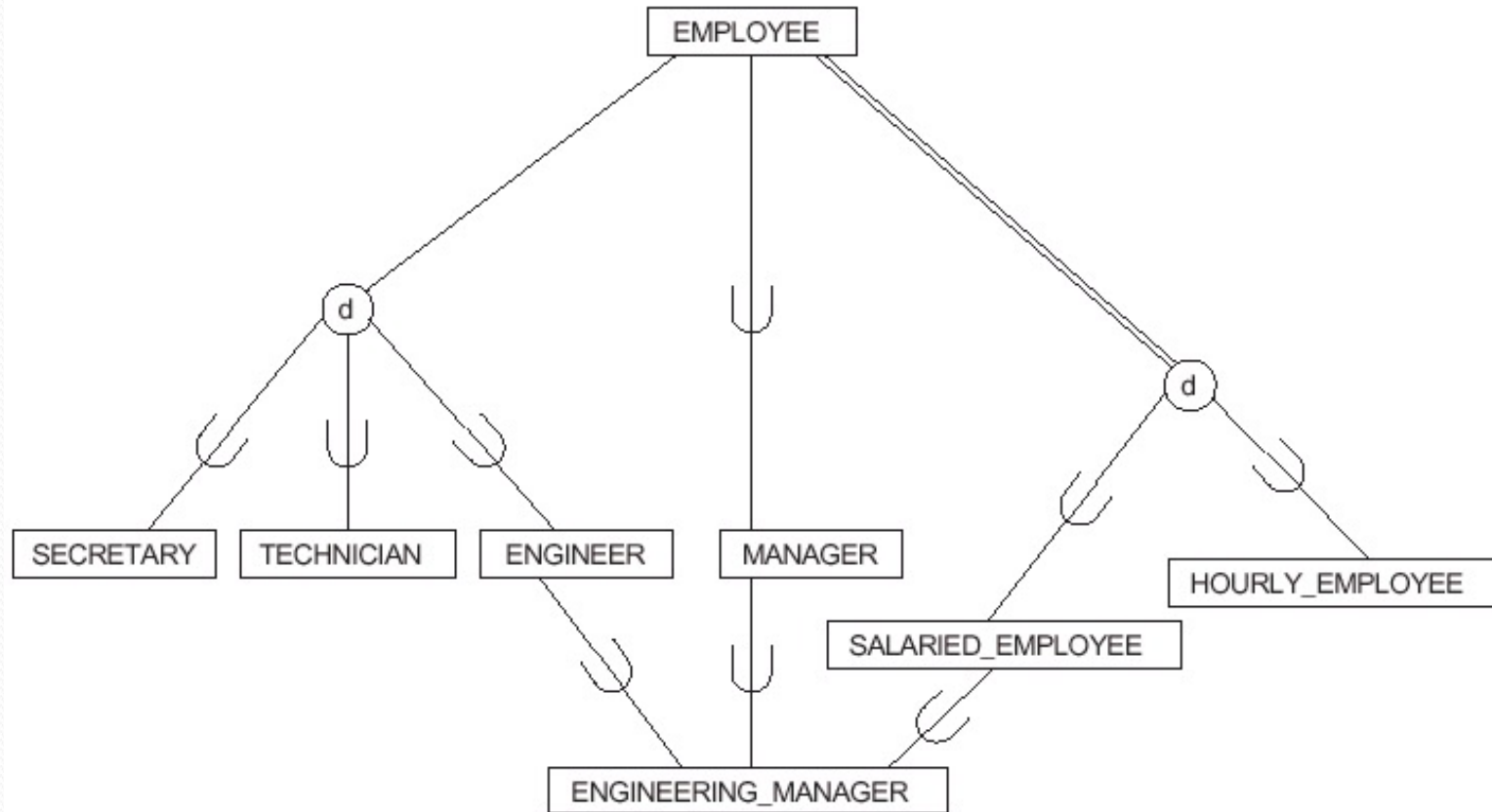
Gerarchie e reticoli di specializzazioni

