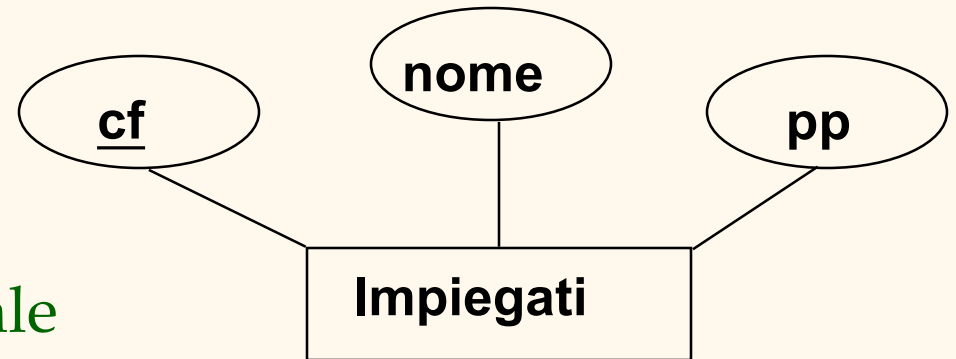


*Progettazione concettuale usando il
modello Entità-Relazione (ER) e
Progettazione Logica*

Introduzione alla progettazione delle basi di dati

- ❖ Progettazione concettuale (in questa fase si usa il modello ER)
 - Quali sono le entità e le relazioni dell'organizzazione?
 - Quali informazioni su queste entità e relazioni dovrebbero essere memorizzate nella base di dati?
 - Quali sono i vincoli di integrità o le *business rules* in vigore?
 - Uno "schema" di base di dati nel modello ER può essere rappresentato graficamente (diagrammi ER)
 - Si può tradurre un diagramma ER in uno schema relazionale
- ❖ Raffinamento dello schema (normalizzazione): controllo dello schema relazionale per trovare ridondanze e relative anomalie.
- ❖ Progettazione fisica e ulteriore raffinamento dello schema: si considerano il carico di lavoro e le prestazioni del sistema per effettuare ulteriori modifica sullo schema.

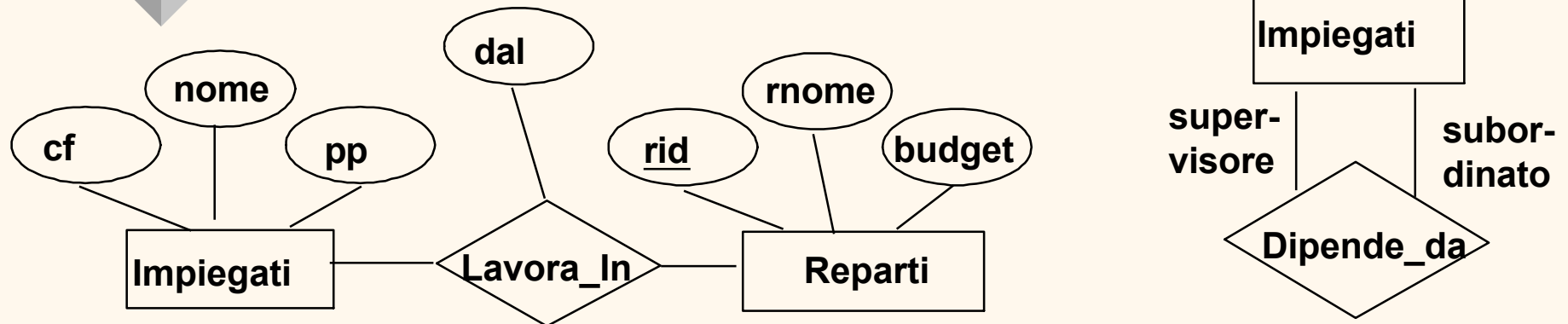
Basi del modello ER



- ❖ Entità: oggetto del mondo reale distinguibile da altri oggetti
- ❖ Una entità è descritta (nel DB) usando un insieme di attributi
 - Insieme di entità: una collezione di entità simili. Ad esempio, tutti gli impiegati
 - Tutte le entità in un insieme di entità hanno lo stesso insieme di attributi (almeno fino a quando non consideriamo gerarchie ISA!)
 - Ciascun insieme di entità ha una chiave
 - Ciascun attributo ha un dominio
 - Si può tradurre facilmente un insieme di entità in una tabella relazionale

```
CREATE TABLE Impiegati  
(cf CHAR(15),  
nome CHAR(20),  
pp INTEGER,  
PRIMARY KEY (cf))
```

Basi del modello ER (segue)



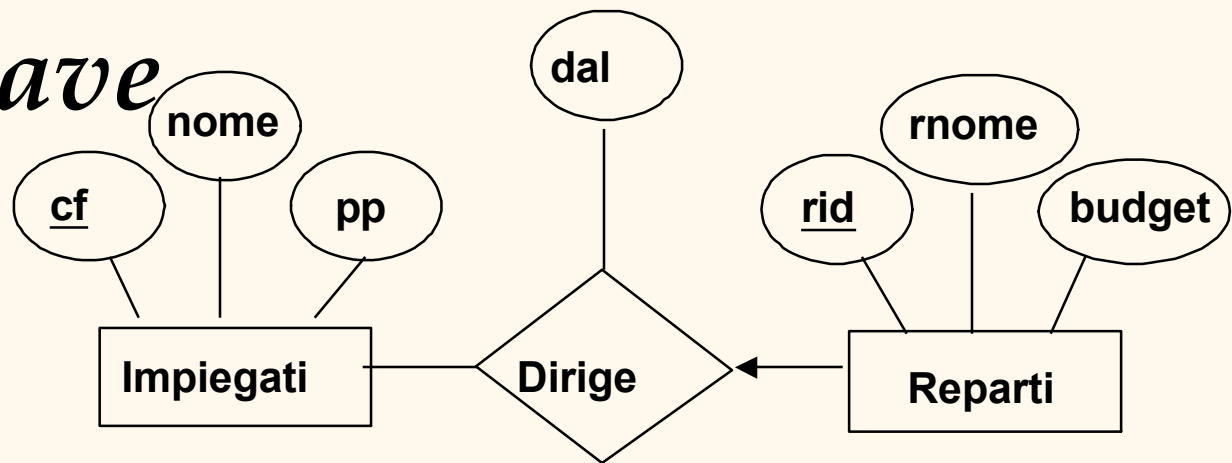
- ❖ **Relazione:** associazione tra 2 o più entità. Ad esempio, Attishoo lavora nel reparto Farmacia
- ❖ **Insieme di relazioni:** collezione di relazioni simili
 - Un insieme n-ario di relazioni R correla n insiemi di entità $E_1.. E_n$; ciascuna relazione in R coinvolge entità $e_1 \in E_1, \dots, e_n \in E_n$
 - ◆ Lo stesso insieme di entità può partecipare a diversi insiemi di relazioni, o in “ruoli” differenti all’interno dello stesso insieme

Basi del modello ER (segue)

- ❖ Gli insiemi di relazioni possono anche avere attributi descrittivi (ad esempio l'attributo *dal* di Lavora_In)
- ❖ Nel tradurre un insieme di relazioni ER in una tabella relazionale, gli attributi della tabella devono includere:
 - chiavi per ciascun insieme di entità partecipante (come chiavi esterne)
 - ◆ questo insieme di attributi forma la superchiave per la relazione
 - tutti gli attributi descrittivi

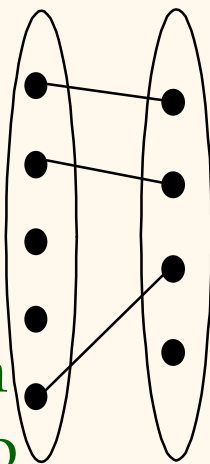
```
CREATE TABLE Lavora_In(  
  cf CHAR(15),  
  rid INTEGER,  
  dal DATE,  
  PRIMARY KEY (cf, rid),  
  FOREIGN KEY (cf)  
    REFERENCES Impiegati,  
  FOREIGN KEY (rid)  
    REFERENCES Reparti)
```

Vincoli di chiave

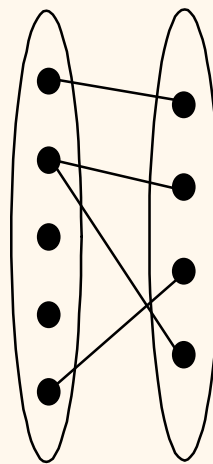


- ❖ Consideriamo Lavora_In: un impiegato può lavorare in molti reparti; un reparto può avere molti impiegati

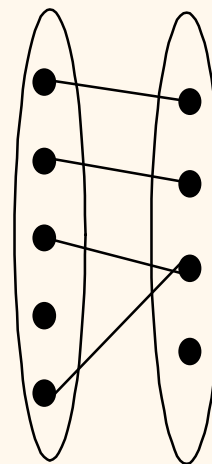
- ❖ Di contro, ciascun reparto ha al più un direttore, in accordo con il vincolo di chiave su Dirige



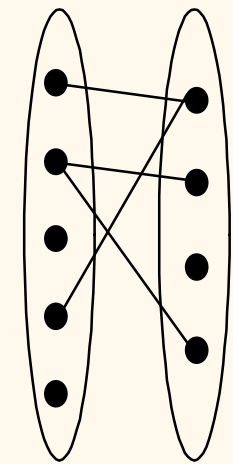
1-a-1



1-a-Molti



Molti-a-1



Molti-a-molti

➡ Traduzione nel modello relazionale?

