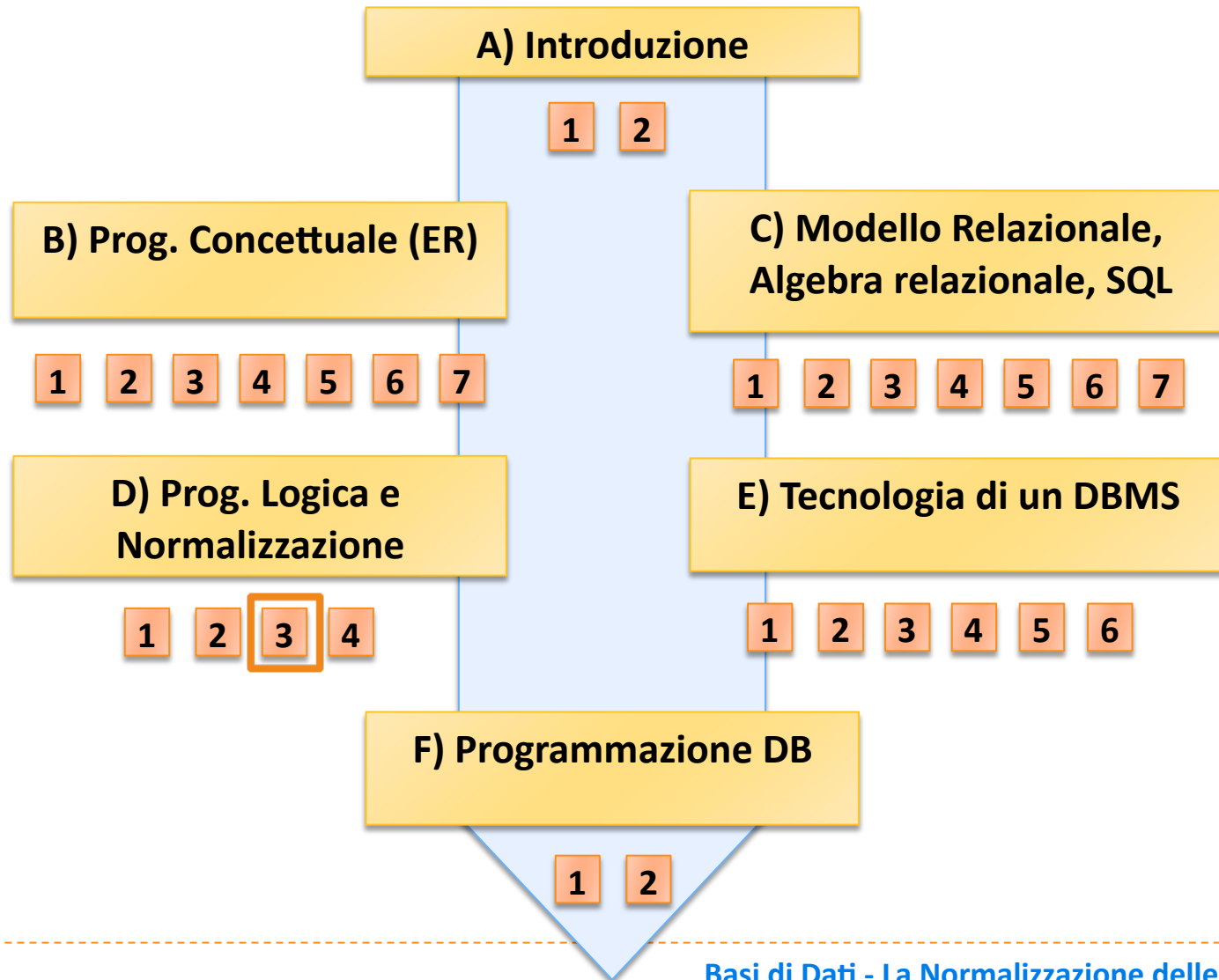


Basi di Dati

La Normalizzazione delle Relazioni

Basi di Dati – Dove ci troviamo?