- Una sottoclasse potrebbe avere a sua volta delle sottoclassi, formanti una gerarchia o un reticolo di specializzazioni.
- Una **gerarchia** di specializzazione ha il vincolo che ogni sottoclasse partecipa, come sottoclasse, solo ad **una** relazione classe/sottoclasse.
- In un **reticolo** di specializzazione, una sottoclasse può partecipare, come sottoclasse, in **più** relazioni classe/sottoclasse.
- Concetti uguali sono applicabili a generalizzazioni, portando a reticoli e gerarchie di generalizzazioni.

Sottoclassi Shared

- Una sottoclasse con più di una superclasse è detta **sottoclasse shared**.
- Ciò porta al concetto di **ereditarietà multipla**, poiché una classe shared eredita gli attributi da più superclassi.
- Non sempre l'ereditarietà multipla è permessa, poiché potenzialmente può provocare problemi nella gestione di conflitti tra nomi di attributi, oltre ad aumentare notevolmente la complessità del sistema.

Gerarchie e reticoli di specializzazioni - *Esempio*



Un reticolo di specializzazioni con una sottoclasse shared:
"Engineering_Manager" è un ingegnere, un manager ed un impiegato a tempo pieno, ereditando attributi da tutte e tre le sottoclassi.

Specializzazione e generalizzazione nel data modeling concettuale

- Nel processo di specializzazione possiamo partire da un tipo di entità, per poi definirne varie sottoclassi mediante specializzazioni.

Tale approccio corrisponde al raffinamento concettuale **top-down**.

- Si può arrivare alla stessa gerarchia (o reticolo) procedendo nella direzione opposta, corrispondente alla sintesi concettuale **bottom-up**.

Specializzazione e generalizzazione nel data modeling concettuale (2)

- In termini strutturali, le gerarchie o i reticoli risultanti dai due processi possono essere uguali, cambiando solo l'ordine in cui sono specificate superclassi e sottoclassi.
- In realtà, però, è difficile seguire esattamente uno dei due approcci.

Il genere si impiega una combinazione dei due processi, inserendo nuove classi nel diagramma non appena queste sono individuate.

