

## Progettazione logica

**Dario Maio**  
<http://bias.csr.unibo.it/maio/>

Progettazione logica 1

## Il secondo passo...

requisiti del Sistema informativo

progettazione concettuale

SCHEMA CONCETTUALE

progettazione logica

SCHEMA LOGICO

progettazione fisica

SCHEMA FISICO

Progettazione logica 2



## Progettazione logica

- Obiettivo della fase di progettazione logica è pervenire, a partire dallo schema concettuale, a uno schema logico che rappresenti **in modo fedele** i concetti e i requisiti analizzati e che sia, al tempo stesso, **"efficiente"**.
- L'**efficienza** è legata alle **prestazioni**, ma poiché queste non sono valutabili precisamente né a livello concettuale né a livello logico si ricorre all'impiego di **indicatori semplificati**.

Progettazione logica



3



## Progettazione logica "fedele" = equivalenza

- Che cosa s'intende precisamente quando si dice che uno schema relazionale  $DB_{rel}$  rappresenta **"fedelmente"** uno schema concettuale (E/R)  $DB_{conc}$ ?
- Intuitivamente **"fedeltà"** vuol dire che mediante  $DB_{rel}$  possiamo rappresentare esattamente le medesime informazioni documentate con lo schema  $DB_{conc}$  (*possiamo memorizzare gli stessi dati*).
- Più precisamente **"fedeltà"** significa che **i due schemi sono equivalenti dal punto di vista della loro capacità informativa**.
- Il concetto di **capacità informativa** ha diverse definizioni, ma per i nostri scopi può essere considerato equivalente all'**insieme delle istanze legali di uno schema**, indicato con  $Sat(DB)$  e dunque:  
 $DB_{rel}$  e  $DB_{conc}$  sono equivalenti se  $Sat(DB_{conc}) = Sat(DB_{rel})$ .

Progettazione logica



4



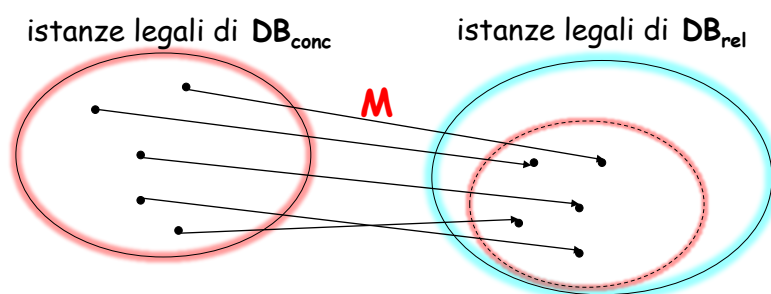
## Progettazione che preserva l'informazione (1)

- Si consideri una progettazione che traduce un dato schema concettuale  $DB_{conc}$  in uno schema logico-relazionale  $DB_{rel}$ .
- Questa attività di progettazione può essere vista, a livello astratto, come la definizione di un **mapping**  $M$  che spiega come trasformare ogni istanza legale  $db_{conc}$  di  $DB_{conc}$  in una corrispondente istanza  $db_{rel}$  di  $DB_{rel}$ .
- La progettazione **preserva l'informazione se  $M$  è totale e iniettiva**:
  - (**totale**) per ogni istanza  $db_{conc}$  di  $DB_{conc}$  esiste un'istanza  $db_{rel}$  di  $DB_{rel}$  tale che  $M(db_{conc}) = db_{rel}$  e
  - (**iniettiva**) non esistono due istanze  $db1_{conc}$  e  $db2_{conc}$  tali che  $M(db1_{conc}) = M(db2_{conc})$ .



## Progettazione che preserva l'informazione (2)

- **Preservare l'informazione**:
  - la definizione intuitivamente asserisce che lo schema relazionale può contenere i dati dello schema E/R (**totalità**) e che si può "ritornare indietro" (**iniettività**).



## Perché ciò non basta

- Si consideri il seguente schema E/R:

e lo schema relazionale:

**Persona**(CF)  
**Auto**(Targa)  
**Proprieta**(CF, Targa, DataAcquisto)  
 FK: CF REFERENCES Persone  
 FK: Targa REFERENCES Auto

- La traduzione preserva l'informazione, ma **esistono infinite istanze che sono legali rispetto a  $DB_{rel}$  e che non lo sono per  $DB_{conc}$ !**

<u>CF</u>	Persona
BLGSTR71B22	
FDLNNR66M45	
BSZNTN82L27	

<u>CF</u>	<u>Targa</u>	DataAcquisto	Proprieta
BLGSTR71B22	CT 001 MJ	12/08/2004	
FDLNNR66M45	CT 001 MJ	15/07/2003	

Progettazione logica 7

## Progettazione che garantisce l'equivalenza

- Diciamo che la progettazione **garantisce l'equivalenza** se:
  - preserva l'informazione e
  - per ogni istanza legale  $db_{rel}$  di  $DB_{rel}$  esiste un'istanza legale  $db_{conc}$  di  $DB_{conc}$  tale che  $M(db_{conc}) = db_{rel}$ .
- La definizione intuitivamente asserisce che esiste una **biiezione** tra gli insiemi di istanze legali.

istanze legali di  $DB_{conc}$       istanze legali di  $DB_{rel}$

Progettazione logica 8



## Come agire in pratica?

- La definizione data di equivalenza non è "operativa", in quanto non dice nulla su come debba essere fatta una traduzione che garantisca l'equivalenza degli schemi.
- Tuttavia può essere usata "localmente":  
in pratica la traduzione da schema E/R a schema relazionale avviene operando una **sequenza di trasformazioni/traduzioni semplici**, per ognuna delle quali è altrettanto semplice rispettare regole che garantiscono l'equivalenza.
- Per quanto visto, possiamo dividere queste regole in:
  - regole che preservano l'informazione (regole sulla "struttura");
  - regole aggiuntive che garantiscono l'equivalenza (regole sui vincoli).
- L'equivalenza può comunque essere solo in parte garantita dal DDL di SQL, infatti alcuni vincoli non possono essere direttamente espressi in SQL.



## Fasi della progettazione logica

- La progettazione logica può essere articolata in due fasi principali:
  - **Ristrutturazione**: eliminazione dallo schema E/R dei costrutti che non possono essere direttamente rappresentati nel modello logico target (**relazionale nel nostro caso**):
    - » eliminazione degli attributi multivalore;
    - » eliminazione delle gerarchie di generalizzazione;
    - » partizionamento/accorpamento di entità e associazioni;
    - » scelta degli identificatori principali.
  - **Traduzione**: si mappano i costrutti residui in elementi del modello relazionale.



## Fase di ristrutturazione

- Si pone l'obiettivo di **semplificare la traduzione** e "ottimizzare" le prestazioni.
- Per confrontare tra loro diverse alternative bisogna conoscere, almeno in maniera approssimativa, il "**carico di lavoro**", ovvero:
  - le principali **operazioni** che la base dati dovrà supportare;
  - i "**volumi**" dei dati in gioco.

*Regola 80-20: il 20% delle operazioni produce l'80% del carico.*

- Gli **indicatori** che deriviamo considerano due aspetti
  - **spazio**: numero di istanze (di entità e associazioni) previste;
  - **tempo**: numero di istanze visitate durante un'operazione.

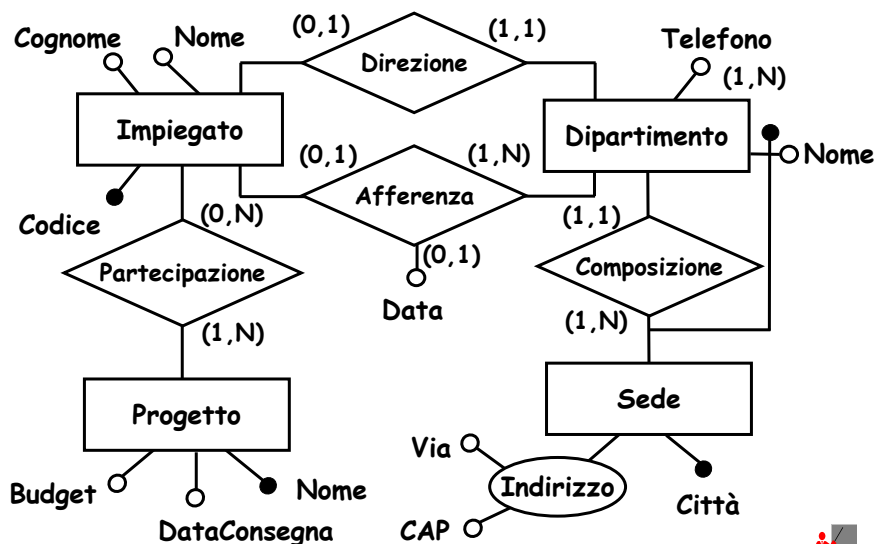
Progettazione logica



11



## Schema di riferimento



Progettazione logica



12



## Tavola dei volumi

- Specifica il numero stimato di istanze per ogni **entità (E)** e **associazione (R)** dello schema.
- I valori sono necessariamente approssimati, ma indicativi.