Nelle lezioni precedenti

- ▶ Abbiamo visto la conversione degli schemi **E/R** in schemi logici relazionali
- ▶ questa attività, che va sotto il nome di **progetto logico**, prevede una serie di fasi che applicano regole di **trasformazione** e **traduzione**

In questa lezione

- ▶ continueremo a risolvere i problemi legati alla costruzione di **schemi relazionali**, vedremo, in particolare, come produrre schemi relazionali **esenti da anomalie** e non suscettibili di **perdita di informazioni** nelle operazioni di join
- ▶ **riprenderemo il concetto di dipendenza funzionale**
- ▶ **introdurremo il concetto di forma normale**

Il caso in esame

impiegato	stipendio	progetto	budget	funzione
Rossi	2	biella	300	tecnico
Verdi	3	valvola	500	progettista
Verdi	3	albero	1500	progettista
Neri	7	albero	1500	direttore
Neri	7	valvola	500	consulente
Neri	7	biella	300	consulente
Mori	6	biella	300	direttore
Mori	6	albero	1500	progettista
Bianchi	6	albero	1500	progettista
Bianchi	6	biella	300	progettista

Ridondanze e anomalie

1) ridondanza :

- ▶ **si ripete più volte** la notizia che un impiegato percepisce un certo stipendio
- ▶ **si ripete più volte** che un progetto ha un certo budget
- ▶ i valori di **progetto** e di **impiegato** si ripetono e quindi non possono singolarmente essere presi come chiave
- ▶ la chiave è **(progetto, impiegato)** : non si hanno ripetizioni

Ridondanze e anomalie

2) aggiornamento :

- ▶ **poiché si ripete più volte** la notizia che un impiegato percepisce un certo stipendio, se lo stipendio viene aggiornato questo deve essere fatto su **tutte le tuple** che riguardano un certo impiegato
- ▶ **poiché si ripete più volte** che un progetto ha un certo budget, se il budget viene aggiornato lo si deve fare su **tutte le tuple** che riguardano un certo progetto

Ridondanze e anomalie

3) cancellazione :

- ▶ supponendo che un impiegato lasci l'azienda o non partecipi a progetti rischiamo di perdere i dati sui progetti se era l'ultimo impiegato del progetto
- ▶ analogamente per i dati degli impiegati se un progetto viene eliminato
- ▶ se la **chiave** è **(progetto, impiegato)** in entrambi i casi di eliminazione si potrebbero avere **valori nulli** nella chiave

Ridondanze e anomalie

4) inserimento :

- ▶ se la **chiave** è **(progetto, impiegato)** non è possibile inserire i dati di un impiegato se non è stato assegnato ad almeno un progetto, analogamente per un nuovo progetto a cui non è stato ancora assegnato un impiegato
- ▶ accettare un inserimento di **(progetto)** o, **(impiegato)** vuol dire che si inseriscono valori nulli (incompatibili con la chiave)