Tipi Unione e Categorie

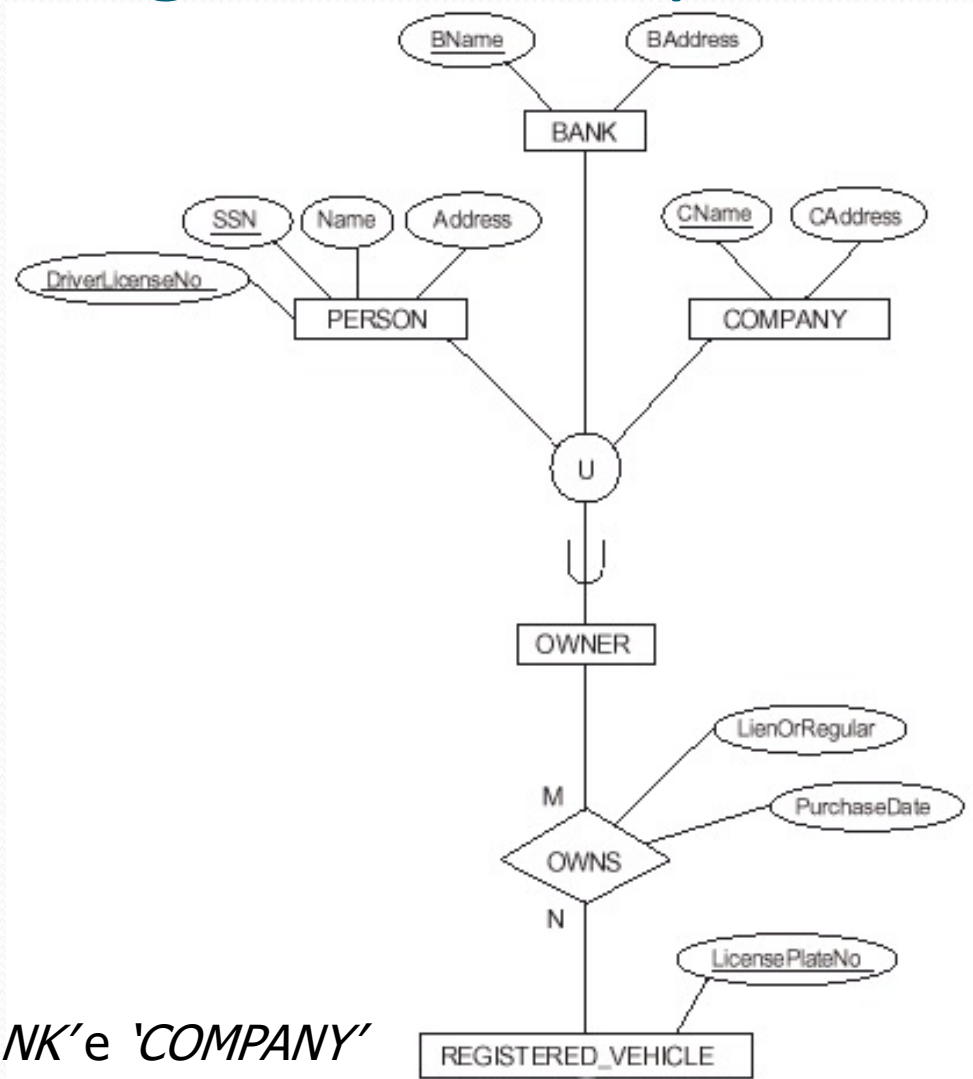


Tipi Unione e Categorie

- A volte può sorgere la necessità di modellare una singola relazione superclasse/sottoclasse, con più di una superclasse.
 - **Esempio:** Consideriamo i tipi di entità *Persona*, *Banca* e *Società*. Ciascuno può essere il possessore di un'auto. Abbiamo bisogno di creare una classe che includa entità dei tre tipi per il ruolo di '*vehicle owner*'.
- La soluzione è creare una sottoclasse che è **l'unione** di più tipi di entità.

Tale sottoclasse è detta **Tipo Unione** o **Categoria**.

Tipi Unione e Categorie - *Esempio*



Union delle entità '*PERSON*', '*BANK*' e '*COMPANY*'

Tipi di relazione con grado maggiore di 2

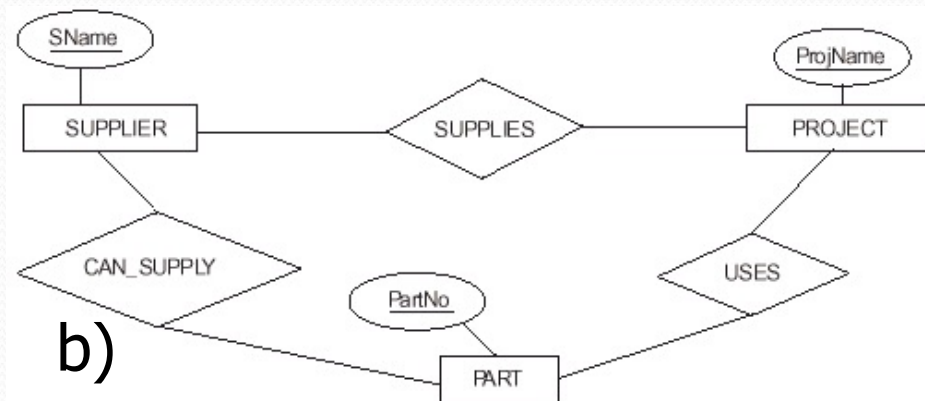
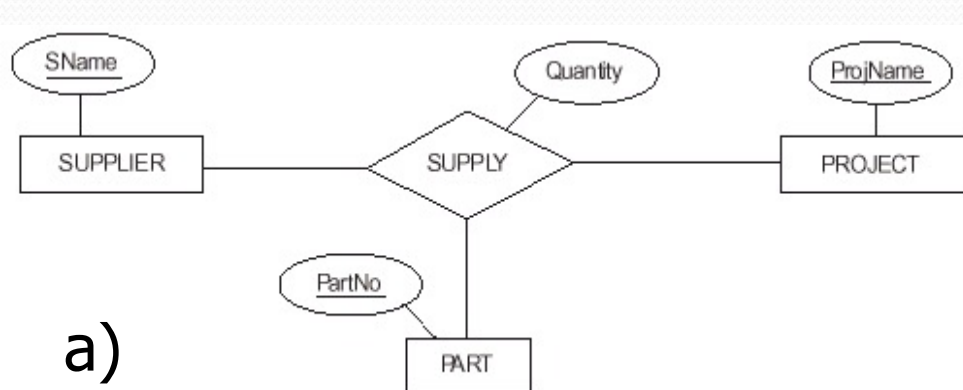