Concetto	Tipo	Volume
Sede	E	10
Dipartimento	E	80
Impiegato	E	2000
Progetto	E	500
Composizione	R	80
Afferenza	R	1900
Direzione	R	80
Partecipazione	R	6000

Progettazione logica



13



## Descrizione delle operazioni

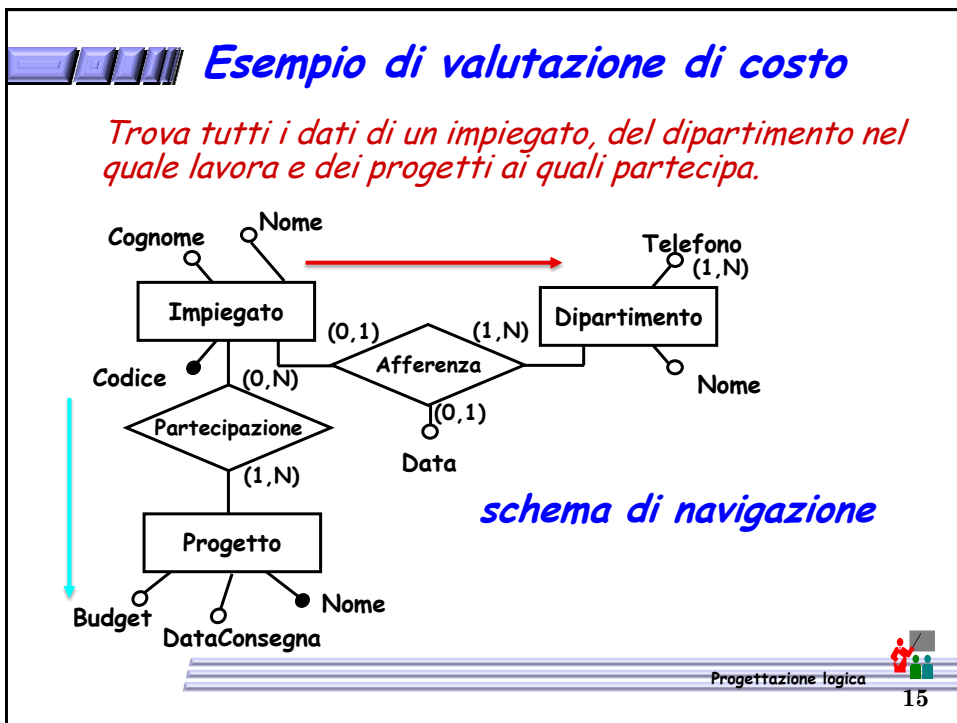
- L'analisi delle operazioni principali richiede la codifica di:
  - **tipo dell'operazione**: **Interattiva (I)** o **Batch (B)**;
  - **frequenza**: numero medio di esecuzioni in un certo periodo di tempo;
  - **schema di navigazione**: frammento dello schema E/R interessato dall'operazione sul quale viene evidenziato (con **freccie**) il "**cammino logico**" da percorrere per accedere alle informazioni di interesse.
- Per ogni operazione si costruisce una **tavola degli accessi** basata sullo schema di navigazione:
  - il campo **costrutto** specifica il tipo di concetto (entità o associazione);
  - nel campo **accessi** si conta il numero degli accessi;
  - il campo **tipo** è riferito al tipo di operazione: le operazioni di **scrittura (S)** sono più onerose di quelle di **lettura (L)**.

*Il peso degli accessi in scrittura è in genere considerato doppio di quello delle letture.*

Progettazione logica



14



### Esempio di tavola degli accessi

- Per ogni entità e per ogni associazione interessate dall'operazione, la tavola degli accessi riporta il **numero di istanze interessate**, e il **tipo di accesso** (L: lettura; S: scrittura)
- Il numero delle istanze si ricava dalla tavola dei volumi mediante semplici operazioni (usualmente assumendo uniformità): *esempio: in media ogni impiegato partecipa a  $6000/2000 = 3$  progetti.*

Concetto	Costrutto	Accessi	Tipo
Impiegato	Entità	1	L
Afferenza	Associazione	1	L
Dipartimento	Entità	1	L
Partecipazione	Associazione	3	L
Progetto	Entità	3	L

Progettazione logica 16





## Analisi delle ridondanze

- Una **ridondanza** in uno schema E-R è un'**informazione significativa** ma **derivabile** da altre.
- In questa fase si decide se eliminare o meno le ridondanze eventualmente presenti; *è quindi comunque importante averle individuate in fase di progettazione concettuale!*
- Se si mantiene una ridondanza
  - si **semplificano** alcune interrogazioni, ma
  - si **appesantiscono** gli aggiornamenti e
  - si occupa **maggior spazio**.
- Le possibili ridondanze riguardano
  - **attributi derivabili** da altri attributi;
  - associazioni derivabili dalla composizione di altre associazioni (**presenza di cicli**).

Progettazione logica

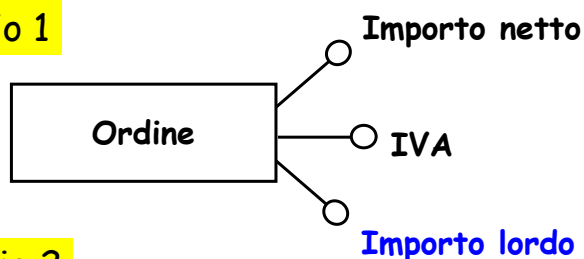


17

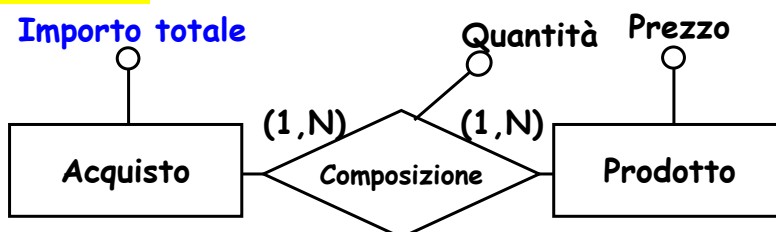


## Attributi derivabili

### Esempio 1



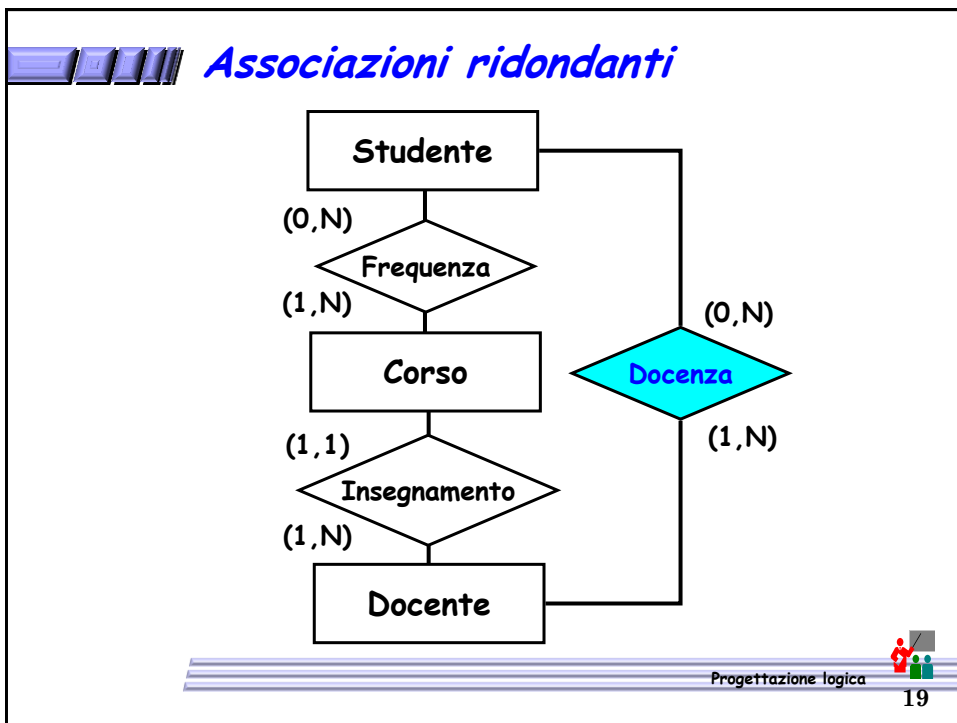
### Esempio 2



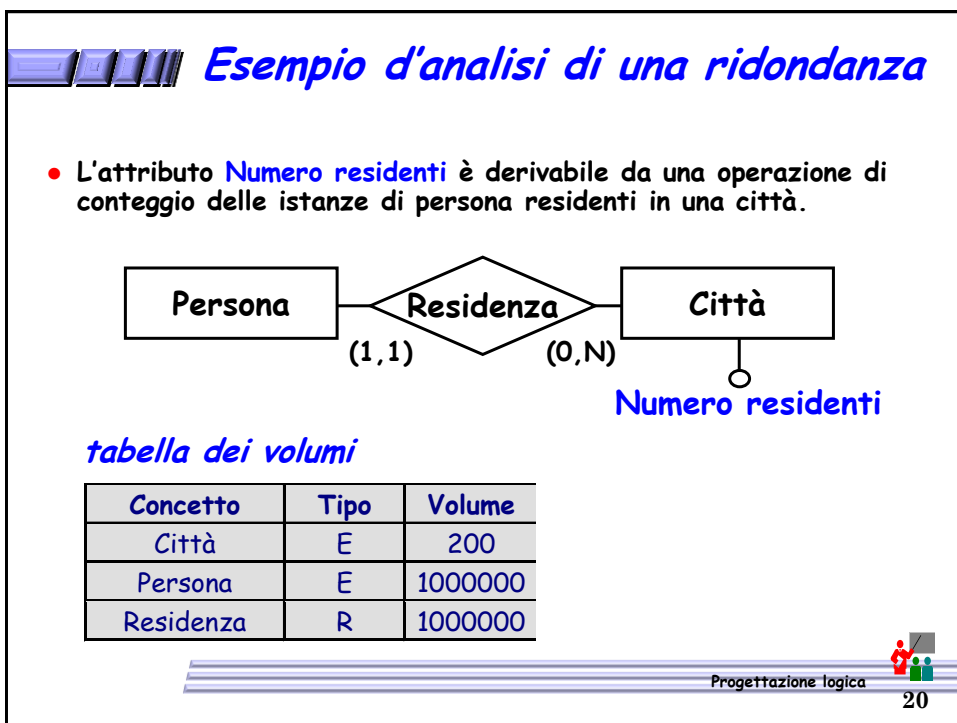
Progettazione logica



18



19



20



## Le operazioni...

- Si considerano innanzitutto le operazioni influenzate dalla ridondanza, considerando anche le loro frequenze di esecuzione:
- **operazione 1**: inserisci una nuova persona con la relativa città di residenza (**500 volte al giorno**);
- **operazione 2**: visualizza tutti i dati di una città (incluso il numero di residenti) (**2 volte al giorno**);
- e si costruiscono le tavole degli accessi.