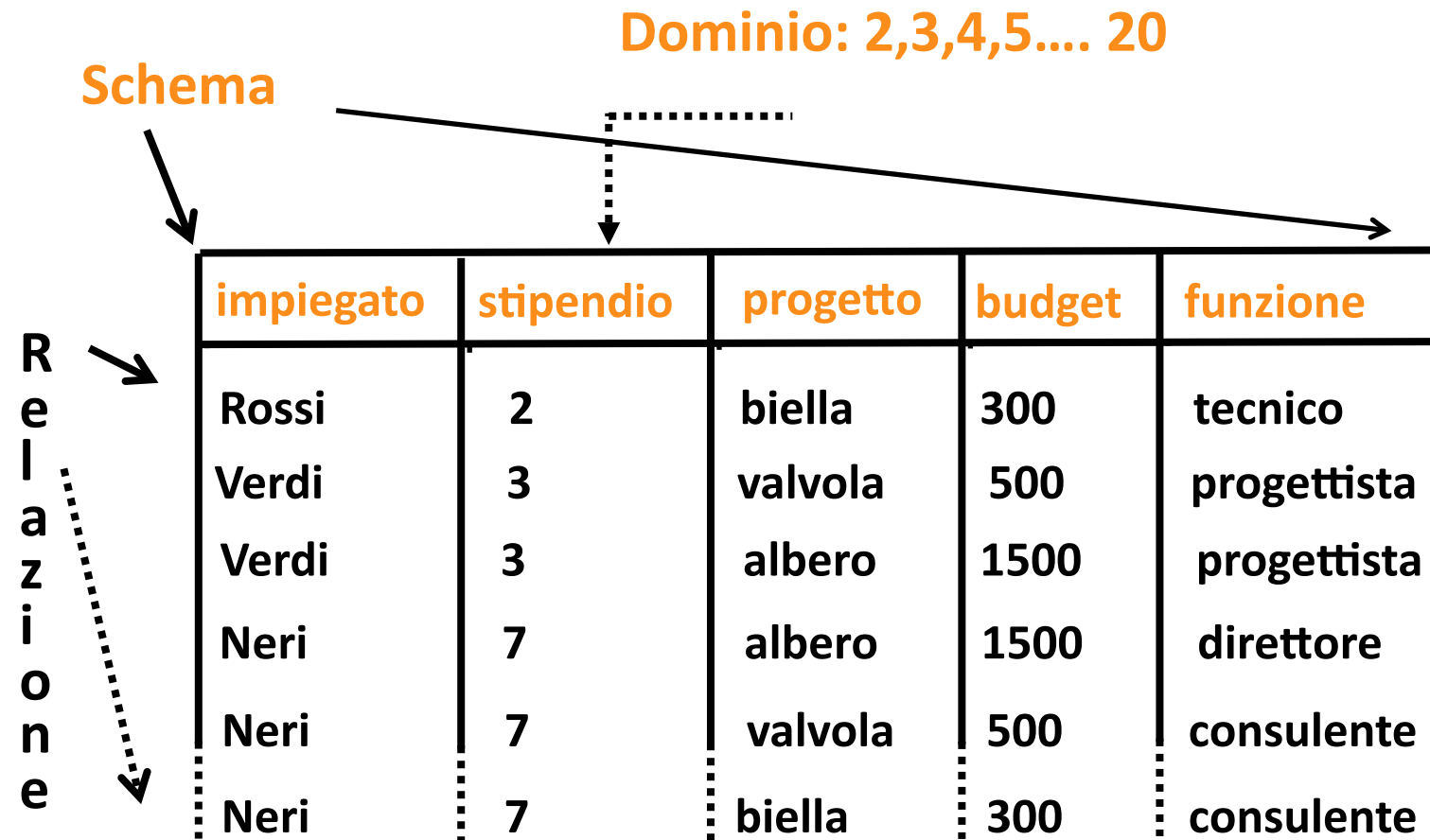
Ridondanze e anomalie

- ▶ casi così **eclatanti** non succedono se si è seguita la **prassi corretta di progettazione**: prima lo schema E/R e poi la traduzione in schema relazionale
- ▶ può però succedere che **carenze** di specifiche o **errori** di schematizzazione possano portare a relazioni con **anomalie**
- ▶ i casi sono invece più frequenti quando si esaminano **vecchi DB** scarsamente documentati o, addirittura, si cerca di intuire la natura dei dati da documenti che sintetizzano le informazioni su **moduli cartacei**

Dipendenze funzionali

- ▶ La **dipendenza funzionale** è un **vincolo di integrità** per il modello relazionale
- ▶ dall'osservazione della relazione ricaviamo che:
 - ▶ ogni volta che in una tupla compare un certo impiegato lo stipendio è **sempre lo stesso**
 - ▶ possiamo dire che il valore dell'impiegato **determina** il valore dello stipendio, cioè:
 - ▶ esiste una **funzione che associa** ad ogni valore nel dominio impiegato **uno ed un solo** valore nel dominio stipendio
 - ▶ analogamente per un valore di progetto

Ricordiamo che:



Dipendenze funzionali

- ▶ La **dipendenza funzionale** si può definire formalmente :
 - ▶ data una relazione **R** definita su uno schema **S(X)** e due sottoinsiemi di attributi **Y** e **Z** non vuoti di **X**, esiste una dipendenza funzionale **$Y \rightarrow Z$** , se, per ogni coppia di tuple **t1** e **t2** aventi lo stesso valore di **Y** risulta che hanno lo stesso valore di **Z**
- ▶ dall'osservazione della relazione ricaviamo che:
 - ▶ **impiegato** \rightarrow **stipendio** e **progetto** \rightarrow **budget**