Grado e rappresentazione di tipi di relazioni

- Il grado di un tipo di relazione è il numero dei tipi di entità partecipanti
 - Un tipo di relazione di grado 2 è detto **binario**;
 - Un tipo di relazione di grado 3 è detto **ternario**;
 - ...
- In un diagramma *ER/EER*, un tipo di relazione *R* di grado *n* avrà **n archi**, dove ogni arco congiunge *R* con un tipo di entità partecipante.

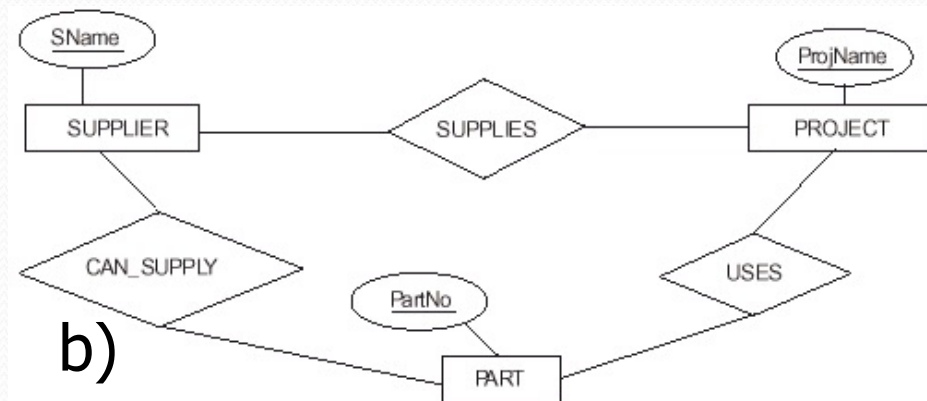
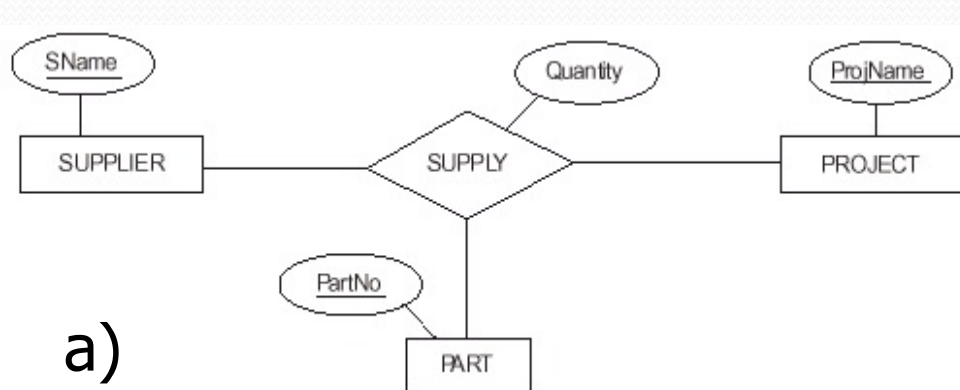
Grado maggiore di 2 - *Esempio*

- Consideriamo la relazione esistente tra *Fornitore*, *Progetto* e *Parte*.
- Tale relazione potrebbe essere modellata come un tipo di relazione ternario (Fig. a) o tre tipi di relazione binari (Fig. b).