## ...in presenza di ridondanza...

### Operazione 1

Concetto	Costrutto	Accessi	Tipo
Persona	Entità	1	S
Residenza	Associazione	1	S
Città	Entità	1	L
Città	Entità	1	S

### Operazione 2

Concetto	Costrutto	Accessi	Tipo
Città	Entità	1	L



## *...in assenza di ridondanza*

### Operazione 1

Concetto	Costrutto	Accessi	Tipo
Persona	Entità	1	S
Residenza	Associazione	1	S

### Operazione 2

Concetto	Costrutto	Accessi	Tipo
Città	Entità	1	L
Residenza	Associazione	5000	L

Progettazione logica



23



## *Mantenere o no la ridondanza?*

- È importante considerare la frequenza delle operazioni:
- con ridondanza:
  - operazione 1: 1500 accessi in scrittura e 500 accessi in lettura al giorno;
  - operazione 2: 2 accessi in lettura al giorno;
  - totale: 3502 accessi al giorno;
- senza ridondanza:
  - operazione 1: 1000 accessi in scrittura al giorno;
  - operazione 2: 10002 accessi in lettura al giorno;
  - totale: 12002 accessi al giorno.
- Si decide pertanto di **mantenere la ridondanza, privilegiando l'efficienza.**
- In generale si devono fare anche considerazioni sullo spazio in più richiesto per mantenere la ridondanza.

Progettazione logica



24

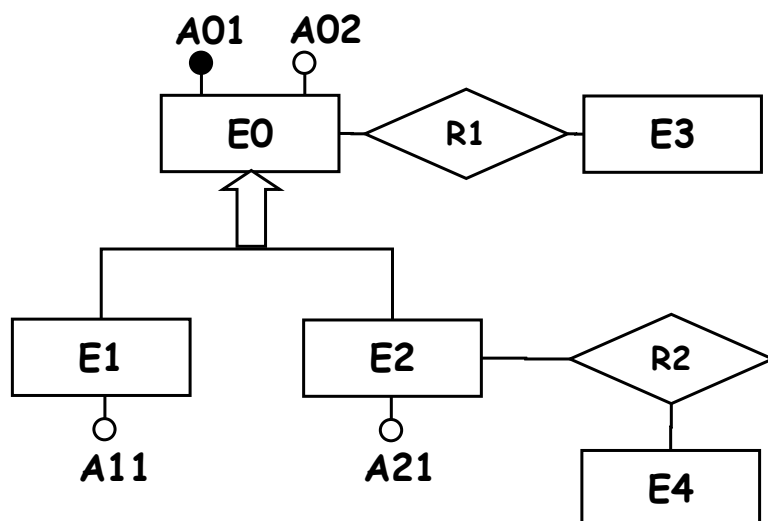


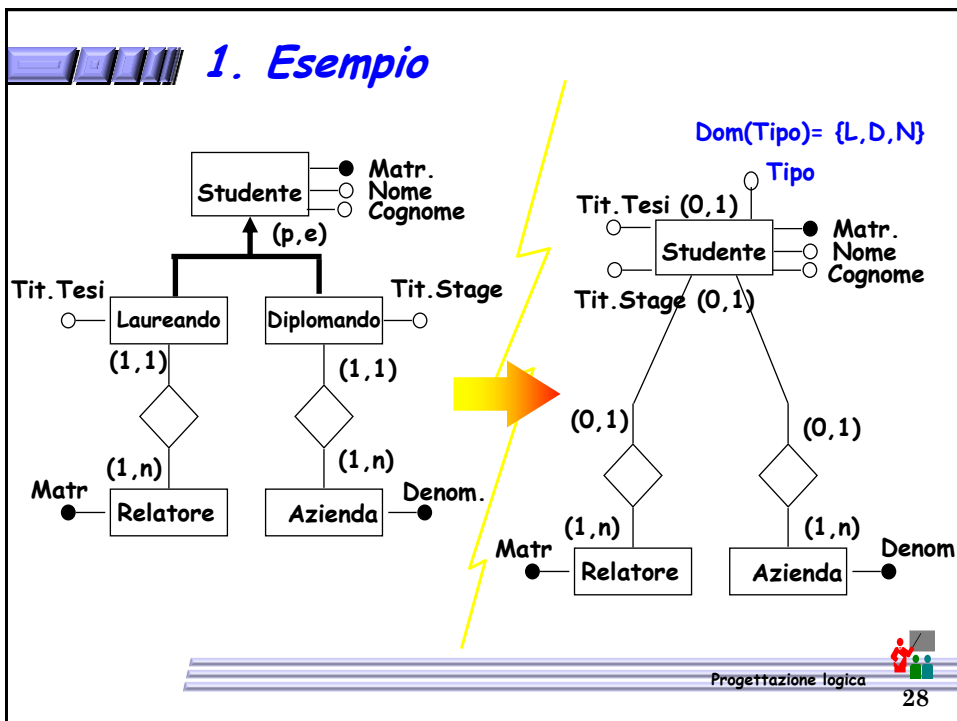
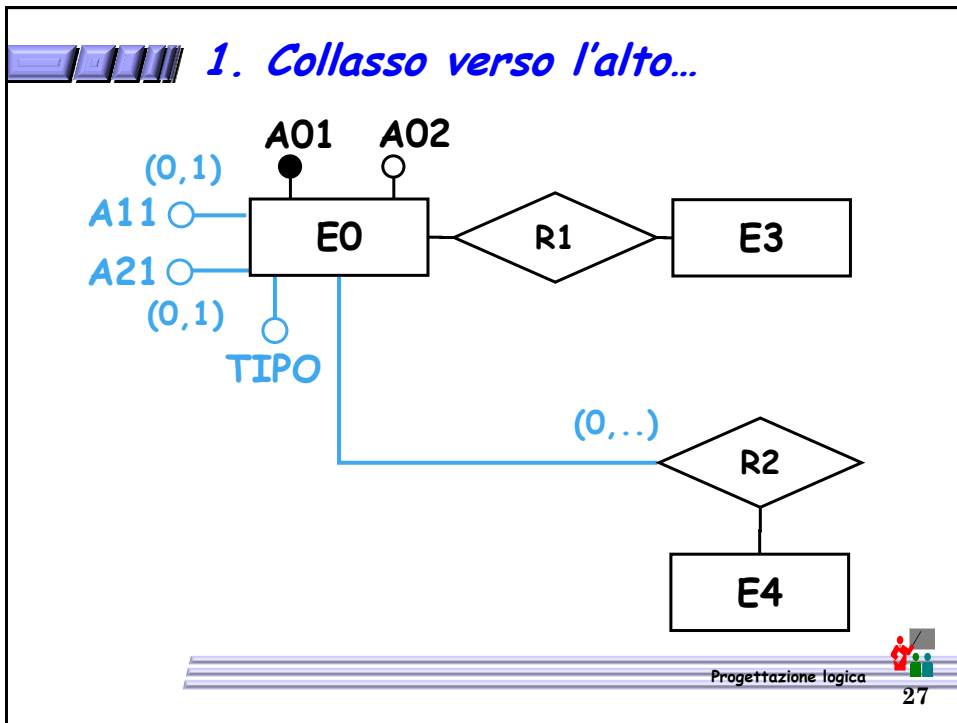
## Eliminazione delle gerarchie


- Il modello relazionale non può rappresentare direttamente le generalizzazioni.
- Entità e associazioni sono invece direttamente rappresentabili.
- Si eliminano perciò le gerarchie, sostituendole con entità e relazioni.
- Vi sono 3 possibilità (più altre soluzioni intermedie):
  - accorpare le entità figlie nel genitore (collasso verso l'alto);
  - accorpare il genitore nelle entità figlie (collasso verso il basso);
  - sostituire la generalizzazione con associazioni.



## Schema di riferimento










## Collasso verso l'alto: osservazioni

- "Tipo"** è un attributo **selettore** che specifica se una singola istanza di E appartiene a una delle N sotto-entità.
- Copertura**
  - totale esclusiva:** Tipo assume N valori, quante sono le sotto-entità;
  - parziale esclusiva:** Tipo assume N+1 valori; il valore in più serve per le istanze che non appartengono a nessuna sotto-entità;
  - sovrapposta:** occorrono tanti selettori quante sono le sotto-entità, ciascuno a valore booleano Tipo\_i, che è vero per ogni istanza di E che appartiene a E\_i; se la copertura è parziale i selettori possono essere tutti falsi, oppure si può aggiungere un selettore.
- Le eventuali associazioni connesse alle sotto-entità si trasportano su E, le eventuali cardinalità minime diventano 0.