Traduzione dei diagrammi ER con vincoli di chiave

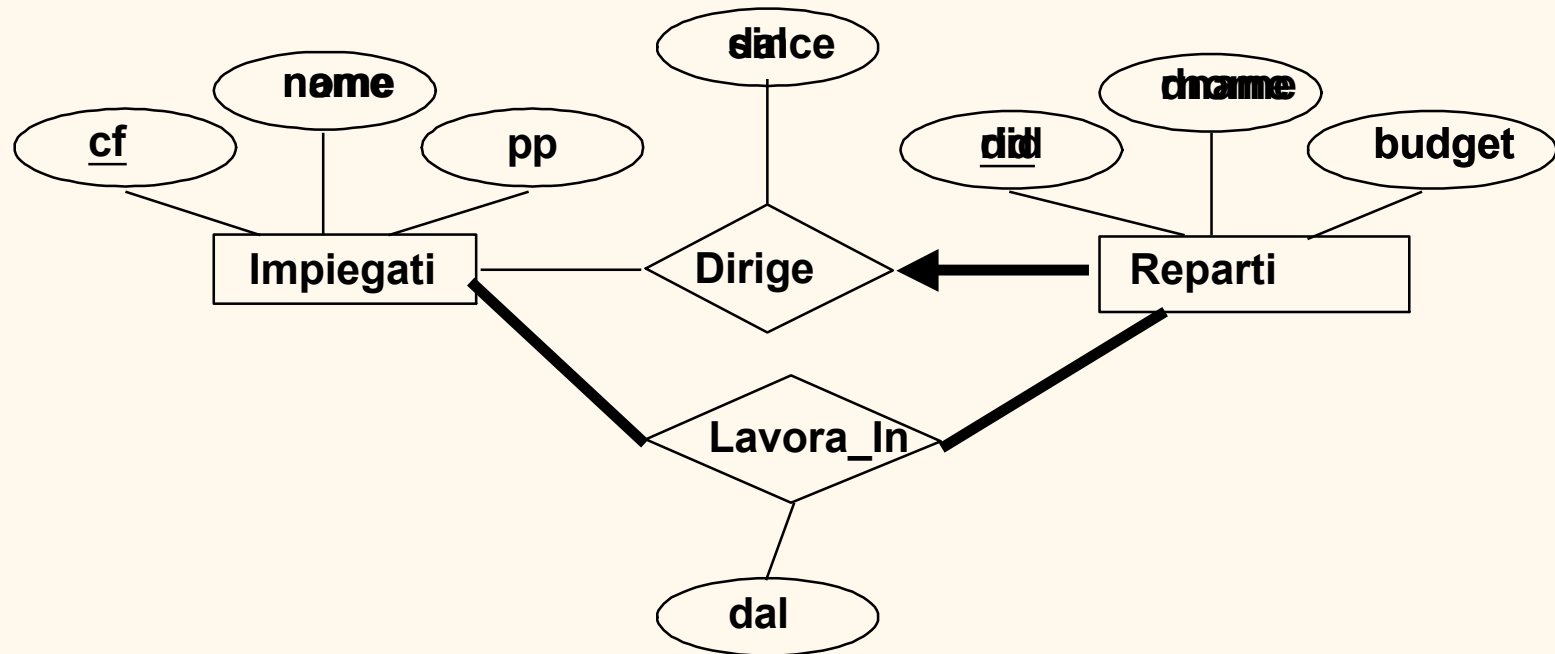
- ❖ Tradurre una relazione in una tabella:
 - notare che adesso la chiave è *rid*!
 - separare le tabelle per Impiegati e Reparti
- ❖ Poiché ciascun reparto ha un unico direttore, possiamo invece combinare Dirige e Reparti

```
CREATE TABLE Dirige(  
  cf CHAR(15),  
  rid INTEGER,  
  dal DATE,  
  PRIMARY KEY (rid),  
  FOREIGN KEY (cf) REFERENCES Impiegati,  
  FOREIGN KEY (rid) REFERENCES Reparti)
```

```
CREATE TABLE Rep_Dir(  
  Rid INTEGER,  
  Rnome CHAR(20),  
  Budget REAL,  
  Cf CHAR(15),  
  Dal DATE,  
  PRIMARY KEY (rid)  
  FOREIGN KEY (cf) REFERENCES Impiegati)
```

Vincoli di partecipazione

- ❖ C'è un direttore per ogni reparto?
 - Se sì, questo è un vincolo di partecipazione: la partecipazione di Reparti in Dirige viene detta *totale* (piuttosto che *parziale*)
 - ◆ Ogni valore *rid* nella tabella Reparti deve apparire in una riga della tabella Gestisce (con un valore di *cf* non nullo!)



Vincoli di cardinalità

- ❖ Vincoli di chiave e di partecipazione considerati insieme sono anche detti vincoli di cardinalità (essi definiscono una cardinalità massima e minima, rispettivamente)
- ❖ I vincoli di cardinalità definiscono il massimo (minimo) numero di istanze della relazioni cui partecipa una istanza dell'entità

Ex.:



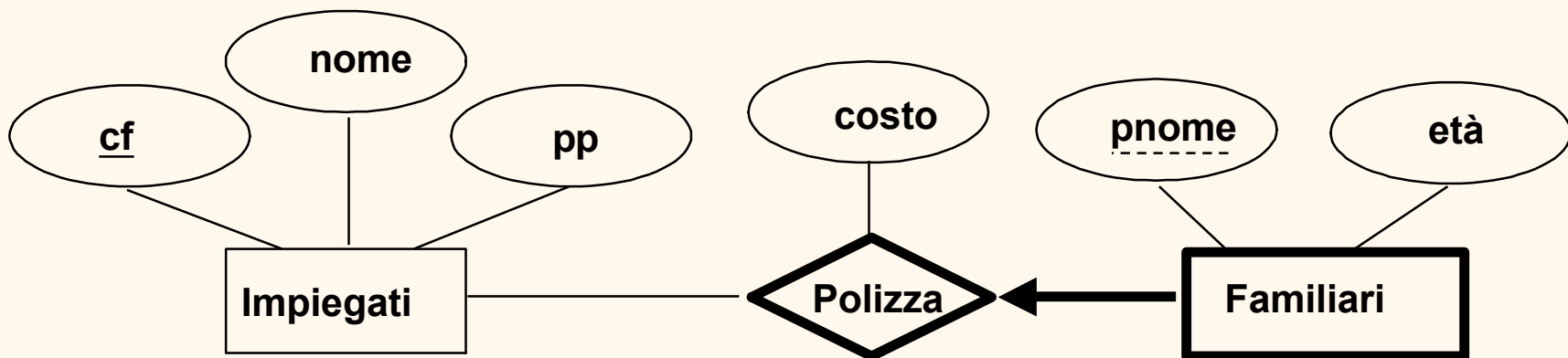
Vincoli di partecipazione in SQL

- Possiamo catturare i vincoli di partecipazione che coinvolgono un insieme di entità in una relazione binaria, ma poche altre cose (senza ricorrere a vincoli CHECK)

```
CREATE TABLE Rep_Dir(  
  Rid INTEGER,  
  Rnome CHAR(20),  
  Budget REAL,  
  Cf CHAR(15),  
  Dal DATE,  
  PRIMARY KEY (rid),  
  FOREIGN KEY (cf) REFERENCES Impiegati,  
  ON DELETE NO ACTION)
```

Entità deboli

- ❖ Una entità debole può essere indentificata univocamente solo considerando la chiave primaria di un'altra entità (proprietario)
- ❖ L'insieme di entità proprietarie e l'insieme di entità deboli devono partecipare in un insieme di relazioni uno-a-molti (1 proprietario, molte entità deboli)
- ❖ L'insieme di entità deboli deve avere partecipazione totale in questo insieme di relazioni identificanti

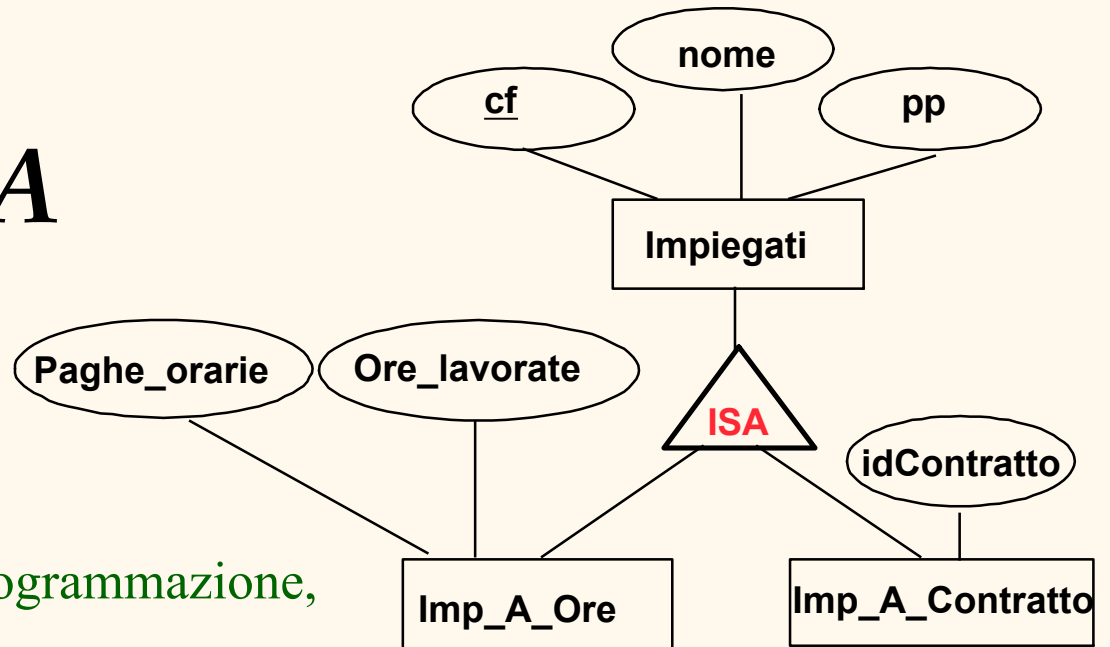


Traduzione di insiemi di entità deboli

- ❖ Un insieme di entità deboli e l'insieme di relazioni identificanti sono tradotte in una tabella singola
 - Quando l'entità proprietaria viene cancellata, anche tutte le sue entità deboli devono essere cancellate

```
CREATE TABLE Politica_Rep(  
  Rnome CHAR(20),  
  età INTEGER,  
  costo REAL,  
  cf CHAR(15) NOT NULL,  
  PRIMARY KEY (rnome, cf)  
  FOREIGN KEY (cf) REFERENCES Impiegati,  
  ON DELETE CASCADE)
```

Gerarchie ISA



Come in C++, o altri linguaggi di programmazione, gli attributi vengono ereditati.

Se dichiariamo A ISA B, ogni entità A

è considerata anche come entità B (le risposte alle interrogazioni dovrebbero riflettere questo fatto)

Vincoli di sovrapposizione: può Joe essere un Imp_A_Ore e anche una entità di Imp_A_Contratto? (Permesso/non permesso)

Vincoli di copertura: deve, ogni entità di Impiegati, essere anche una entità di Imp_A_Ore o una entità di Imp_A_Contratto? (Sì/No)

Motivi per usare ISA:

- Aggiungere attributi descrittivi specifici a una sottoclasse

- Identificare le entità che partecipano a una relazione

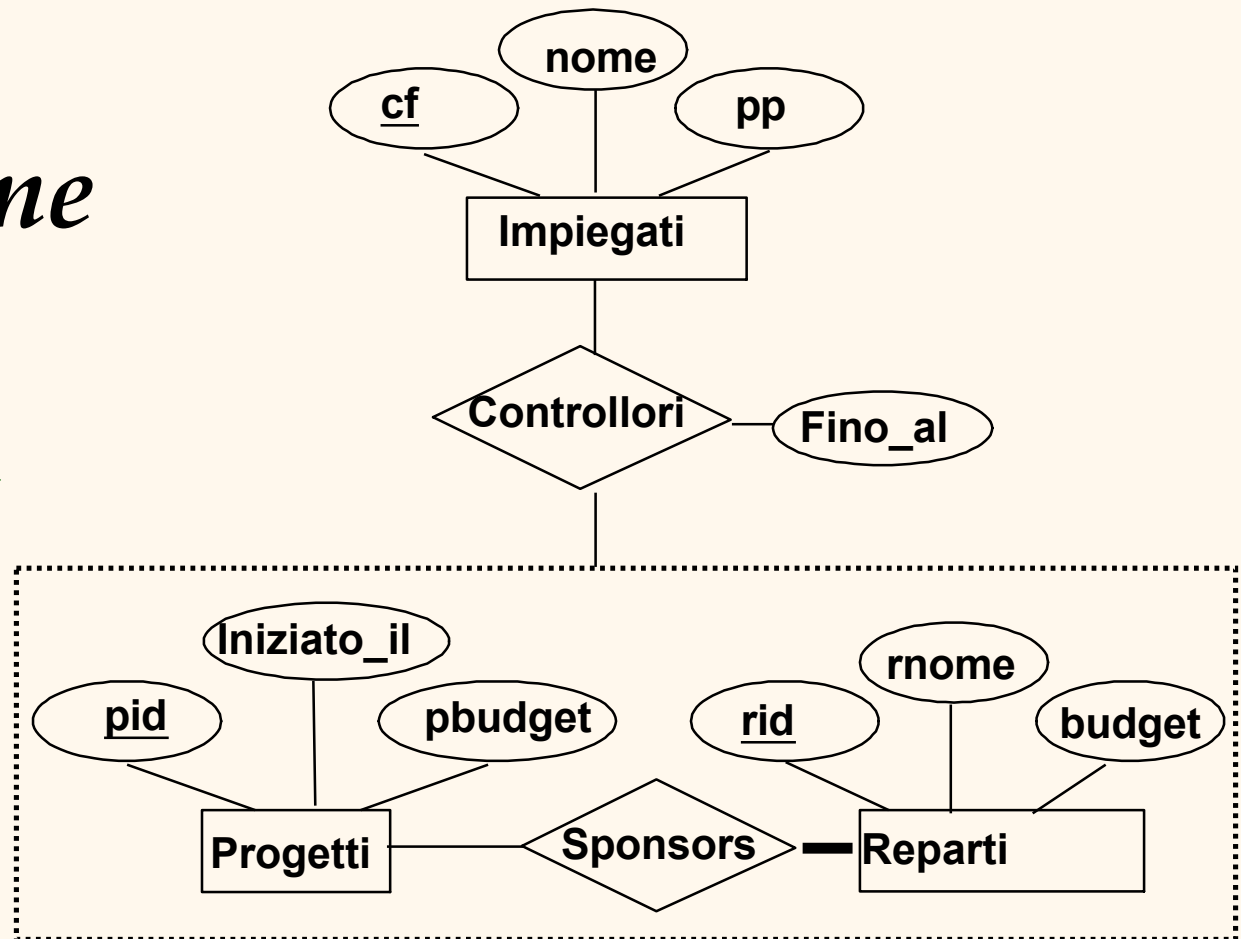
Traduzione delle gerarchie ISA in tabelle relazionali

- ❖ **Approccio generale:**
 - 3 relazioni: Impiegati, Imp_A_Ore e Imp_A_Contratto
 - ◆ Imp_A_Ore: ogni impiegato è registrato in Impiegati. Per gli impiegati a ore, informazioni aggiuntive sono registrate in Imp_A_Ore (paghe_orarie, ore_lavorate, CF); bisogna cancellare le tuple di Imp_A_Ore se viene cancellata la relativa tupla di Impiegati
 - ◆ Le interrogazioni che coinvolgono tutti gli impiegati sono facili, quelle che coinvolgono solo Imp_A_Ore richiedono un join per leggere alcuni attributi
- ❖ **Alternativa: solo Imp_A_Ore e Imp_A_Contratto**
 - Imp_A_Ore: cf, nome, pp, paghe_orarie, ore_lavorate
- ❖ **Ciascun impiegato deve appartenere a una di queste due sottoclassi**

Aggregazione

❖ Usata quando dobbiamo modellare una relazione che coinvolge (insiemi di entità e) un insieme di relazioni

- L'aggregazione ci permette di trattare un insieme di relazioni come un insieme di entità allo scopo di permetterne la partecipazione in (altre) relazioni
- Controllori è tradotta con una tabella come qualunque altro insieme di relazioni



➡ **Aggregazione verso Relazione ternaria:**
Controllori è una relazione distinta, con un attributo descrittivo
Inoltre, possiamo dire che ogni sponsorizzazione è monitorata da al più un impiegato

Progettazione concettuale usando il modello ER

- ❖ **Scelte di progetto:**
 - Un concetto dovrebbe essere modellato come una entità o come un attributo?
 - Un concetto dovrebbe essere modellato come una entità o come una relazione?
 - Identificare le relazioni: binarie o ternarie? Aggregazione?
- ❖ **Vincoli nel modello ER:**
 - molta della semantica dei dati può (e dovrebbe) essere catturata
 - però alcuni vincoli non possono essere catturati dai diagrammi ER
- ❖ **Necessità di ulteriori raffinamenti dello schema:**
 - lo schema relazionale ottenuto dai diagrammi ER è un buon punto di partenza. Ma il progetto ER è soggettivo e non può esprimere certi vincoli; quindi questo lo schema relazionale può aver bisogno di raffinamenti

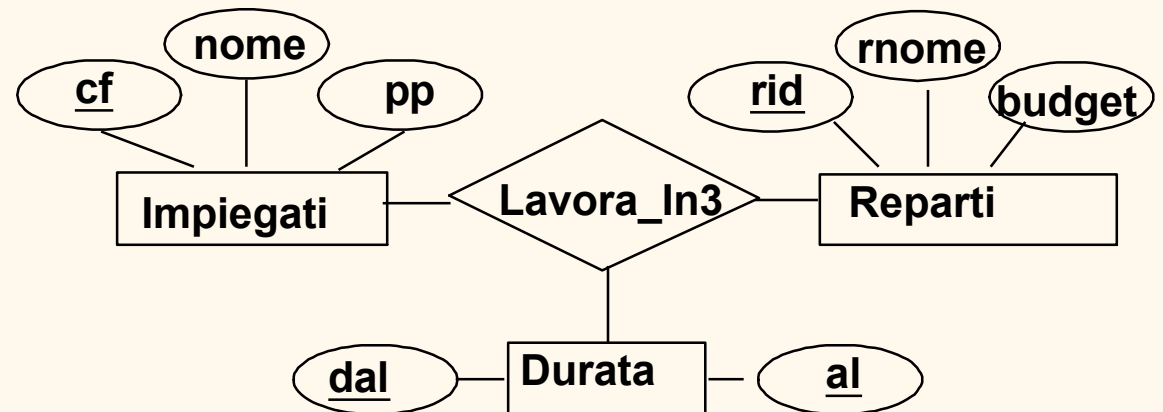
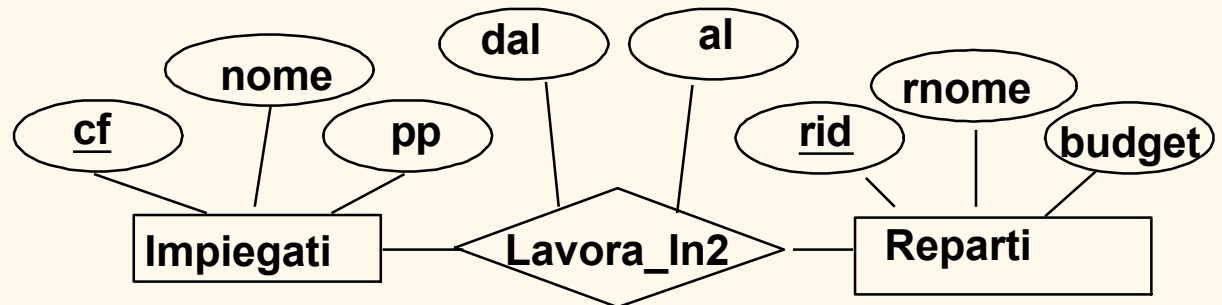


Entità verso Attributi

- ❖ *Indirizzo* dovrebbe essere un attributo di Impiegati o una entità (connessa a Impiegati da una relazione)?
- ❖ Dipende dall'uso che vogliamo fare delle informazioni sull'indirizzo, e dalla semantica dei dati:
 - se abbiamo diversi indirizzi per impiegati, *indirizzo* deve essere una entità (poiché gli attributi non possono assumere più valori)
 - se la struttura (città, via, etc) è importante, ad esempio vogliamo trovare gli impiegati in una data città, *indirizzo* deve essere modellato come una entità (poiché i valori degli attributi sono atomici)

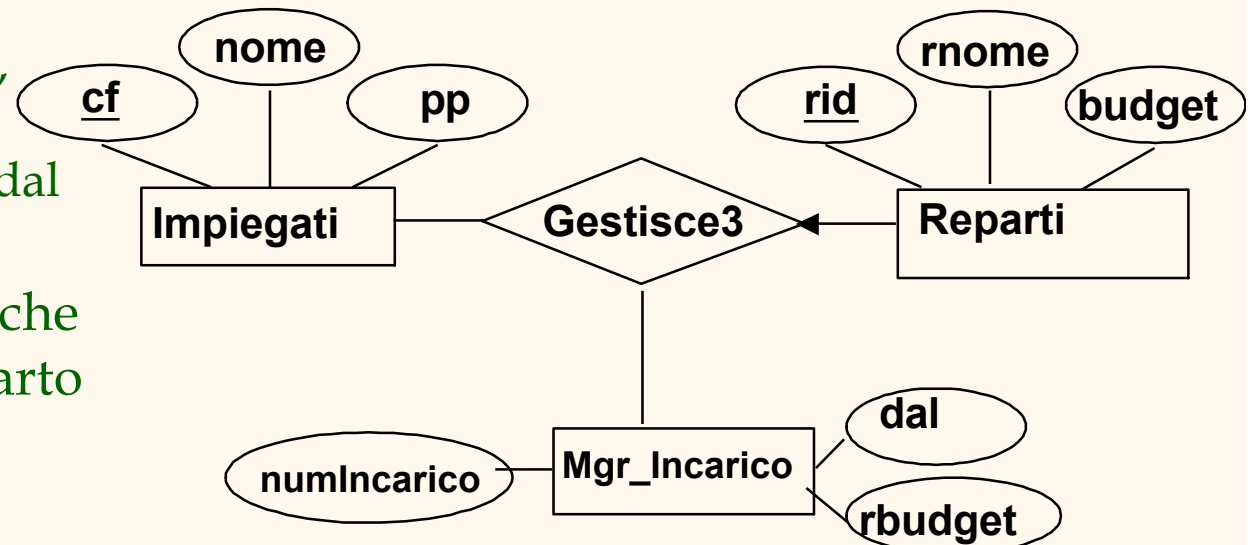
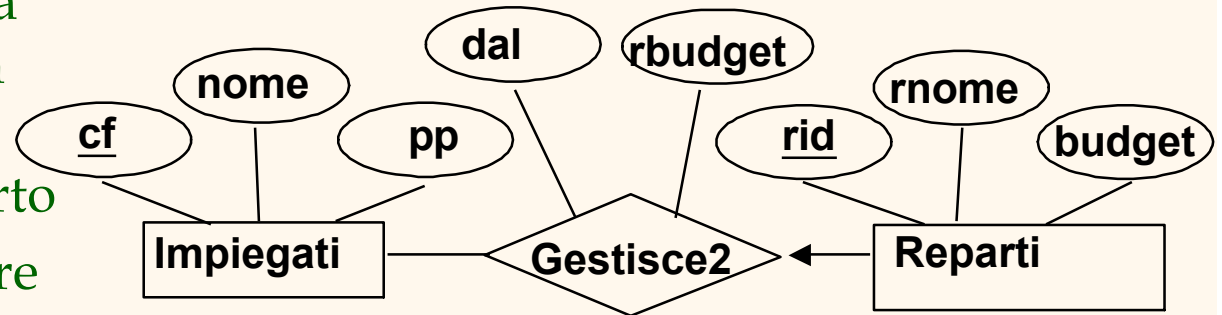
Entità verso Attributi (segue)