Dipendenze funzionali

- ▶ **Attenzione** : se prendiamo la chiave K della relazione R si verifica facilmente che esiste una **dipendenza funzionale tra K ed ogni attributo** dello schema
- ▶ infatti **per definizione di chiave** esiste un solo valore di K in R e quindi la dipendenza di cui sopra è **banalmente soddisfatta**
- ▶ nell'esempio:
impiegato, progetto → **stipendio, budget, funzione**

Dipendenze funzionali

Però:

▶ **impiegato, progetto** → **funzione**

è una dipendenza completa,

▶ mentre

impiegato, progetto → **stipendio** e

impiegato, progetto → **budget**

sono in realtà

impiegato → **stipendio** e **progetto** → **budget**

queste sono dipendenze parziali che causano anomalie

Dipendenze funzionali

- ▶ Le ridondanze e le anomalie sono causate da dipendenze $X \rightarrow Y$ che permettono **ripetizioni** all'interno della relazione (**impiegato**, **stipendio** e **progetto**, **budget** si ripetono nella relazione), in altre parole :
- ▶ Le ridondanze e le anomalie sono causate da dipendenze $X \rightarrow Y$ tali che **X non contiene la chiave** della relazione
- ▶ Una relazione R è in forma normale (Boyce e Codd) se, **per ogni dipendenza $X \rightarrow Y$ in R, X contiene una chiave K di R (X è superchiave)**

Dipendenze funzionali

- ▶ Una relazione non in forma normale è **possibile** che venga **decomposta** in due o più relazioni in forma normale
- ▶ la **decomposizione** si può attuare effettuando **proiezioni** in modo tale da ottenere che ciascuna dipendenza funzionale corrisponda ad una relazione separata
- ▶ nell'esempio :
FUNZIONI per **impiegato, progetto** → **funzione**
IMPIEGATI per **impiegato** → **stipendio**
PROGETTI per **progetto** → **budget**

Dipendenze funzionali

impiegato	stipendio
Rossi	2
Verdi	3
Neri	7
Mori	6
Bianchi	6

progetto	budget
biella	300
valvola	500
albero	1500

impiegato	progetto	funzione
Rossi	biella	tecnico
Verdi	valvola	progettista
Verdi	albero	progettista
Neri	albero	direttore
Neri	valvola	consulente
Neri	biella	consulente
Mori	biella	direttore
Mori	albero	progettista
Bianchi	albero	progettista
Bianchi	biella	direttore