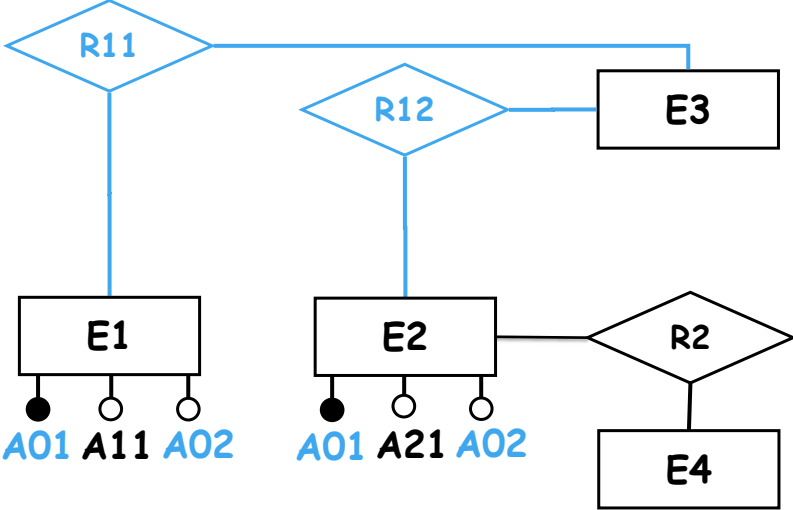
Progettazione logica




29




## 2. Collasso verso il basso...

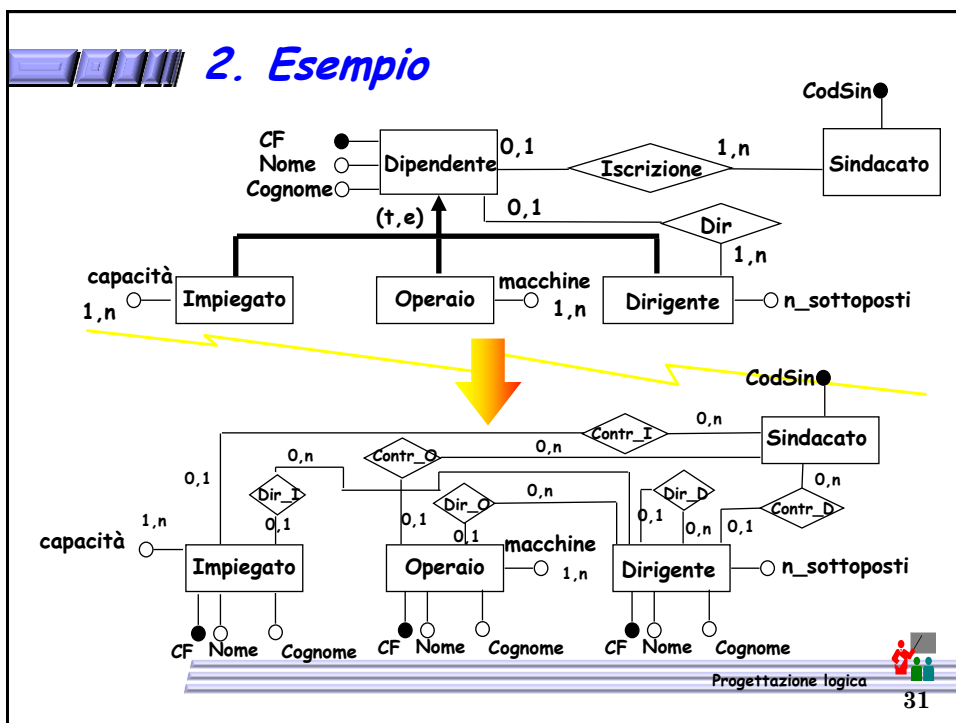




Progettazione logica



30



## Collasso verso il basso: osservazioni

- Se la copertura non è completa il collasso verso il basso non si può applicare:
  - non si saprebbe infatti dove collocare le istanze di E che non sono né in E1, né in E2.
- Se la copertura non è esclusiva introduce ridondanza:
  - una certa istanza può essere sia in E1 sia in E2, e quindi si rappresentano due volte gli attributi che provengono da E.

E1

●  
K

○  
A

○  
A1

E2

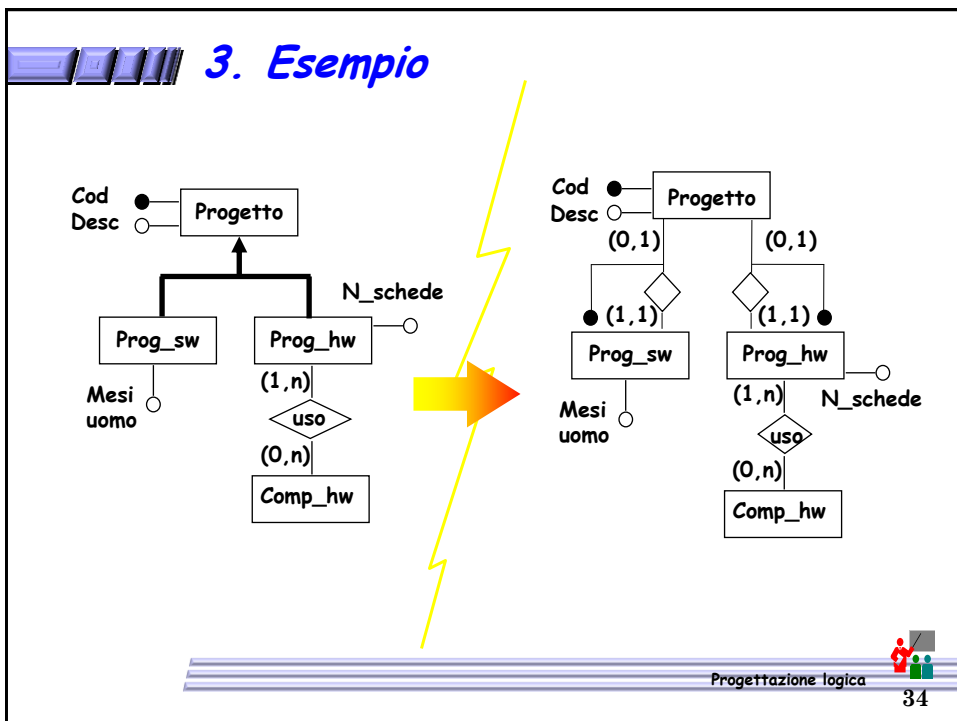
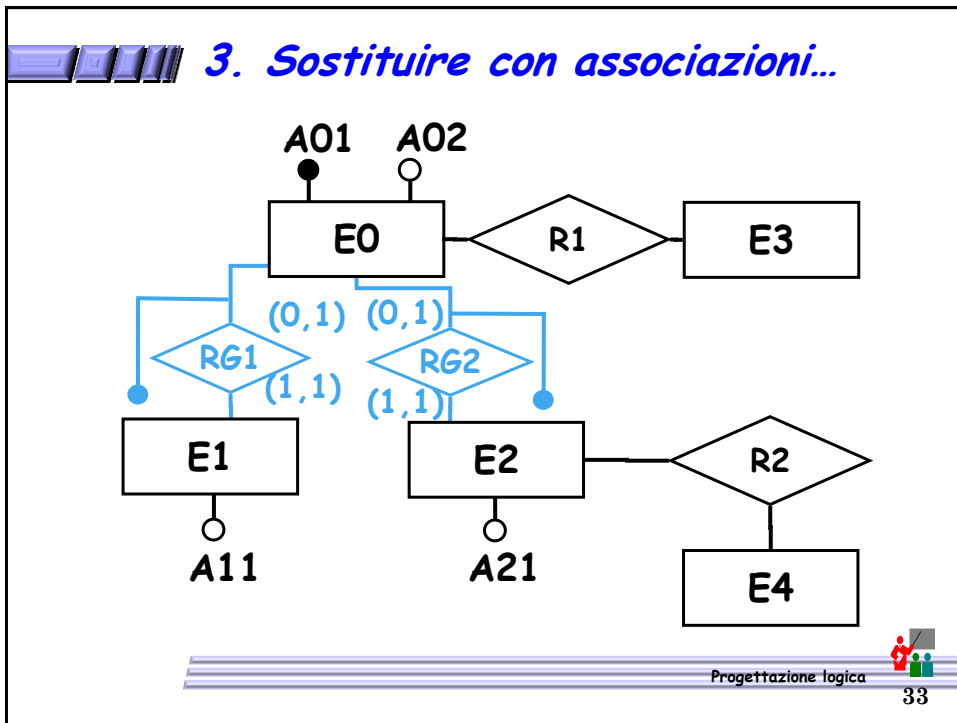
○  
A2

○  
A

●  
K

Progettazione logica 32







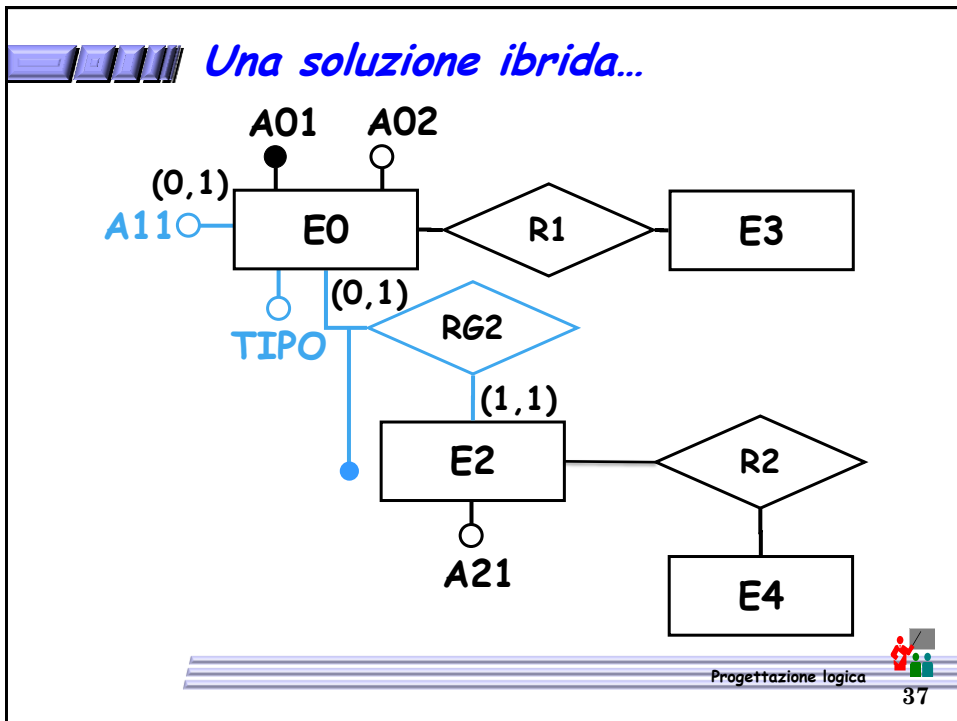
## *Sostituire con associazioni: osservazioni*

- Tutte le entità vengono mantenute: le entità **figlie** sono in associazione binaria con l'entità **padre** e sono **identificate esternamente**.
- La sostituzione con associazioni è sempre possibile indipendentemente dalla copertura della gerarchia.