- ❖ Lavora_In2 non permette a un impiegato di lavorare in un reparto per due o più periodi
- ❖ Simile al problema della registrazione di indirizzi multipli per un impiegato: vogliamo registrare diversi valori degli attributi descrittivi per ciascuna istanza della relazione



Entità verso Relazioni

- ❖ Il primo diagramma ER va bene se un direttore ha un budget discrezionale separato per ciascun reparto
- ❖ Che succede se un direttore ha un budget discrezionale che copre *tutti* i reparti da lui diretti?
 - Ridondanza di rbudget, che è memorizzato per ciascun reparto gestito dal direttore
- ❖ Ingannevole: suggerisce che rbudget sia legato al reparto

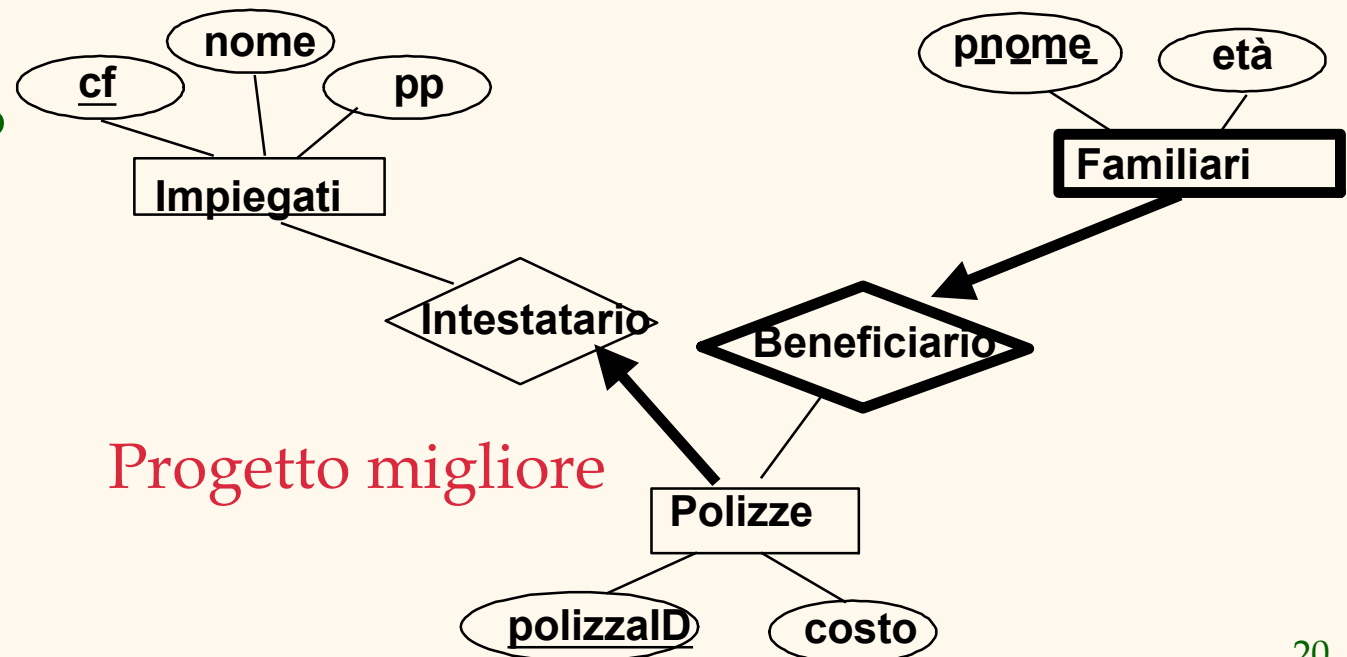
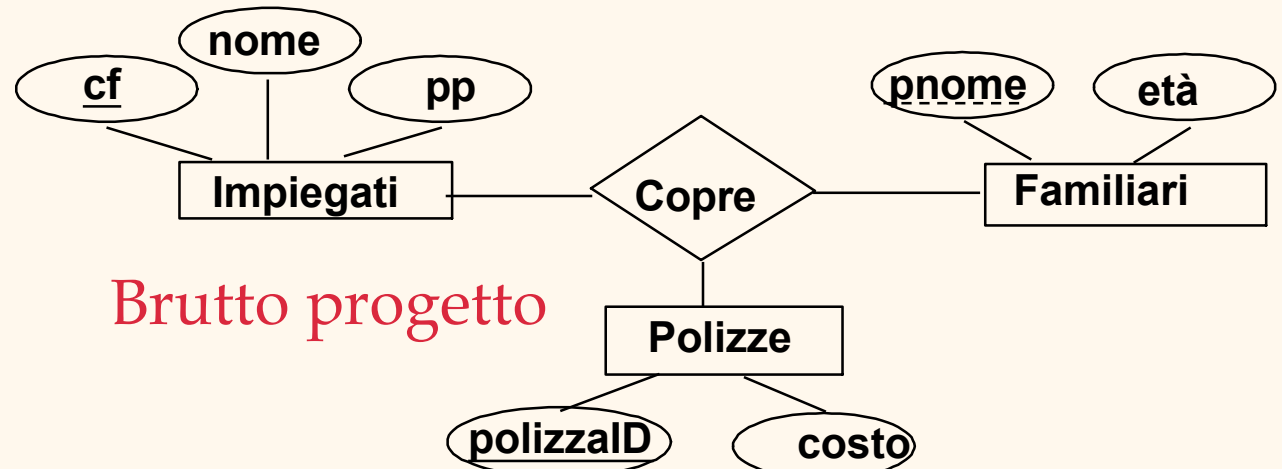


Relazioni binarie verso relazioni ternarie

❖ Se ogni polizza è posseduta da un solo impiegato:

- un vincolo di chiave su Polizze significherebbe che una polizza può coprire solo un familiare!

❖ Quali sono i vincoli aggiuntivi nel secondo diagramma?



Relazioni binarie verso relazioni ternarie (segue)

- ❖ Il vincolo di chiave ci permette di combinare Acquirente con Polizze e Beneficiario con Familiari

```
CREATE TABLE Polizze(  
  polizzaID INTEGER,  
  costo REAL,  
  cf CHAR(15) NOT NULL,  
  PRIMARY KEY (polizzaID),  
  FOREIGN KEY (CF) REFERENCES Impiegati  
  ON DELETE CASCADE)
```

- ❖ I vincoli di partecipazione portano a vincoli NOT NULL

```
CREATE TABLE Familiari (  
  Pnome CHAR(20),  
  Età INTEGER,
```

- ❖ Che succederebbe se Polizze fosse un insieme di entità deboli?

```
  polizzaID INTEGER,  
  PRIMARY KEY (pnome, polizzaID),  
  FOREIGN KEY (polizzaID) REFERENCES Polizze  
  ON DELETE CASCADE)
```

Relazioni binarie verso relazioni ternarie (segue)

- ❖ L'esempio precedente illustra un caso in cui due relazioni binarie sono migliori di una relazione ternaria
- ❖ Un esempio nell'altra direzione: una relazione ternaria Contratti mette in relazione gli insiemi Pezzi, Reparti e Fornitori, e ha l'attributo descrittivo *qta*. Nessuna combinazione di relazioni binarie è un sostituto adeguato:
- ❖ F "può fornire" P, D "ha bisogno" di P, e D "tratta con" F non implicano che D sia d'accordo sul comprare P da F
- ❖ Come registriamo *qta*?

Vincoli oltre il modello ER

❖ Dipendenze funzionali:

- ad esempio, un reparto può ordinare due pezzi distinti dallo stesso fornitore
 - ◆ Non si può esprimere in termini della relazione ternaria Contratti
- La normalizzazione raffina il progetto ER tenendo in considerazione le DF

❖ Dipendenze di inclusione:

- caso speciale: chiavi esterne (si possono esprimere nel modello ER)
- ad esempio, almeno una persona deve dipendere da ciascun direttore (l'insieme dei valori *cf* in Dirige deve essere un sottoinsieme dei valori *supervisore_cf* in Dipende_da) Chiave esterna? Esprimibile nel modello ER?

❖ Vincoli generali

- Ad esempio, budget discrezionale del direttore minore del 10% dei budget combinati di tutti i reparti che gestisce

Riassunto della progettazione concettuale

- ❖ La progettazione concettuale segue l'analisi dei requisiti
 - Porta a una descrizione ad alto livello dei dati che devono essere memorizzati
- ❖ Il modello ER è molto usato per la progettazione concettuale
 - I costrutti sono espressivi, vicini al modo in cui la gente pensa alle applicazioni
- ❖ Costrutti di base: entità, relazioni e attributi (di entità e relazioni)
- ❖ Alcuni costrutti addizionali: entità deboli, gerarchie ISA e aggregazione
- ❖ Nota: ci sono molte varianti del modello ER

Riassunto della progettazione concettuale (segue)

- ❖ Nel modello ER si possono esprimere diversi tipi di vincoli di integrità: vincoli di chiave, vincoli di partecipazione e vincoli di sovrapposizione/ copertura per le gerarchie ISA. Nella definizione di insieme di relazioni sono anche impliciti alcuni vincoli di chiave esterna.
 - Alcuni di questi vincoli possono essere espressi in SQL solo se usiamo vincoli CHECK o asserzioni generali.
 - Alcuni vincoli (in particolare le dipendenze funzionali) non possono essere espressi nel modello ER
 - I vincoli giocano un ruolo importante nel determinare il miglior progetto di base di dati per una organizzazione.

Riassunto della progettazione concettuale (segue)

- ❖ Il progetto ER è soggettivo. Ci sono spesso molti modi per modellare un dato scenario! Analizzare le alternative può essere complicato, specialmente per una grande azienda. Scelte comuni includono:
 - ❖ Entità verso attributo, entità verso relazione, relazione binaria o n-aria, uso delle gerarchie ISA, uso dell'aggregazione
 - ❖ Garantire un buon progetto della base di dati: lo schema relazionale risultante dovrebbe essere analizzato e ulteriormente raffinato. Le informazioni delle DF e le tecniche di normalizzazione sono particolarmente utili.