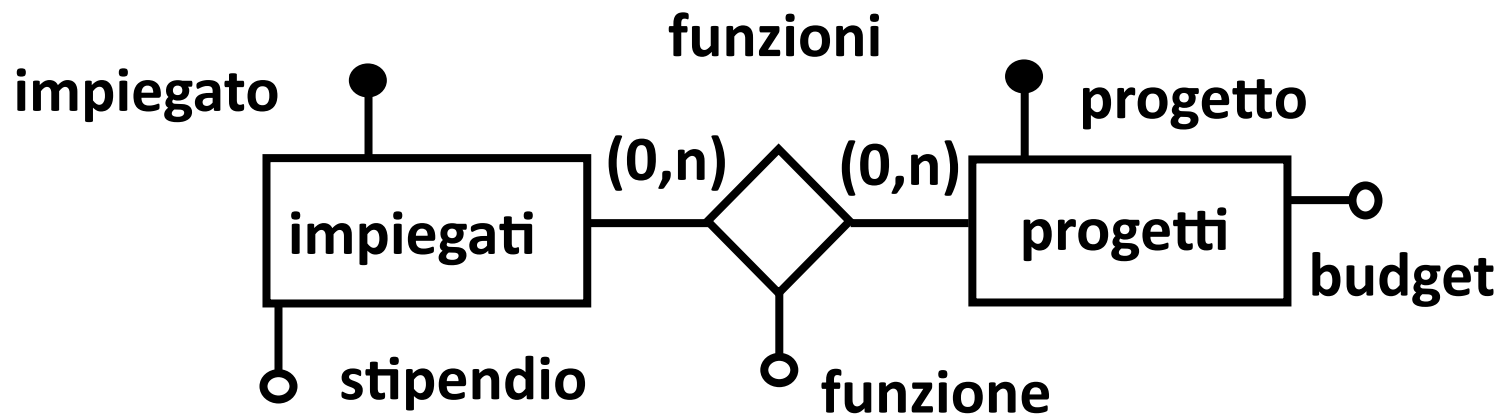
Dipendenze funzionali

- ▶ **FUNZIONI, IMPIEGATI e PROGETTI** sono normalizzate perché soddisfano la definizione di forma normale
- ▶ la relazione non decomposta può essere ricostruita con il **join**:

```
SELECT *  
FROM   IMPIEGATI I, PROGETTI P, FUNZIONI F  
WHERE  I.IMPIEGATO = F.IMPIEGATO  
AND    F.PROGETTO = P.PROGETTO
```

Dipendenze funzionali

- Quando la relazione originale è **ricostruibile** con il **join** la decomposizione è **corretta** e si dice essere **senza perdita**
- notare che lo schema corretto corrisponde alla **traduzione** di:

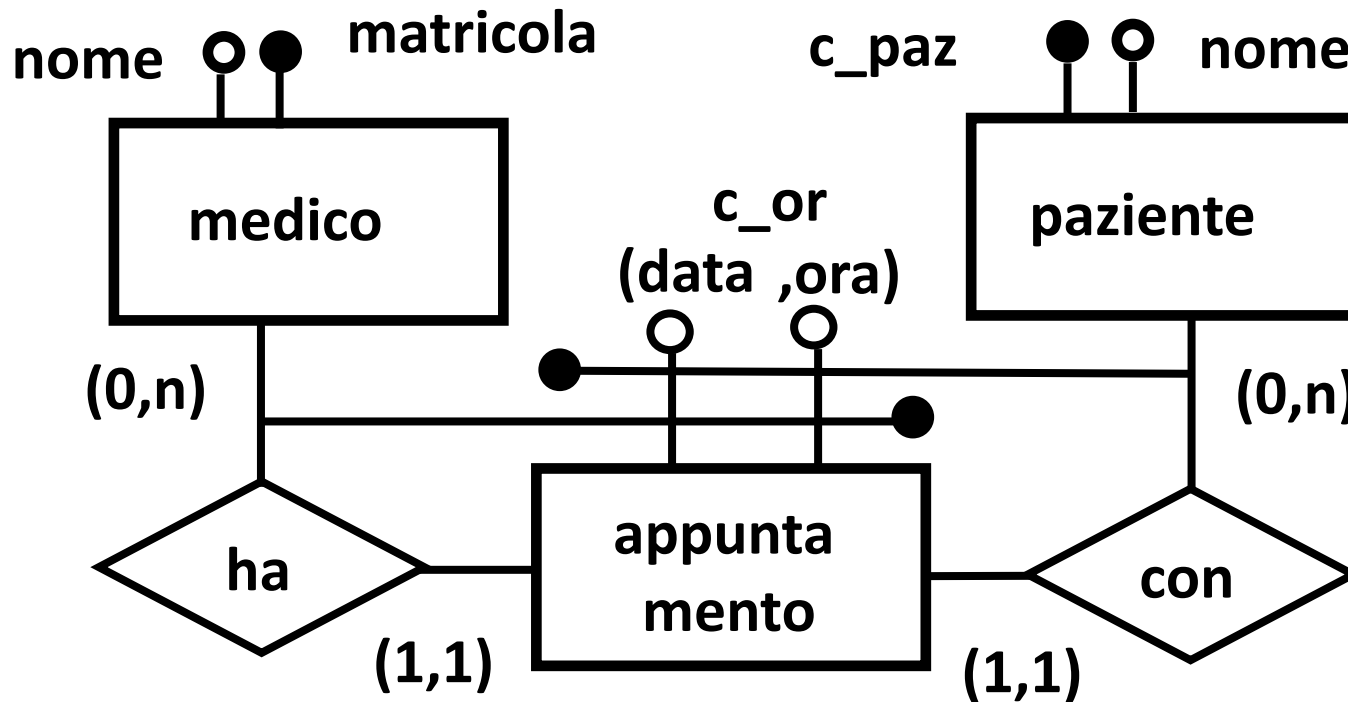


Dipendenze funzionali

- ▶ **Schemi E/R corretti** producono in generale buoni **schemi relazionali** senza problemi di **anomalie** e **ridondanze** e corrispondono a decomposizioni **senza perdita**
- ▶ Schemi E/R dove non tutti i vincoli sono espressi nello schema e/o in presenza di associazioni n_arie possono però venire tradotti (non intenzionalmente) in **schemi relazionali non ben normalizzati**
- ▶ È quindi importante **ricontrollare la normalizzazione**: operazione questa non sempre facile o, possibile per carenza di specifiche