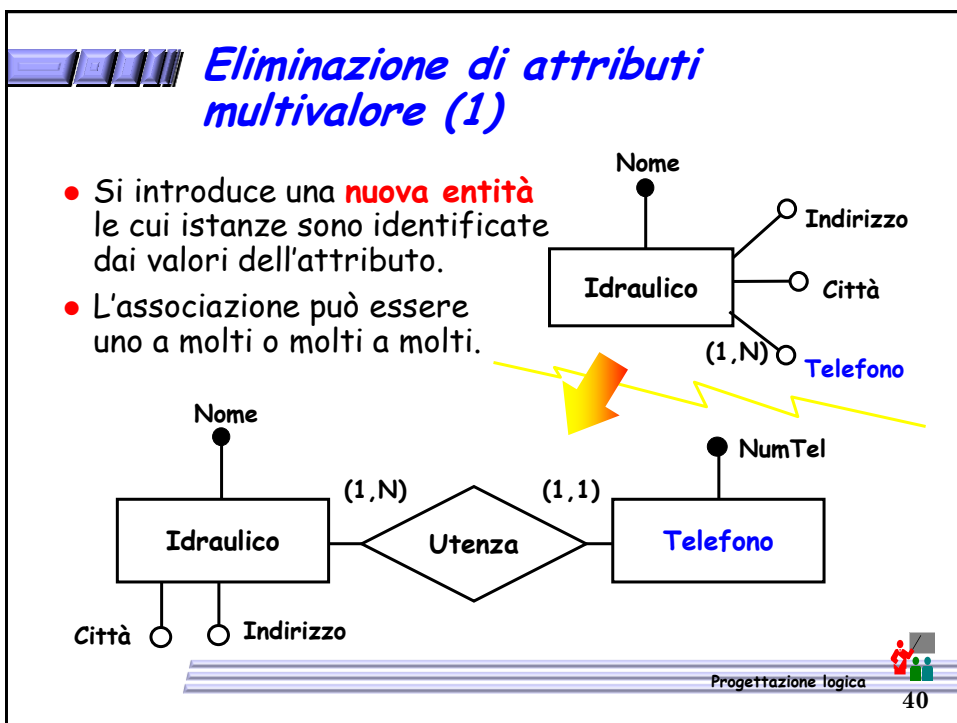
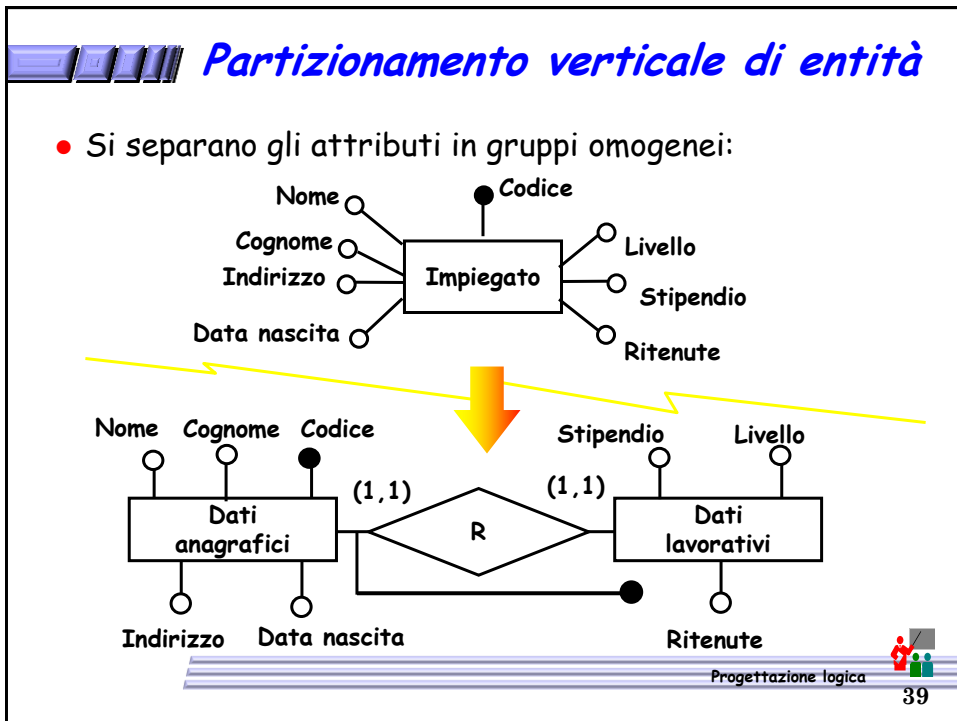


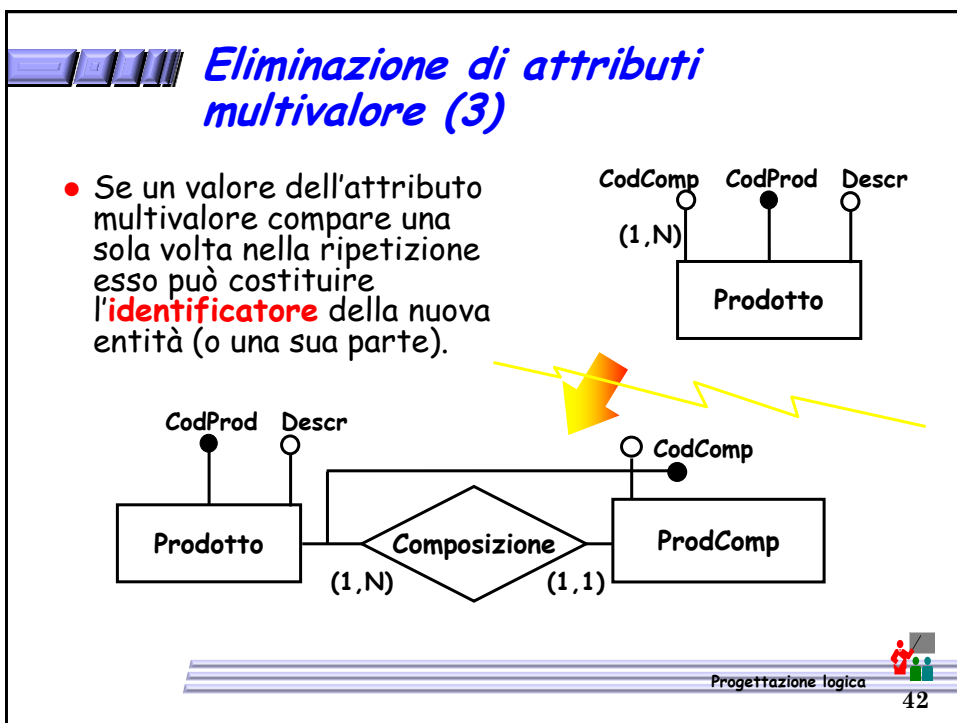
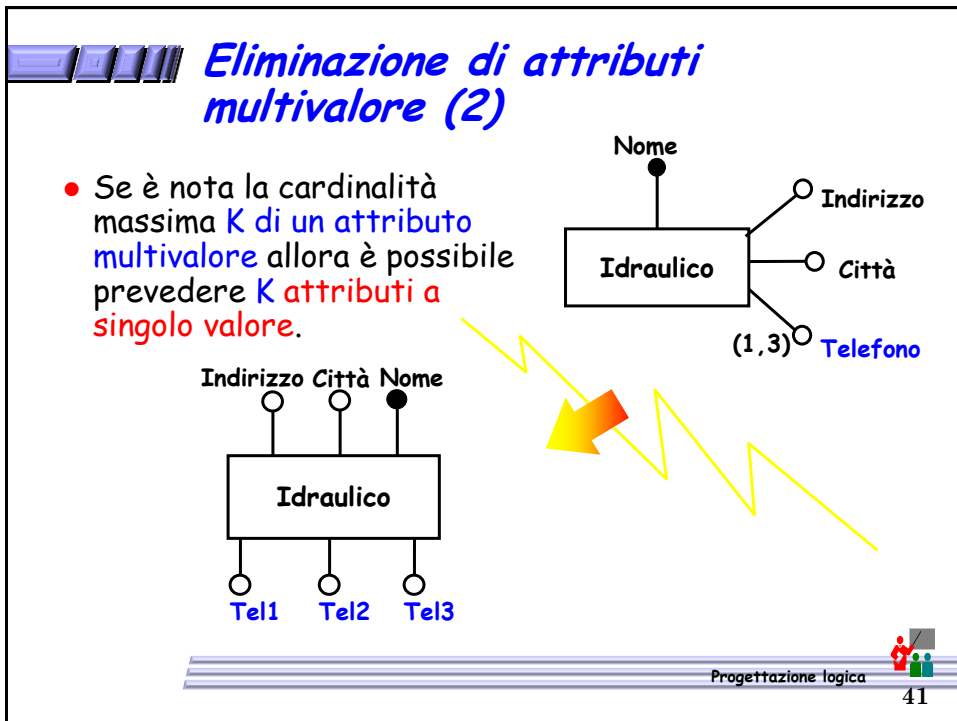
## *Quale alternativa scegliere?*