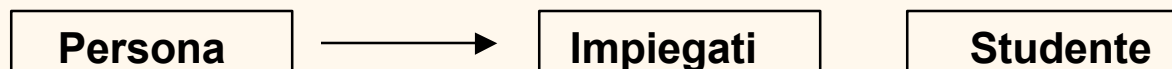
Metodologie di progetto

- ❖ Le metodologie per la progettazione concettuale sono basate su:
 - raffinamento incrementale dello schema
 - astrazione come meccanismo di raffinamento
 - controllo della qualità dello schema concettuale
 - decomposizione in sottoschemi
 - analisi integrata di dati e funzioni
- ❖ Le principali metodologie sono
 - top-down
 - bottom-up
 - mista

Top-down

- ❖ Implementa una serie di trasformazioni dello schema iniziando con concetti molto astratti per andare verso concetti più concreti e dettagliati
- ❖ Il primo schema contiene (in una rappresentazione molto astratta) tutto il contenuto di informazione della realtà da rappresentare
- ❖ Meccanismo di trasformazione: raffinamento

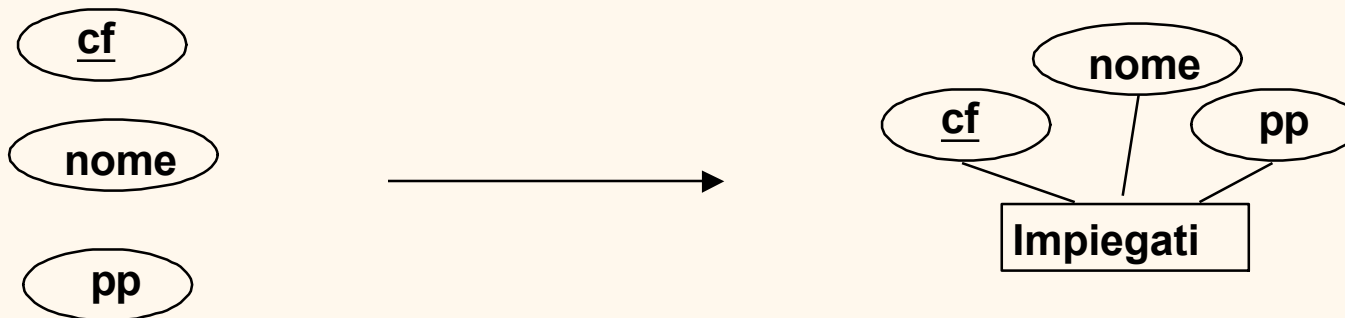
Es.



Bottom-up

- ❖ Parte dai requisiti dettagliati, raggruppandoli in concetti più astratti
- ❖ Meccanismo di trasformazione: astrazione

Es.



Problemi: conflitti tra le definizioni dei concetti, ridondanze, necessità di ristrutturazione (ma PIÙ NATURALE)



Mista

- ❖ Decomposizione “controllata” dei requisiti
- ❖ Struttura globale dello schema
- ❖ Generazione bottom-up dei sottoschemi
- ❖ Integrazione dei sottoschemi guidata dalla struttura



Qualità dello schema concettuale

- ❖ Correttezza: lo schema rappresenta correttamente (sintatticamente e semanticamente) i requisiti iniziali
- ❖ Completezza: lo schema rappresenta tutti i requisiti
- ❖ Minimalità: lo schema rappresenta solo i requisiti e ogni loro aspetto appare solo una volta
- ❖ Leggibilità: lo schema è facile da interpretare ed esprime i requisiti in modo naturale
- ❖ Modificabilità: lo schema può essere facilmente modificato se i requisiti cambiano
- ❖ Auto-documentabilità: lo schema non ha bisogno di materiale aggiuntivo esterno

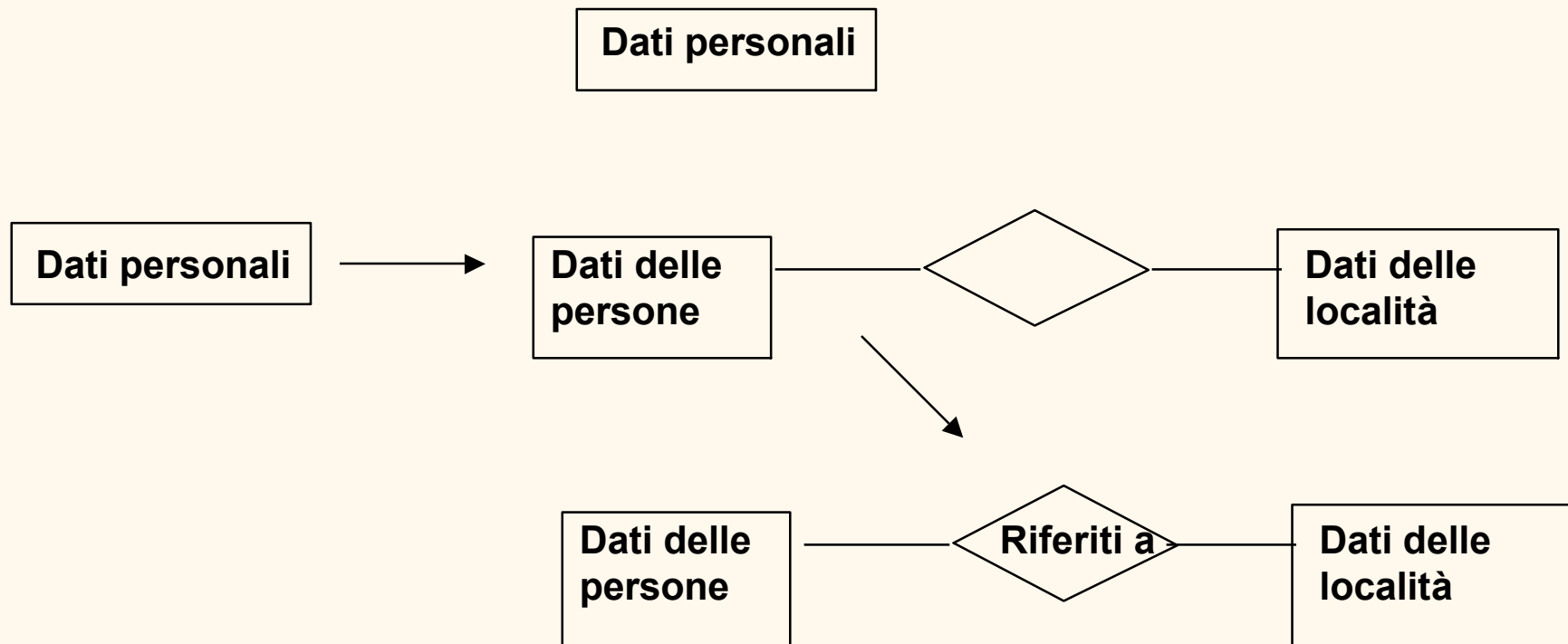
Esempio di top-down

Requisiti

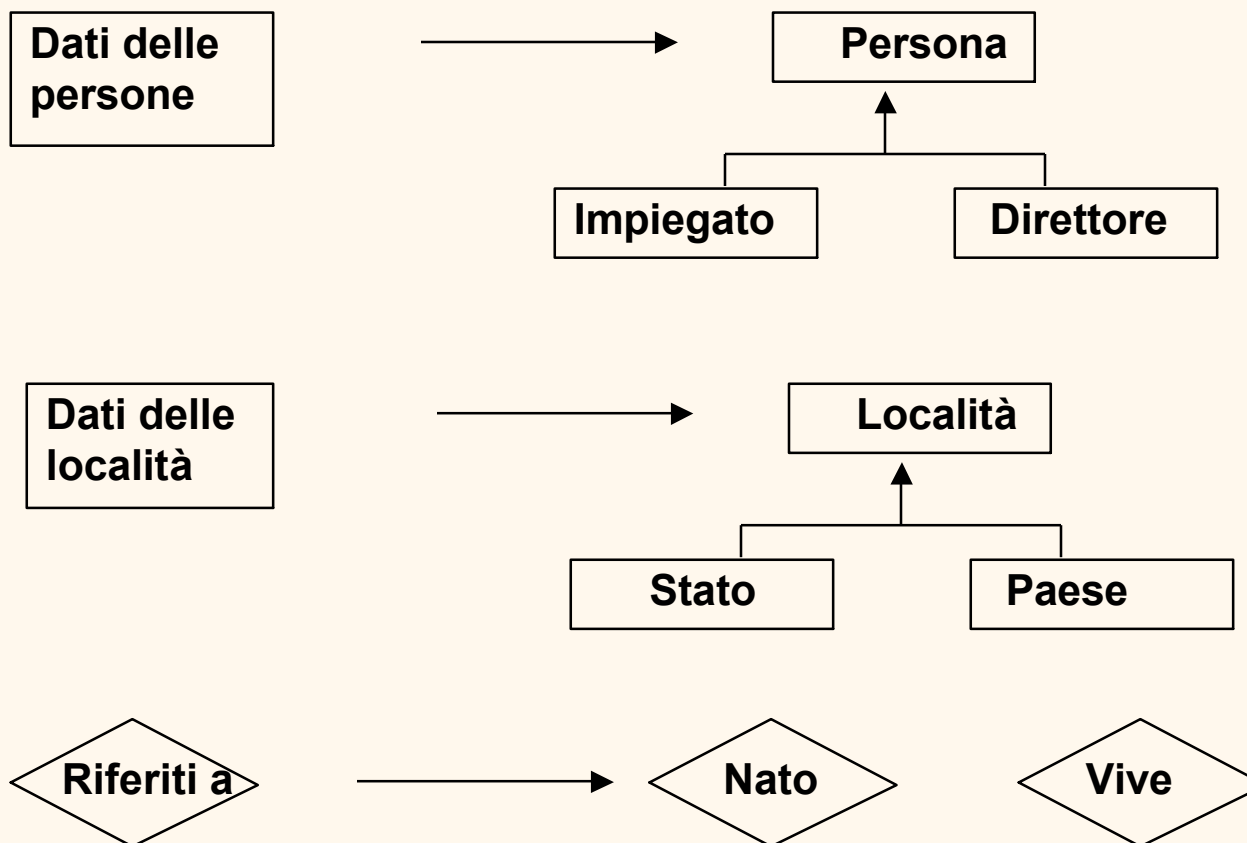
Vogliamo rappresentare le informazioni sugli impiegati. In particolare, i dati si riferiscono a:

- ❖ persone, con nome, data di nascita e sesso;
- ❖ località dove le persone sono nate e vivono attualmente. Ci sono due tipi di località: stati degli USA e paesi stranieri. Nel primo caso, vogliamo rappresentare anche l'ammontare della popolazione dello stato, mentre nel secondo caso vogliamo rappresentare la capitale del paese;
- ❖ impiegati con il loro salario;
- ❖ direttori con associato il nome del reparto che dirigono