Dipendenze funzionali

ad esempio, lo schema già visto in precedenza:



Dipendenze funzionali

Per mezzo dell'identificazione esterna individua correttamente le **due dipendenze**:

matricola, data, ora → **c_paz**

c_paz, data, ora → **matricola**

e si traduce nello schema relazionale:

medico (**matricola**, **nome**)

paziente (**c_paz**, **cognome**)

appuntamento (**matricola**, **data**, **ora**, **c_paz**)

oppure

appuntamento (**c_paz** , **data**, **ora**, **matricola**)

Dipendenze funzionali

- Non sempre le decomposizioni producono effetti desiderabili
- **decomposizioni errate** possono generare relazioni che, ricongiunte con il join, producono relazioni con dati incerti si ha quindi una **perdita di informazione**
- consideriamo un esempio di relazione:

SEDI (impiegato, progetto, sede)
con le dipendenze:

impiegato, progetto → **sede**
impiegato → **sede** e **progetto** → **sede**

Dipendenze funzionali

chiave di **SEDI** :
impiegato, progetto

Vincolo:
gli impiegati hanno
come sede la sede
dei loro progetti

impiegato	progetto	sede
Rossi	biella	milano
Verdi	valvola	torino
Verdi	albero	torino
Bianchi	cinghia	milano
Neri	valvola	torino

decomponendo secondo le due dipendenze:



Dipendenze funzionali

i
m
p
i
e
g
a
t
i

impiegato	sede
Rossi	milano
Verdi	torino
Bianchi	milano
Neri	torino

progetto	sede
biella	milano
valvola	torino
albero	torino
cinghia	milano

p
r
o
g
e
t
t
i

il join sull'attributo comune:

```
SELECT I.IMPIEGATO, P.PROGETTO, P.SEDE  
FROM IMPIEGATI I, PROGETTI P  
WHERE I.SEDE = P.SEDE
```



Dipendenze funzionali

impiegato	progetto	sede
Rossi	biella	milano
Rossi	cinghia	milano
Verdi	valvola	torino
Verdi	albero	torino
Bianchi	biella	milano
Bianchi	cinghia	milano
Neri	valvola	torino
Neri	albero	torino

← crea tuple
che non
esistevano!

Dipendenze funzionali

- Inoltre , anche se sono corrette, le due relazioni **non rispettano il vincolo** che la sede di un impiegato è la sede dei suoi progetti, un progetto potrebbe cambiare sede indipendentemente dagli impiegati
- **Regola:**
una buona decomposizione deve prevedere la ricostruzione della relazione di partenza con operazioni di **join su chiavi**
- **Osservazione :**
i join su attributi che si corrispondono **n a m** sono **rischiosi**

Dipendenze funzionali

**p
r
o
g
e
t
t
i**

progetto	sede
biella	milano
valvola	torino
albero	torino
cinghia	milano

impiegato	progetto
Rossi	biella
Verdi	valvola
Verdi	albero
Bianchi	cinghia
Neri	valvola

**i
m
p
i
e
g
a
t
i**

il join sull'attributo comune:

```
SELECT I.IMPIEGATO, P.PROGETTO, P.SEDE  
FROM IMPIEGATI I, PROGETTI P  
WHERE I.PROGETTO = P.PROGETTO
```