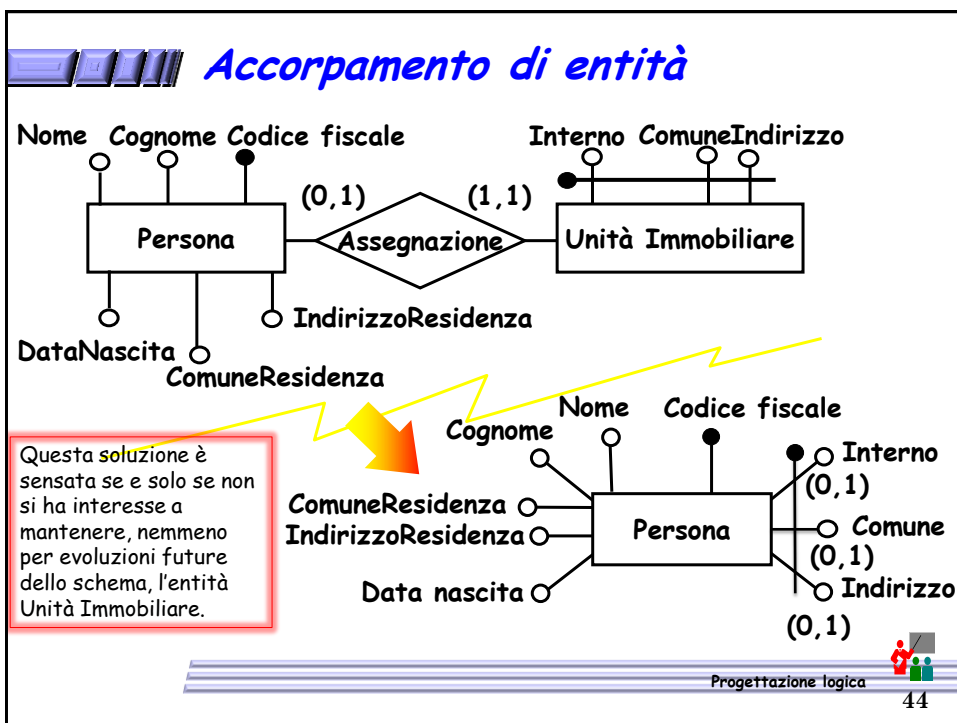
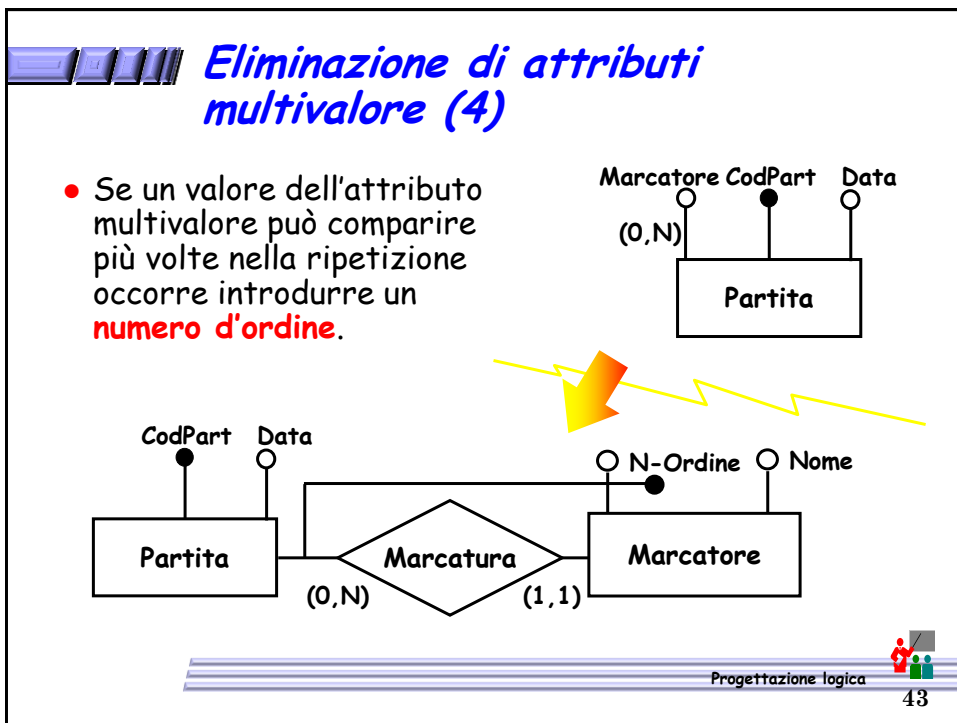
- La scelta fra le alternative illustrate si può fare adottando un **metodo simile a quello visto per l'analisi delle ridondanze**, considerando sia il numero degli accessi sia l'occupazione di spazio.
- È possibile seguire alcune **semplici regole generali** (ovvero: ***mantieni insieme ciò che viene usato insieme***):
  1. conviene se gli accessi all'entità padre e alle entità figlie sono contestuali;
  2. conviene se gli accessi alle entità figlie sono distinti, ma d'altra parte è possibile solo con generalizzazioni totali;
  3. conviene se gli accessi alle entità figlie sono separati dagli accessi al padre.
- Sono anche possibili soluzioni **"ibride"**, soprattutto in presenza di gerarchie a più livelli.

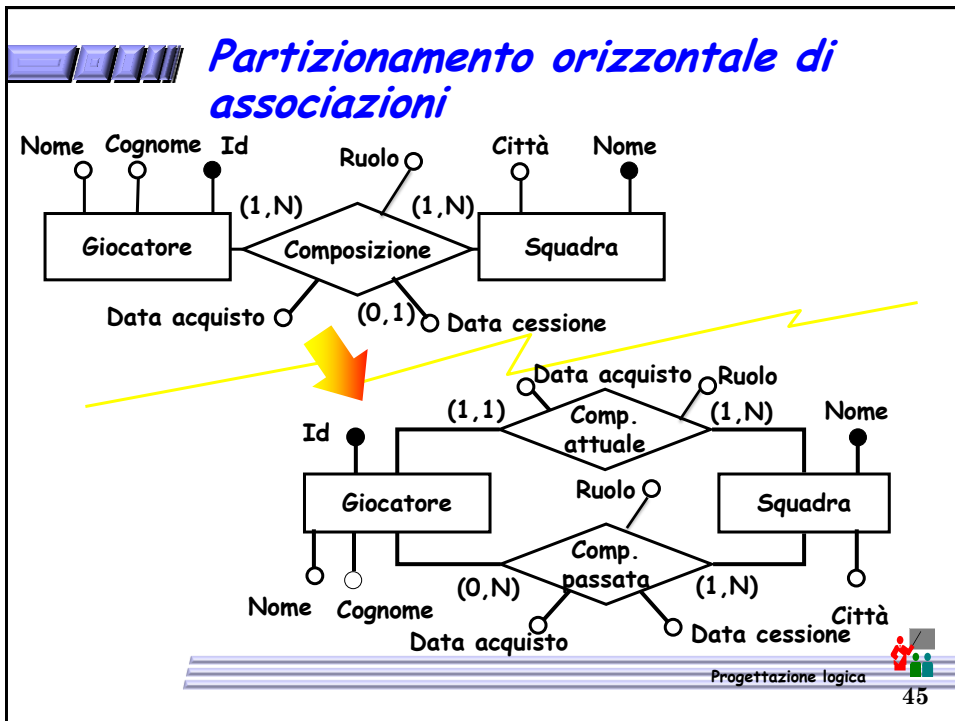


- Partizionamenti e accorpamenti**
- È possibile ristrutturare lo schema accorpando o partizionando entità e associazioni.
  - Tali ristrutturazioni sono effettuate per rendere più efficienti le operazioni in base al principio già visto, ovvero:
    - **gli accessi si riducono:**
      - separando attributi di un concetto che vengono acceduti separatamente;
      - raggruppando attributi di concetti diversi a cui si accede insieme.
    - I casi principali sono:
      - **partizionamento verticale di entità;**
      - **partizionamento orizzontale di associazioni;**
      - **eliminazione di attributi multivalore;**
      - **accorpamenti di entità e associazioni.**
- Progettazione logica
- 38









- ### Scelta degli identificatori principali
- È un'operazione indispensabile per la traduzione nel modello relazionale, e corrisponde alla scelta della **chiave primaria**.
  - I criteri da adottare sono:
    - assenza di opzionalità (valori NULL);
    - semplicità;
    - utilizzo nelle operazioni più frequenti o importanti.
  - Se nessuno degli identificatori soddisfa i requisiti si introducono nuovi attributi (**codici**) ad hoc.
- Progettazione logica 46

## Identificatori principali: esempio

- L'identificatore {Interno, Comune, Indirizzo} è **opzionale**, quindi non può essere scelto come chiave primaria.
- Tra gli attributi **CodiceFiscale** e **CodiceSSN** la scelta dipende da quale fra questi è più **frequentemente** usato per accedere ai dati di una persona.

Progettazione logica 47

## Traduzione delle entità

### Idea di base:

- Ogni entità è tradotta con una relazione con gli stessi attributi.
  - La **chiave primaria** coincide con l'**identificatore principale** dell'entità.
  - Gli **attributi composti** vengono ricorsivamente suddivisi nelle loro componenti, oppure sono mappati in un singolo attributo della relazione, il cui dominio deve essere opportunamente definito.
  - Per brevità, si usa l'asterisco (\*) per indicare la possibilità di **valori nulli**.

**Persona**(CF, Cognome, Nome, Via, NCivico\*, Città, CAP)

Progettazione logica 48





## Traduzione delle associazioni

### Idea di base:

- Ogni associazione è tradotta con una relazione con gli stessi attributi, cui si aggiungono gli identificatori di tutte le entità che essa collega.
  - Gli **identificatori delle entità** collegate costituiscono una **superchiave**.
  - La **chiave** dipende dalle **cardinalità massime** delle entità nell'associazione.
  - Le **cardinalità minime** determinano, a seconda del tipo di traduzione effettuata, la presenza o meno di **valori nulli** (e quindi incidono sui vincoli e sull'occupazione di memoria).

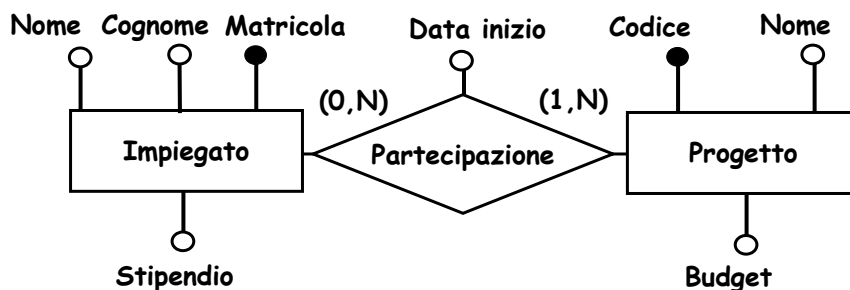
Progettazione logica



49



## Entità e associazione molti a molti



**Impiegato**(Matricola, Nome, Cognome, Stipendio)

**Progetto**(Codice, Nome, Budget)

**Partecipazione**(Matricola, Codice, DataInizio)

FK: Matricola REFERENCES Impiegato

FK: Codice REFERENCES Progetto

Progettazione logica



50

## *Nomi delle foreign key*

- Non è ovviamente necessario mantenere, per gli attributi chiave della relazione che traduce l'associazione, gli stessi nomi delle primary key referenziate, conviene piuttosto far ricorso a nomi più espressivi.

Partecipazione(Impiegato, CodProgetto, DataInizio)

FK: Impiegato REFERENCES Impiegato

FK: CodProgetto REFERENCES Progetto

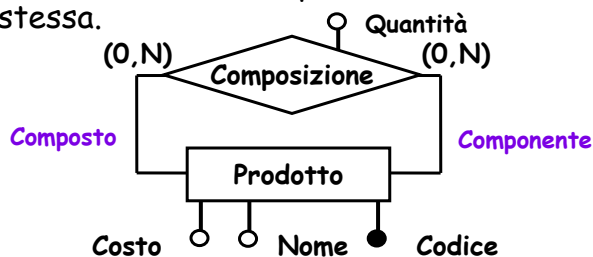
- Ovviamente se le entità collegate hanno un attributo con lo stesso nome la ridenominazione è obbligatoria!

Progettazione logica

51

## *Associazioni ad anello molti a molti*

- In questo caso i nomi degli attributi che formano la chiave primaria della relazione che traduce l'associazione si possono derivare dai **ruoli** presenti nei rami dell'associazione stessa.