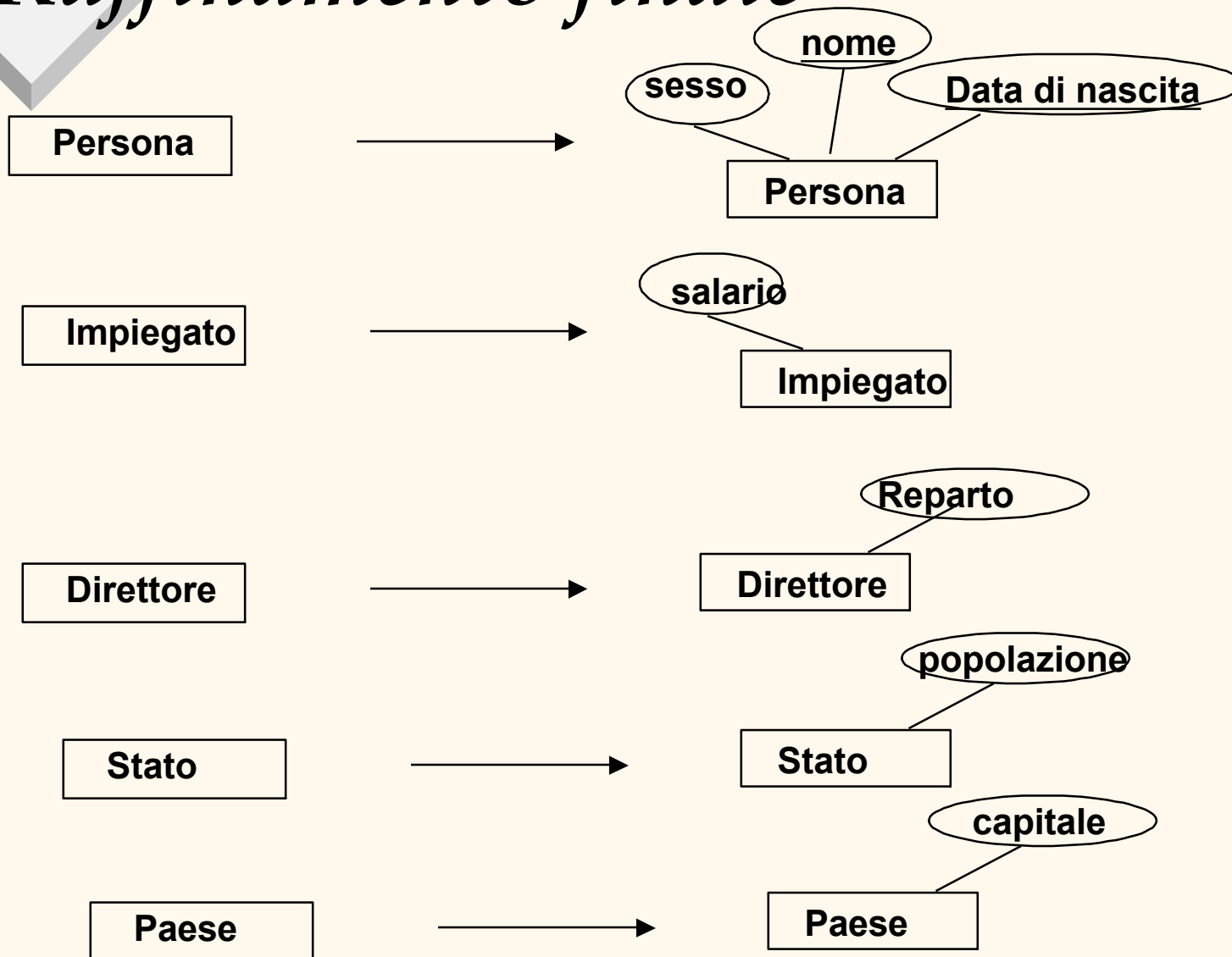
Primi raffinamenti



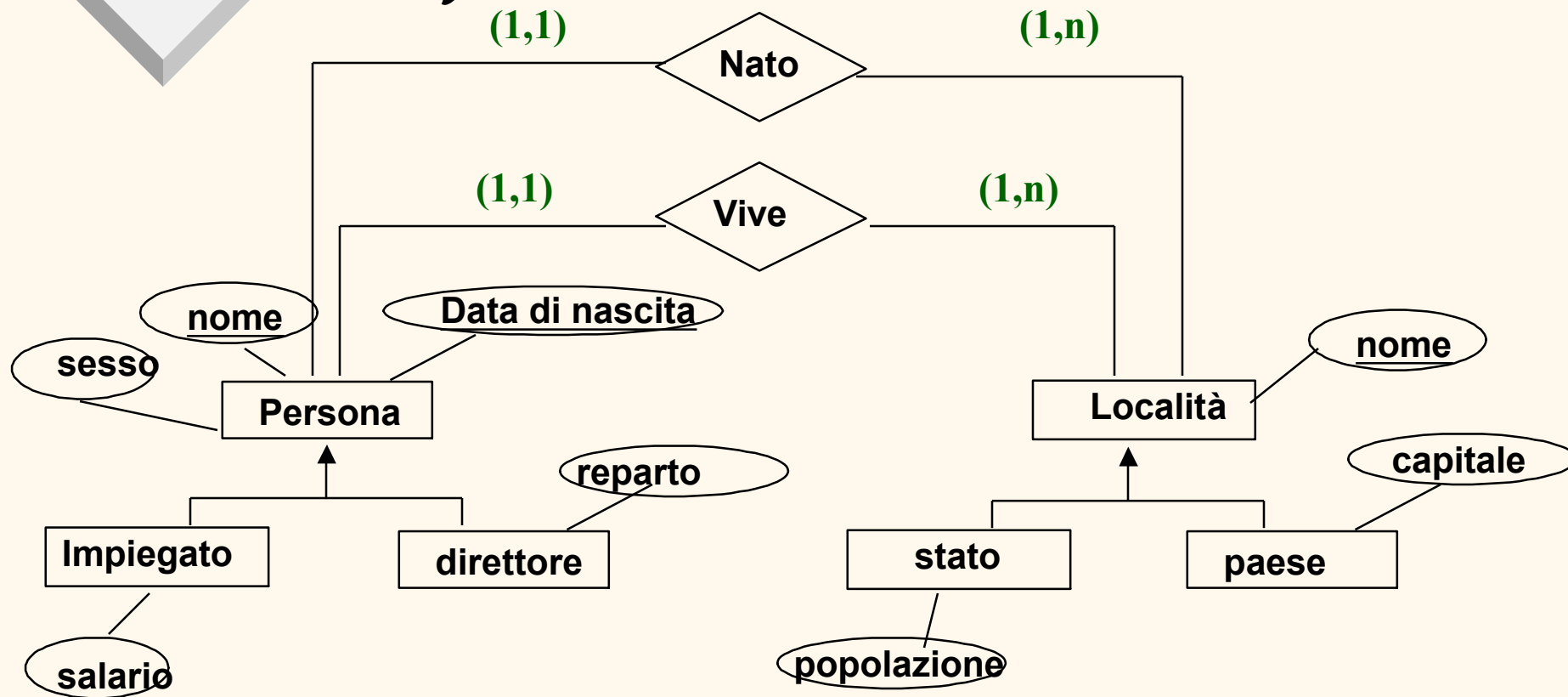
Ulteriore raffinamento



Raffinamento finale



Schema finale



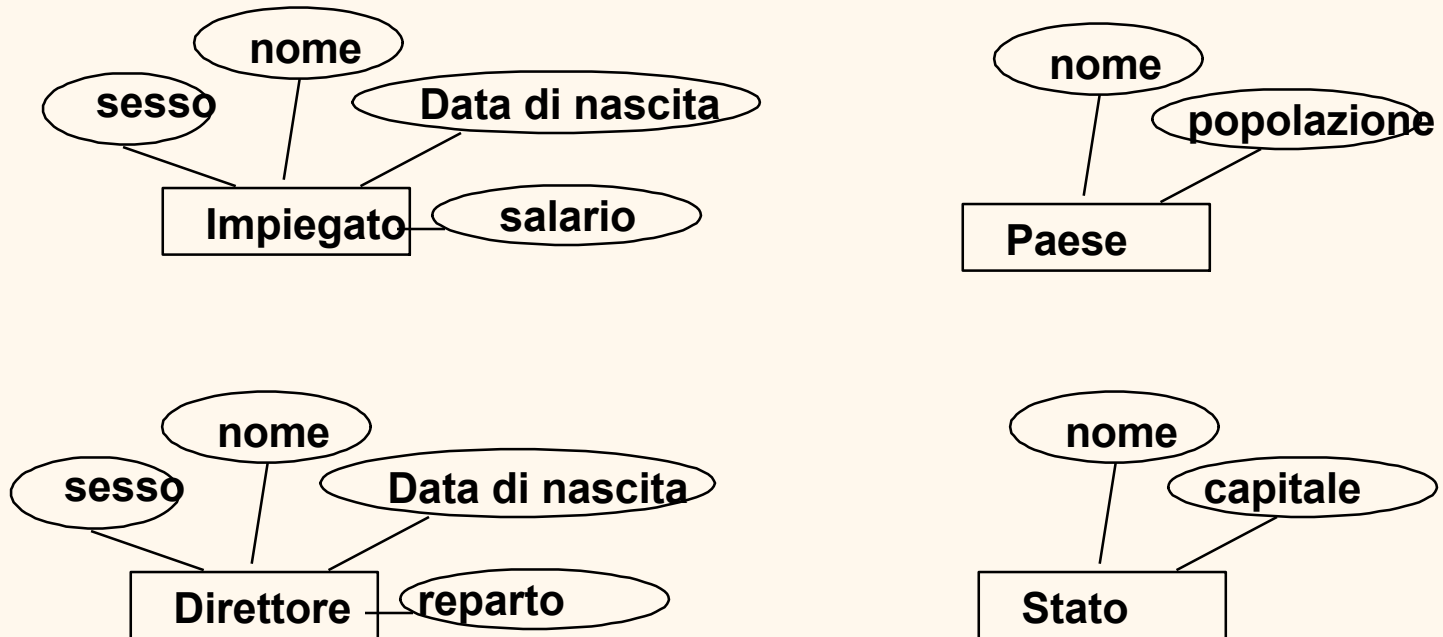


Bottom-up

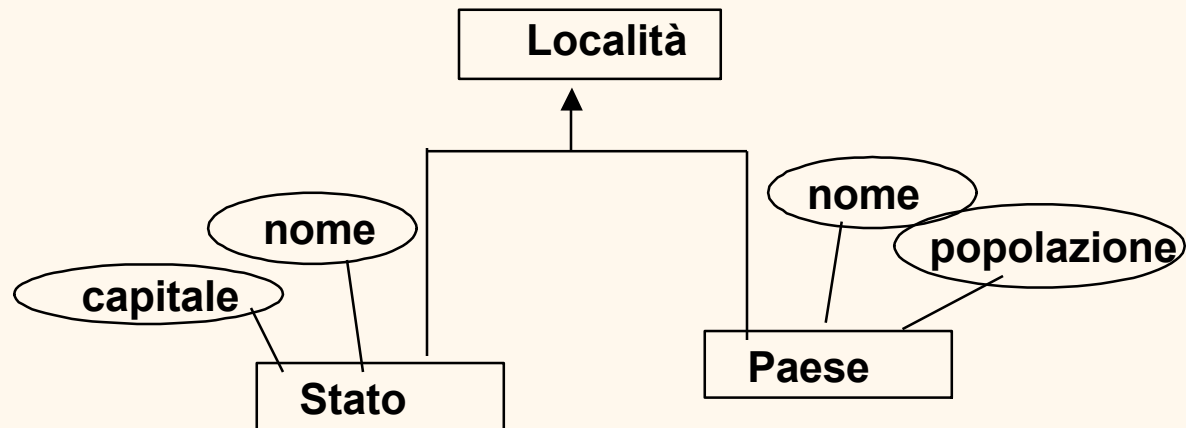
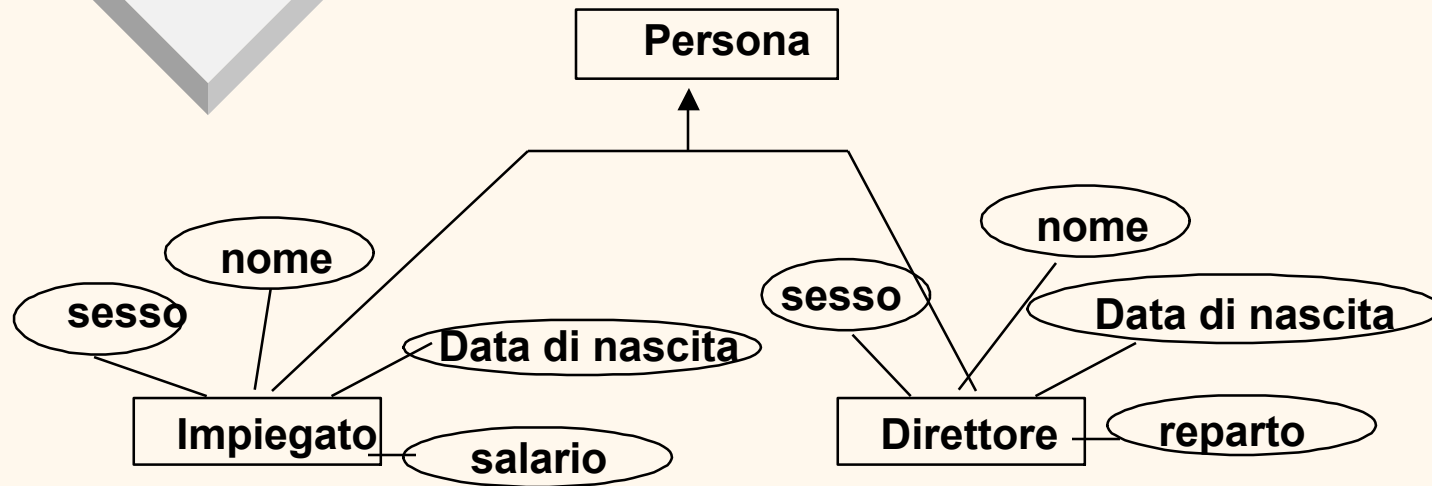
Scelta degli attributi