Dipendenze funzionali

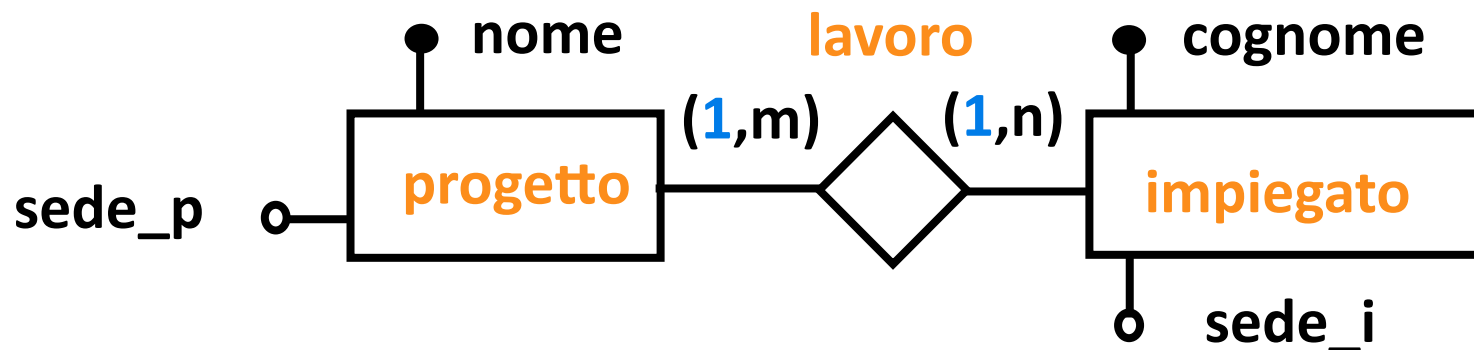
la decomposizione
è corretta
però abbiamo perso
nello schema
precedente la
dipendenza
impiegato → sede

impiegato	progetto	sede
Rossi	biella	milano
Verdi	valvola	torino
Verdi	albero	torino
Bianchi	cinghia	milano
Neri	valvola	torino

problema: che ne è di Verdi se il progetto Albero
va a Roma?

Dipendenze funzionali

Se fossimo partiti dallo schema E/R?



avremmo comunque dovuto dichiarare, a parte dallo schema E/R, il vincolo che la sede di un impiegato deve essere uguale alla sede dei progetti in cui lavora, e lo schema relazionale:



Dipendenze funzionali

progetti

progetto	sede
biella	milano
valvola	torino
albero	torino
cinghia	milano

impiegato	sede
Rossi	milano
Verdi	torino
Bianchi	milano
Neri	torino

impiegati

impiegato	progetto
Rossi	biella
Verdi	valvola
Verdi	albero
Bianchi	cinghia
Neri	valvola

lavoro

questa soluzione
consente anche (0,n)
nell'associazione

Dipendenze funzionali

il join adesso sarà:

```
SELECT I.IMPIEGATO, P.PROGETTO, P.SEDE  
FROM IMPIEGATI I, PROGETTI P, LAVORO L  
WHERE I.IMPIEGATO = L.IMPIEGATO  
AND L.PROGETTO = P.PROGETTO
```

che ottiene la relazione richiesta senza perdita
perché lavora su chiavi

però non c'è garanzia sull'uguaglianza di sede

Conclusioni particolari

- ▶ **è difficile progettare** una soluzione relazionale che mantenga le tre dipendenze senza l'aiuto di software scritto ad hoc
- ▶ **è possibile ricavare** una soluzione senza perdita
- ▶ partendo dallo schema E/R si ottiene una **soluzione più flessibile** perché consente inserimenti indipendenti di nuovi progetti ed impiegati **senza** introdurre valori **nulli**

Conclusioni generali

- ▶ Progettare i dati è **difficile**
- il lavoro di gruppo è **importantissimo** per evitare differenze di **percezione** e di **visione** dei problemi
- DFD, schemi E/R, dipendenze funzionali sono **utilissimi** per descrivere e capire i problemi
 - ▶ **tanta più** conoscenza si riesce a descrivere negli schemi, **tanta meno** verrà espressa in con vincoli meno leggibili o, dispersa in programmi di difficile lettura e aggiornamento

Conclusioni generali

- è bene però anche **non eccedere** con schemi particolarmente complicati contenenti un eccesso di concetti fittizi e di collegamento che rendono difficile la lettura e la soluzione innaturale
- la documentazione di progetto è pertanto **fondamentale**