Prodotto(Codice, Nome, Costo)

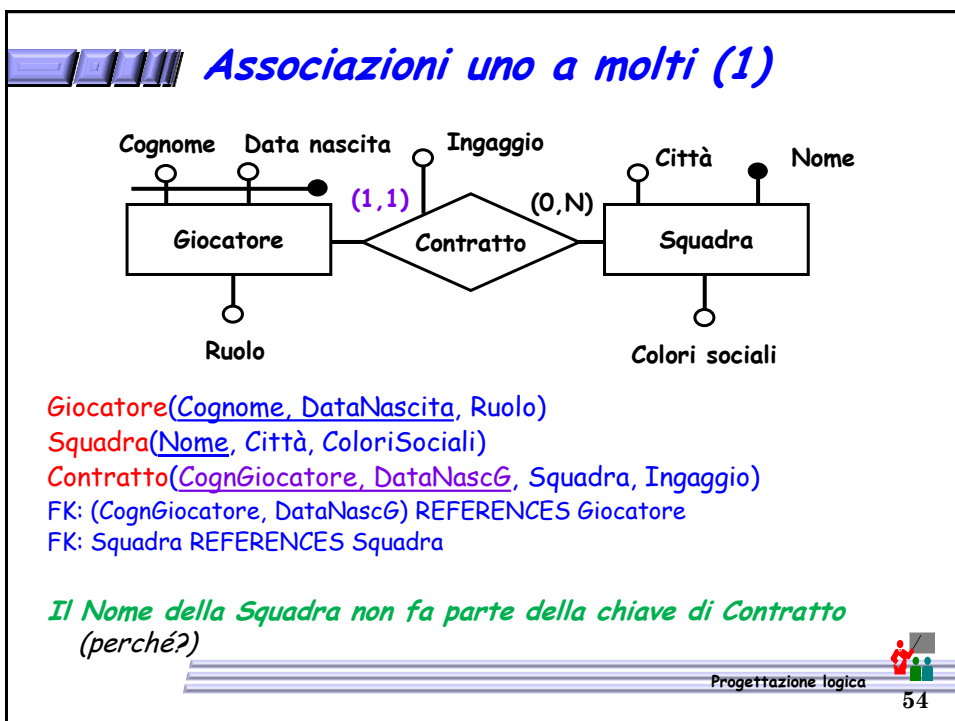
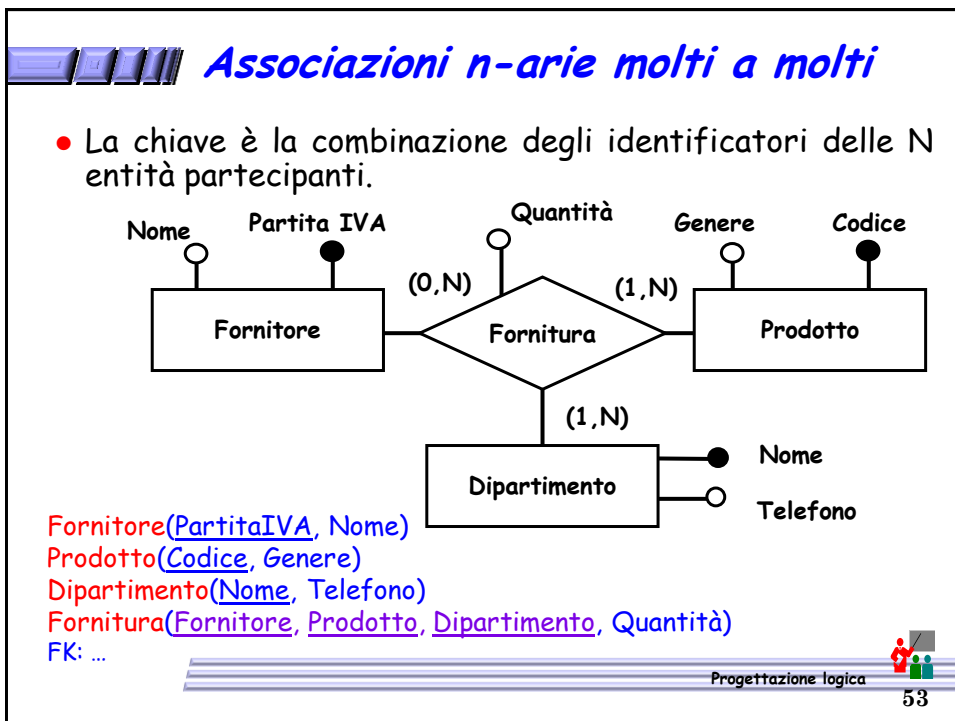
Composizione(Composto, Componente, Quantità)

FK: Composto REFERENCES Prodotto

FK: Componente REFERENCES Prodotto

Progettazione logica

52





## Associazioni uno a molti (2)

- Poiché un giocatore ha un contratto con una sola squadra, nella relazione Contratto un giocatore non può apparire in più tuple.
- Si può pertanto adottare anche una **soluzione più compatta, che fa uso di 2 sole relazioni:**

**Giocatore**(Cognome, DataNasc, Ruolo, Squadra, Ingaggio)

FK: Squadra REFERENCES Squadra

**Squadra**(Nome, Città, ColoriSociali)

che corrisponde a tradurre l'associazione insieme a Giocatore (ovvero all'entità che partecipa con cardinalità massima 1)

- Se fosse  $\text{min-card}(\text{Giocatore}, \text{Contratto}) = 0$ , allora gli attributi **Squadra** e **Ingaggio** dovrebbero entrambi ammettere valore nullo (**e per un giocatore o lo sono entrambi o non lo è nessuno dei due**).

Progettazione logica

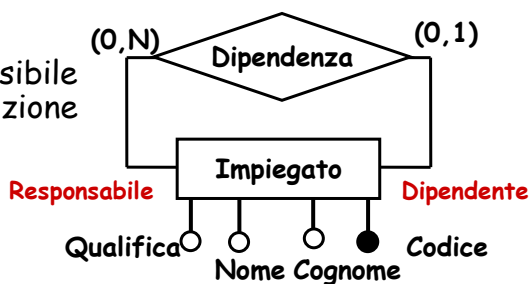


55



## Associazioni ad anello uno a molti

- In questo caso è possibile operare una traduzione con 1 o 2 relazioni.



### 1 relazione:

**Impiegato**(Codice, Nome, Cognome, Qualifica, Responsabile\*)

FK: Responsabile REFERENCES Impiegato

### 2 relazioni:

**Impiegato**(Codice, Nome, Cognome, Qualifica)

**Dipendenza**(Dipendente, Responsabile)

FK: Dipendente REFERENCES Impiegato

FK: Responsabile REFERENCES Impiegato

Progettazione logica



56

## Entità con identificazione esterna

- Nel caso di entità identificata esternamente, si "importa" l'identificatore della/e entità identificante/i.
- L'associazione relativa risulta automaticamente tradotta.

**Studente**(Matricola, Università, Cognome, Nome, AnnoDiCorso)  
 FK: Università REFERENCES Università  
**Università**(Nome, Città, Indirizzo)

Progettazione logica 57

## Identificazioni esterne: una precisazione

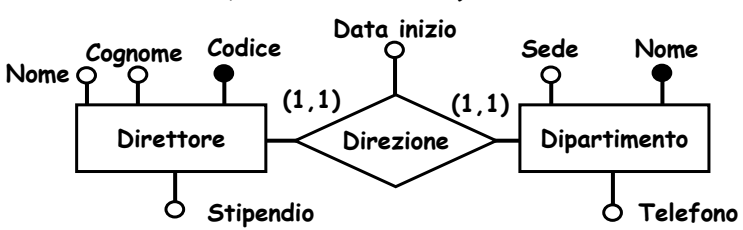
- Nel caso generale, si possono avere **identificazioni esterne in cascata**.
- Per operare correttamente occorre **partire dalle entità non identificate esternamente** e propagare gli identificatori che così si ottengono.

**Università**(Nome, Indirizzo)  
**CorsoDiLaurea**(Università, Codice, Denominazione)  
**Studente**(Università, CodiceCdL, Matricola, Cognome, Nome)

Progettazione logica 58

## Associazioni uno a uno (1)

- Si hanno a disposizione varie possibilità (traduzione con 1, 2 o 3 relazioni)



*L'identificatore di una delle due entità è scelto come chiave primaria, l'altro dà origine a una chiave alternativa.*

*La scelta dipende dall'importanza relativa delle chiavi.*

**Tre relazioni:**


Direttore(Codice, Nome, Cognome, Stipendio)

Dipartimento(Nome, Sede, Telefono)

Direzioe(Direttore, Dipartimento, DataInizio)

FK:...

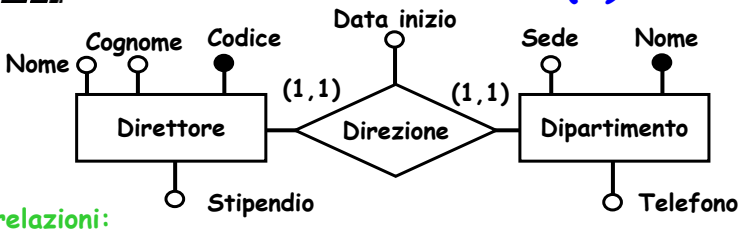
Unique(Dipartimento)



Progettazione logica

59

## Associazioni uno a uno (2)



**Due relazioni:**

Direttore(Codice, Nome, Cognome, Stipendio, Dipartimento, DataInizio)

FK: Dipartimento REFERENCES Dipartimento

Unique(Dipartimento)

Dipartimento(Nome, Sede, Telefono)


oppure

Direttore(Codice, Nome, Cognome, Stipendio)

Dipartimento(Nome, Sede, Telefono, Direttore, DataInizio)

FK: Direttore REFERENCES Direttore

Unique(Direttore)



Progettazione logica

60


### Associazioni uno a uno (3)

**Una relazione:**

**Direttore**(Codice, Nome, Cognome, Stipendio, DataInizio, Dipartimento, Sede, Telefono)  
 Unique(Dipartimento)

oppure

**Dipartimento**(Nome, Sede, Telefono, Direttore, Nome, Cognome, Stipendio, DataInizio)  
 Unique(Direttore)

Progettazione logica  61

### Associazioni uno a uno con opzionalità (1)


- Se  $\text{min-card}(E,R)=0$ , tradurre l'associazione R inglobandola in E non è in generale una buona scelta (dipende dai volumi dei dati in gioco).