- Nome dell'impiegato
- Data di nascita dell'impiegato
- Sesso dell'impiegato
- Salario dell'impiegato
- Nome del direttore
- Data di nascita del direttore
- Reparto del direttore
- Sesso del direttore
- Nome del paese
- Capitale del paese
- Nome dello stato
- Popolazione dello stato

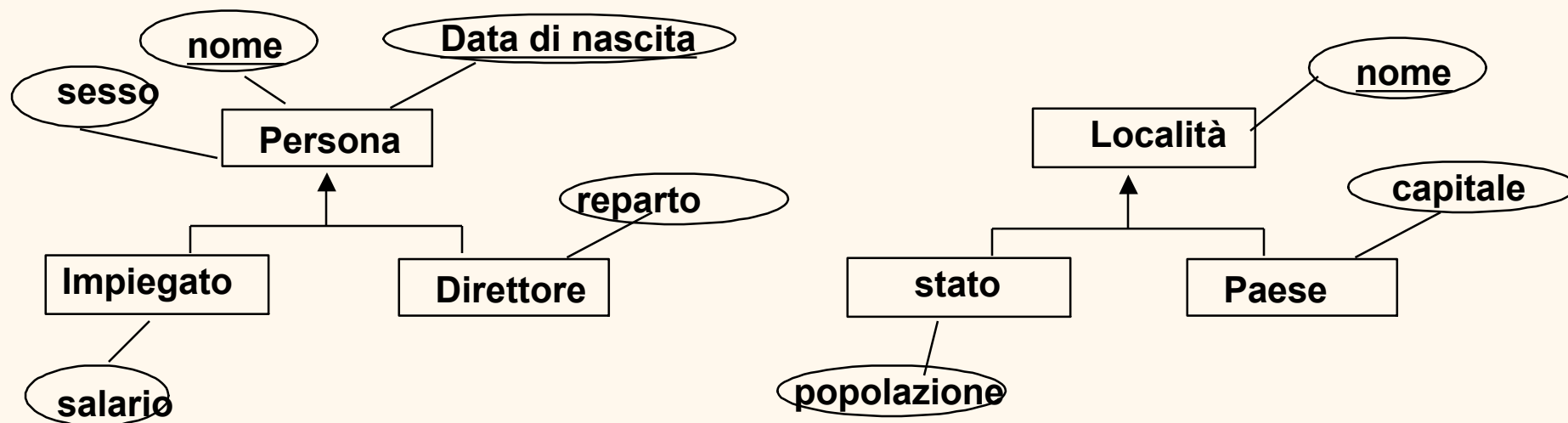
Scelta delle entità



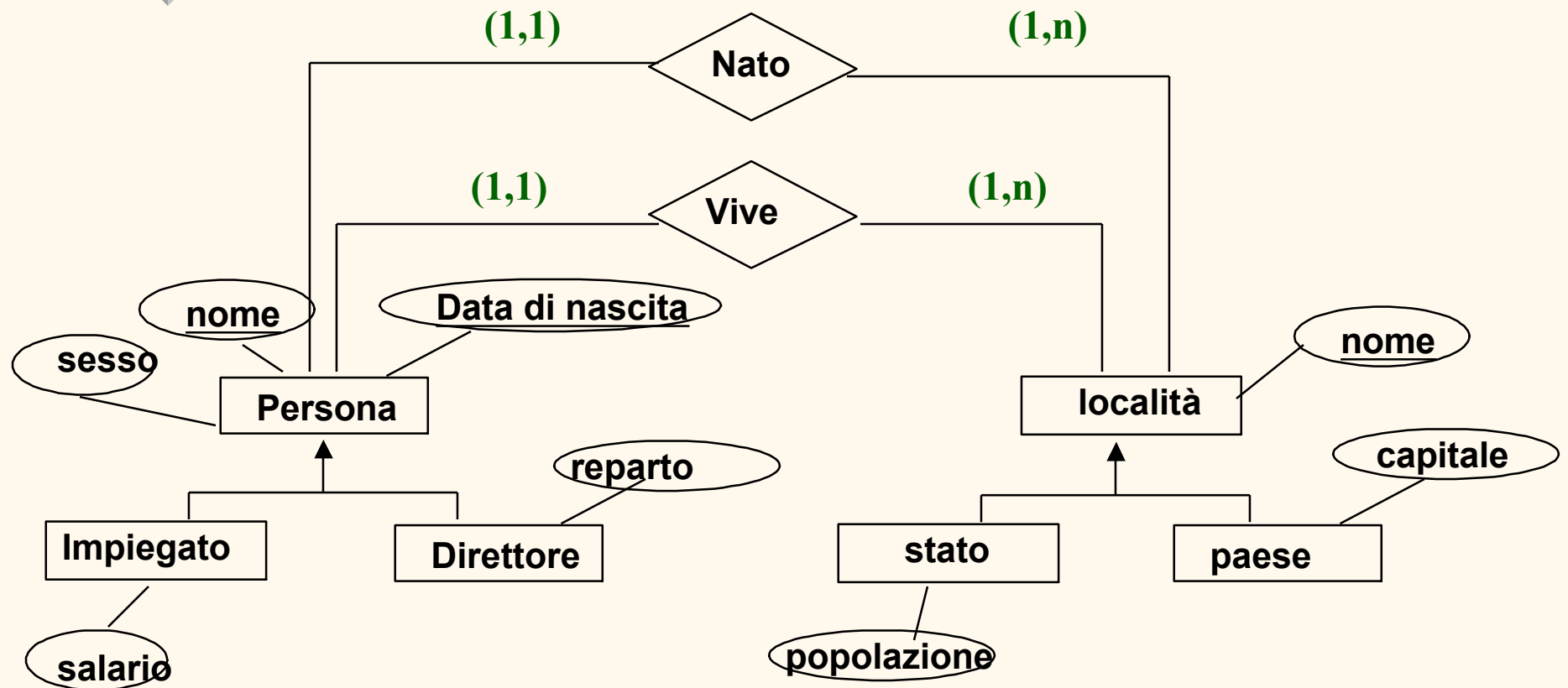
ISA



Ristrutturazione degli attributi



Relazioni





Da concettuale a logico

- ❖ Traduzione di uno schema concettuale (E-R) in uno schema (relazionale) logico
- ❖ Fare attenzione ai vincoli di integrità!
- ❖ La prima ottimizzazione si basa sulla frequenza degli accessi alle entità e alle relazioni
- ❖ Due fasi:
 - ristrutturazione dello schema concettuale e prima ottimizzazione
 - traduzione e ulteriore ottimizzazione (basata sul modello logico)



Ristrutturazione

- ❖ Analisi delle ridondanze
- ❖ Traduzione delle gerarchie IS-A
- ❖ Partizionamento delle entità
- ❖ Fusione delle entità
- ❖ Scelta delle chiavi primarie

Ridondanze

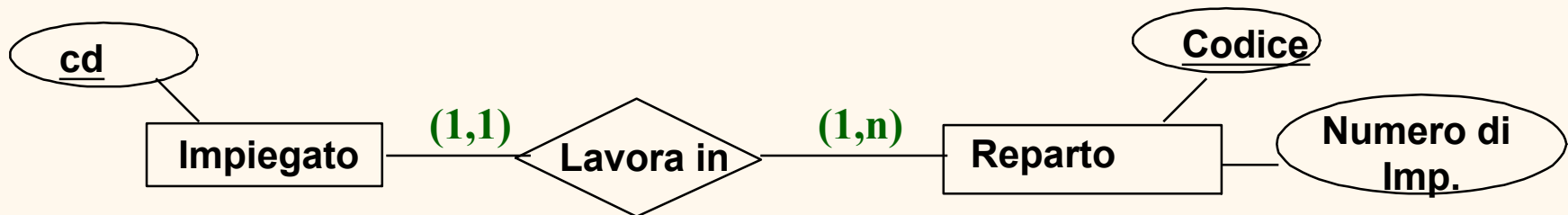
❖ Pro

- Maggiore velocità di esecuzione di alcune interrogazioni

❖ Contro

- Aumento in memoria
- Alto costo del mantenimento dell'integrità

Es.:





Transazioni

1. Per ciascun reparto, visualizzare il corrispondente numero di impiegati (ogni giorno)
2. Assegnare ogni nuovo impiegato a un reparto (ogni giorno)
3. Spostare un impiegato da un reparto a un altro (ogni 3 giorni)

Dimensione di entità e relazioni:

Impiegati – 100

Lavora in – 100

Reparto – 100



Modifiche di tempo e memoria con ridondanza

Operazione 1: 10 accessi a Reparto; -100 accessi a Impiegato

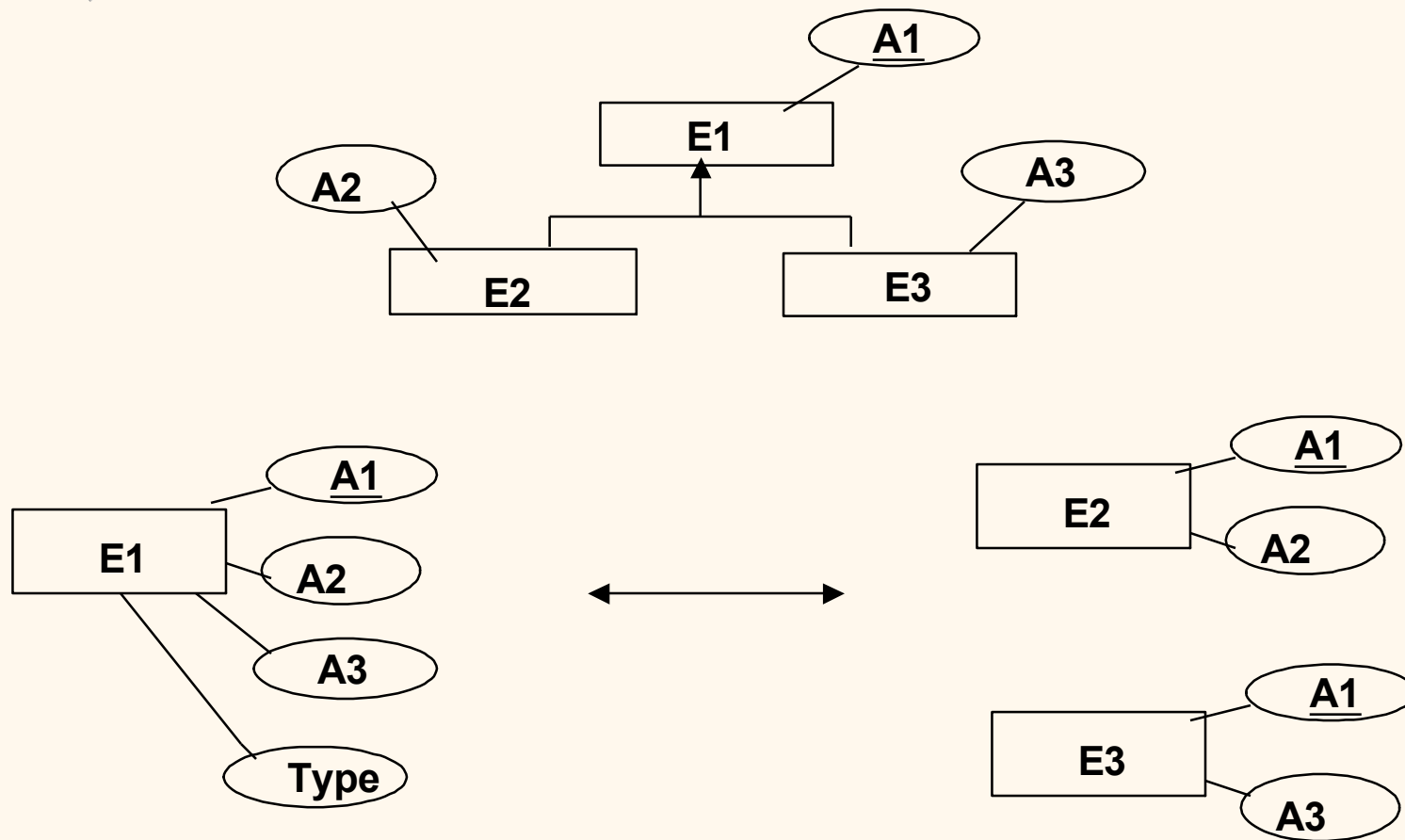
Operazione 2: 1 accesso a Impiegato; +1 accesso a Reparto

Operazione 3: 1 accesso a Impiegato; +2 accessi a Reparto

Incremento di memoria: 10

Accessi (ogni 3 giorni): $-300 + 3 + 2$

Traduzione in gerarchia IS-A



Due estremi



Fusione contro divisione

❖ Fusione

- Grande occupazione di memoria quando le sotto-entità hanno molti attributi
- Inutile se l'entità principale ha pochi attributi e/o relazioni (utile nel caso opposto)
- Valori *null*

❖ Divisione

- Nessuno spreco di memoria se non c'è sovrapposizione tra le sotto-entità
- Inutile se ci sono molti accessi di tipo *join* alle sotto-entità
- Necessità di procedure per il controllo dell'integrità

Confronto tra gli accessi delle transazioni

Confrontare $Z(E)$ (ossia il costo delle transazioni che accedono l'entità principale) con $\text{SUM}(Z(E_i))$ (cioè la somma dei pesi delle transazioni che accedono alle sotto-entità)

Date le transazioni P_1, \dots, P_n che accedono a una entità T :

$$Z(T) = \sum_{i=1, \dots, n} F(P_i) * S(P_i)$$

Dove F è la frequenza e S la priorità di P_i

Vincoli di integrità

❖ Fusione

- $E1 = E$ e $TIPO = E1$
- $E2 = E$ e $TIPO = E2$

❖ Divisione

- $E = E1 \sqcup E2$
- $R = R1 \sqcup R2$ (per ogni R cui partecipa E)

❖ $E1 \sqcap E2 = \square$ (se non sono permesse sovrapposizioni)



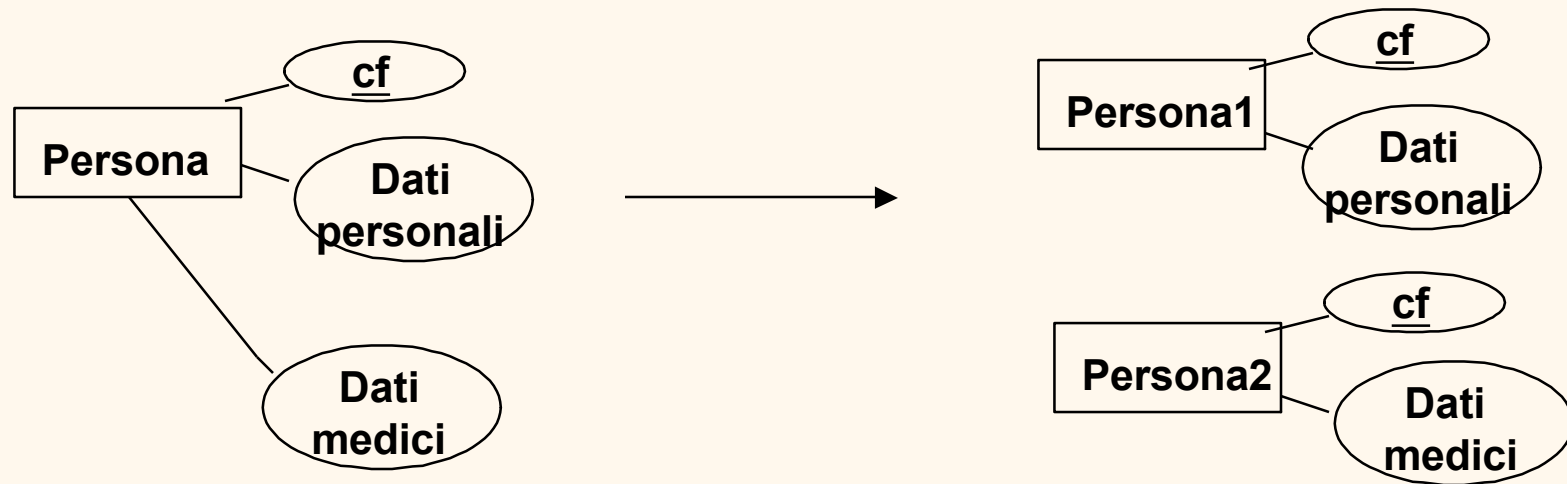
Altre soluzioni

- ❖ Traduzione mista (parte delle entità della gerarchia sono fuse, altre vengono separate)
- ❖ Ridondanza (vengono lasciate tutte le entità)
 - Operazioni di aggiornamento molto complesse (controllo dell'integrità)
 - Spreco di memoria

Partizionamento e fusione delle entità

- Partizionamento: le entità vengono separate in base agli accessi delle transazioni

Es.



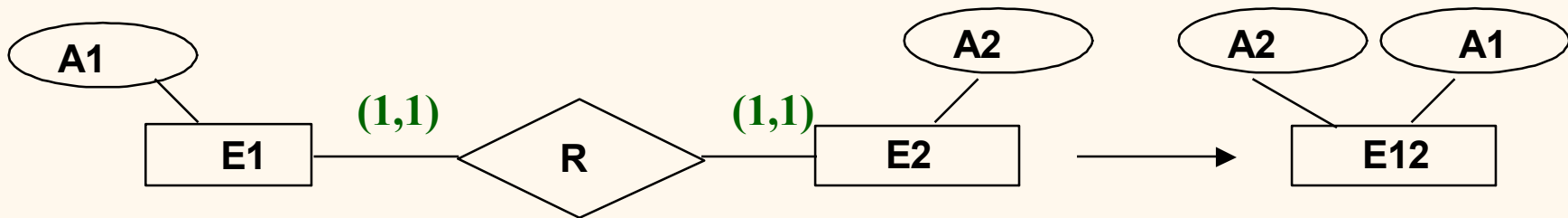
Vincolo di integrità: $\text{Persona1}[\underline{CF}] = \text{Persona2}[\underline{CF}]$

Nota: quando si divide, ricordare che le proprietà delle entità non sono solo attributi, ma anche relazioni e gerarchie


Fusione

❖ Le entità cui si accede spesso possono essere fuse

Es.:



Vincoli di integrità : $E12[A1] = E1[A1]$
 $E12[A2] = E2[A2]$

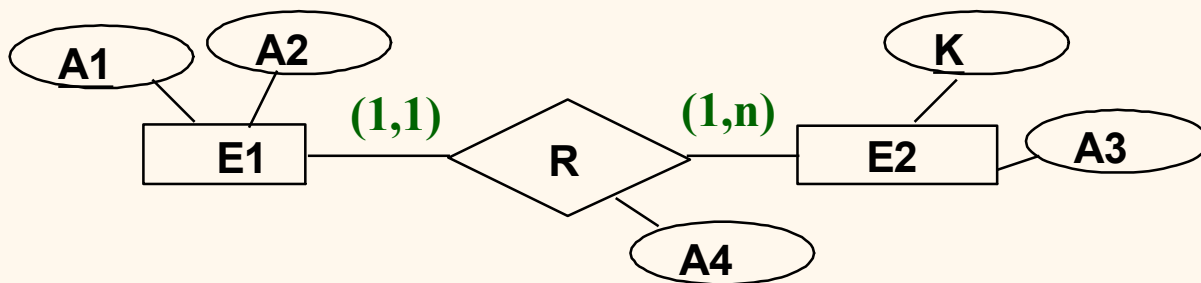


Chiavi primarie

- ❖ Per ciascuna entità si deve scegliere una chiave primaria
- ❖ La chiave primaria di una relazione è data dall'unione delle chiavi primarie delle entità che vi partecipano

Traduzione

- ❖ Ciascuna entità diventa una relazione, con la chiave primaria corrispondente
- ❖ Ciascuna relazione (m:n) diventa una relazione, con la chiave primaria corrispondente
- ❖ Ciascuna relazione (1:n) e (1:1) viene tradotta tramite inclusione della chiave esterna



$R_{E1}(\underline{A1}, A2, K, A4)$

$R_{E2}(\underline{K}, A3)$

$R_{E2}[K] \ \& \ \underline{R}_{E1}[K]$

$R_{E1}[K] \ \& \ \underline{R}_{E2}[K]$

- ❖ Le relazioni con cardinalità minima uguale a 0 possono originare valori null



Note

- ❖ Tutti i vincoli di integrità che si originano durante la progettazione logica e/o la traduzione devono essere importati nello schema relazionale (principalmente come vincoli di inclusione di chiave)
- ❖ Tutte le relazioni devono essere in 1NF (vedi normalizzazione), cioè tutti gli attributi sono atomici e monovalore
- ❖ Le relazioni ottenute traducendo lo schema ER possono essere ulteriormente normalizzate (3NF o BCNF) (vedi normalizzazione)