Grado maggiore di 2 - *Esempio (2)*

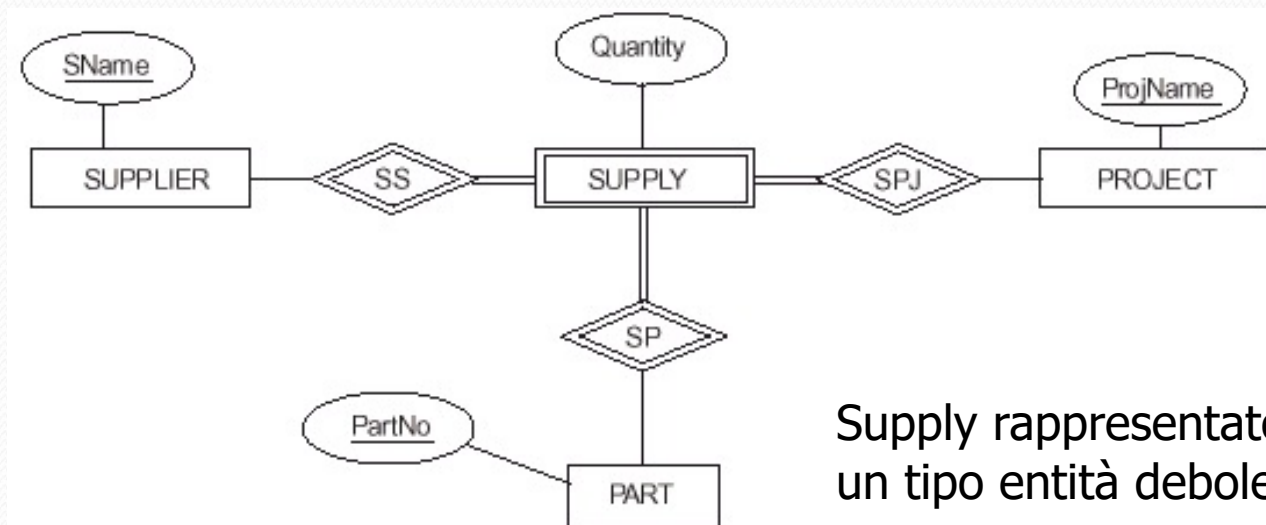
- La relazione ternaria, però, fornisce informazioni maggiori:
 - Supponiamo che *Can_Supply* di Fig. b includa un'istanza (s,p) , che *Uses* includa (j,p) e che *Supplies* includa (s,j) . L'esistenza di queste tre istanze di relazione non implica che esista l'istanza (s,j,p) nella relazione ternaria!
 - (s,p) ogni volta che il fornitore s può fornire parte p (a un progetto qualsiasi);
 - (j,p) ogni volta che un progetto j usa la parte p ;
 - (s,j) ogni volta che il fornitore s fornisce una qualche parte al progetto j .



Grado maggiore di 2 - *Esempio (3)*

- Alcuni tool di supporto alla progettazione di db non permettono la definizione di tipi di relazione con grado maggiore di 2.

In tal caso una relazione ternaria può essere rappresentata con un'entità debole senza chiave parziale, e con più relazioni identificanti.



Supply rappresentato come un tipo entità debole.

Scelta tra binarie ed *n-arie*

- In genere è difficile scegliere quando una relazione debba essere rappresentata con un tipo di relazione *n-ario* e quando invece debba essere divisa in più tipi di relazione di grado minore.
- Il progettista dovrebbe decidere in base alla semantica della situazione da rappresentare.