**Impiegato**(Codice, Nome, Cognome, Stipendio, Dipartimento\*, DataInizio\*)  
 FK: Dipartimento REFERENCES Dipartimento  
 Unique(Dipartimento)

**TROPPI VALORI NULLI!!**

CHECK (((Dipartimento IS NOT NULL) AND (DataInizio IS NOT NULL)) OR ((Dipartimento IS NULL) AND (DataInizio IS NULL)))

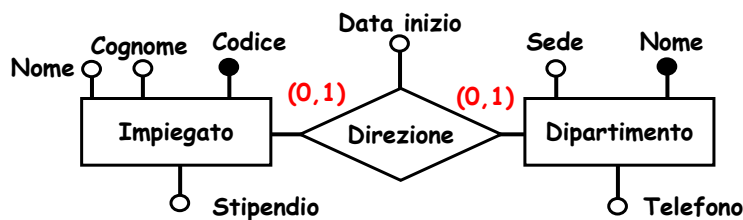
**Dipartimento**(Nome, Sede, Telefono)

Progettazione logica  62



## Associazioni uno a uno con opzionalità (2)

- La traduzione con una sola relazione corrisponde a un accorpamento di entità:
  - Se  $\text{min-card}(E1,R) = \text{min-card}(E2,R) = 1$  si avranno due chiavi, entrambe senza valori nulli (la chiave primaria è "la più importante");
  - Se  $\text{min-card}(E1,R) = 0$  e  $\text{min-card}(E2,R) = 1$  la chiave derivante da E2 ammetterà valori nulli, e la chiave primaria si ottiene da E1;
  - Se  $\text{min-card}(E1,R) = \text{min-card}(E2,R) = 0$  entrambe le chiavi hanno valori nulli, quindi si rende necessario introdurre un codice.



ImpDip(CodiceImpDip, CodiceImp\*, ..., Dipartimento\*, ..., DataInizio\*)

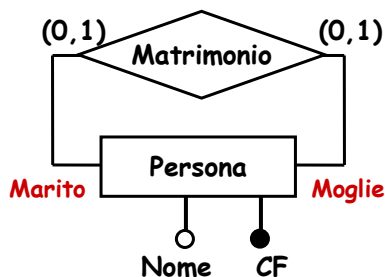
Progettazione logica

63



## Associazioni ad anello uno a uno

- In questo caso è possibile operare una traduzione con una o due relazioni
- La traduzione con una relazione è ancora problematica se entrambe le partecipazioni sono opzionali



Una relazione:

Persona(Codice, CFUomo\*, NomeUomo\*, CFDonna\*, NomeDonna\*)

Due relazioni:

Persona(CF, Nome)

Matrimonio(Marito, Moglie)

FK: Marito REFERENCES Persona

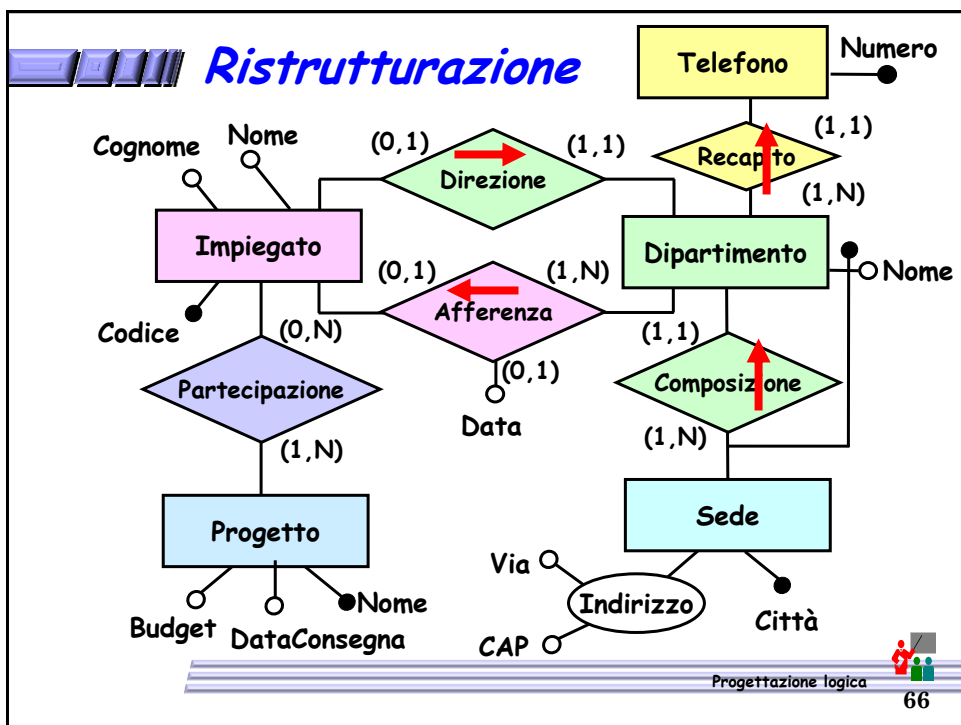
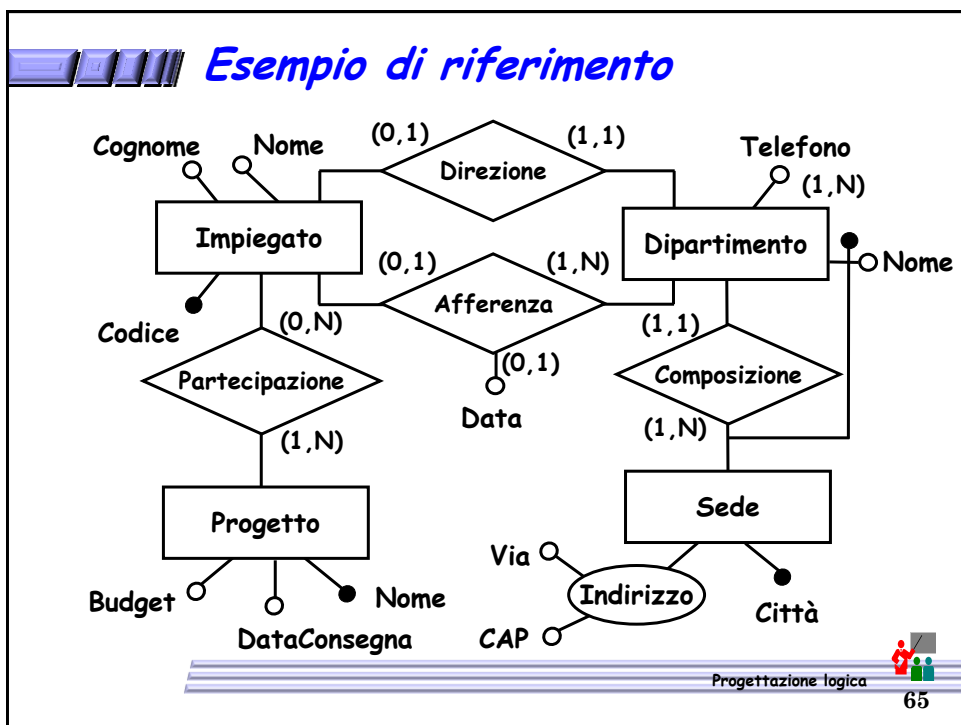
FK: Moglie REFERENCES Persona

Unique (Moglie)

Progettazione logica

64







## Schema logico relazionale

- Per le entità E che partecipano ad associazioni sempre con  $\text{max-card}(E,R) = n$  la traduzione è immediata:  
*Sede*(Città, Via, CAP)  
*Progetto*(Nome, Budget, DataConsegna)
- Anche l'associazione Partecipazione si traduce immediatamente:  
*Partecipazione*(Impiegato, *Progetto*)
- L'entità Dipartimento si traduce importando l'identificatore di Sede e inglobando l'associazione Direzione:  
*Dipartimento*(Nome, Città, Direttore)
- L'entità Telefono si traduce con una relazione che ingloba l'associazione Recapito  
*Telefono*(Numero, Nome, Città)
- Per tradurre l'associazione Afferenza, assumendo che siano pochi gli impiegati che non afferiscono a nessun dipartimento, si opta per una rappresentazione compatta  
*Impiegato*(Codice, Nome, Cognome, Dipartimento\*, Data\*)

Progettazione logica



67



## Osservazioni finali

- La progettazione logica, pur potendosi avvalere di strumenti CASE, non deve essere condotta "alla cieca"; in presenza di diverse alternative occorre valutare diversi fattori, tra cui:
  - la presenza o meno di valori nulli, e la loro incidenza, che dipende dal **volume dei dati**;
  - le porzioni di schema E/R interessate dalle varie **operazioni** (con particolare riferimento ai join tra le relazioni che vengono create);
  - la flessibilità degli schemi relazionali rispetto ad evoluzioni future.
- I casi visti (**semplici esempi a scopo didattico**) non esauriscono certamente l'argomento e lasciano sempre spazio per soluzioni specifiche "ad hoc".
- Ad esempio, associazioni uno a molti con  $\text{max-card}(E2,R) = K$ , con K "piccolo", possono al limite essere tradotte con 1 sola relazione, prevedendo K repliche degli attributi di E2 (es. tipico: numeri di telefono).

Progettazione logica



68



## Sommario:

- ✦ La **fase di progettazione logica** ha lo scopo di derivare uno schema logico che rispetti quanto più possibile i concetti espressi nello schema E/R di partenza e che sia al tempo spesso "efficiente".
- ✦ I confronti tra le diverse alternative sono eseguiti considerando le **principali operazioni** interessate e i **volumi dei dati** in gioco.
- ✦ La fase di **ristrutturazione** elimina dallo schema E/R tutti i costrutti che non possono essere direttamente rappresentati nel modello logico, e apporta modifiche strutturali sulla base di considerazioni di efficienza.
- ✦ La fase di **traduzione** opera traducendo entità e associazioni.
- ✦ Le diverse alternative che si hanno a disposizione per tradurre le associazioni dipendono dalle **cardinalità massime** in gioco, le quali determinano anche le chiavi delle relazioni che si ottengono.
- ✦ Le **cardinalità minime** possono portare, in funzione della traduzione scelta, ad avere valori nulli.

