Basi di Dati

Progettazione Logica (seconda parte)

Marco Maratea
Laurea in Informatica, DeMaCS, UNICAL

17 Ottobre 2025

Requisiti della base di dati

↓

Progettazione concettuale

↓

Schema concettuale

↓

Progettazione logica

↓

Schema logico

↓

Progettazione fisica

↓

Schema fisico

Obiettivo della progettazione logica

- "tradurre" lo schema concettuale in uno schema logico che rappresenti gli stessi dati in maniera corretta ed efficiente

Dati di ingresso e uscita

- Ingresso:
  - schema concettuale
  - informazioni sul carico applicativo
  - modello logico (per noi sarà il relazionale)
- Uscita:
  - schema logico
  - documentazione associata

Non si tratta di una pura e semplice traduzione

- alcuni aspetti non sono direttamente rappresentabili, quindi è organizzata in due fasi separate in cascata
- è necessario considerare le prestazioni

Carico applicativo

Schema concettuale E-R

Ristrutturazione dello schema E-R

Modello logico

Schema E-R ristrutturato

Traduzione nel modello logico

Schema logico

Ristrutturazione schema E-R

- Motivazioni:
  - semplificare la traduzione nel modello relazionale
  - "ottimizzare" le prestazioni
- Osservazione:
  - uno schema E-R ristrutturato non è (più) uno schema concettuale come lo abbiamo studiato, ma è uno schema E-R contenente solo entità, relationship ed attributi semplici

Prestazioni?

- Per ottimizzare il risultato abbiamo bisogno di analizzare le prestazioni a questo livello
- Ma:
  - le prestazioni non sono valutabili con precisione su uno schema concettuale!

Prestazioni, approssimate

- Consideriamo:
  - "indicatori" dei parametri che regolano le prestazioni
- spazio:
  - numero di occorrenze previste
- tempo:
  - numero di occorrenze (di entità e relationship) visitate durante un'operazione, divise in operazioni di scrittura (aggiungere una occorrenza o modificare il valore di attributo) e lettura

Attività della ristrutturazione

- Analisi delle ridondanze
- Eliminazione delle generalizzazioni
- Partizionamento/accorpamento di entità e relationship
- Scelta degli identificatori primari

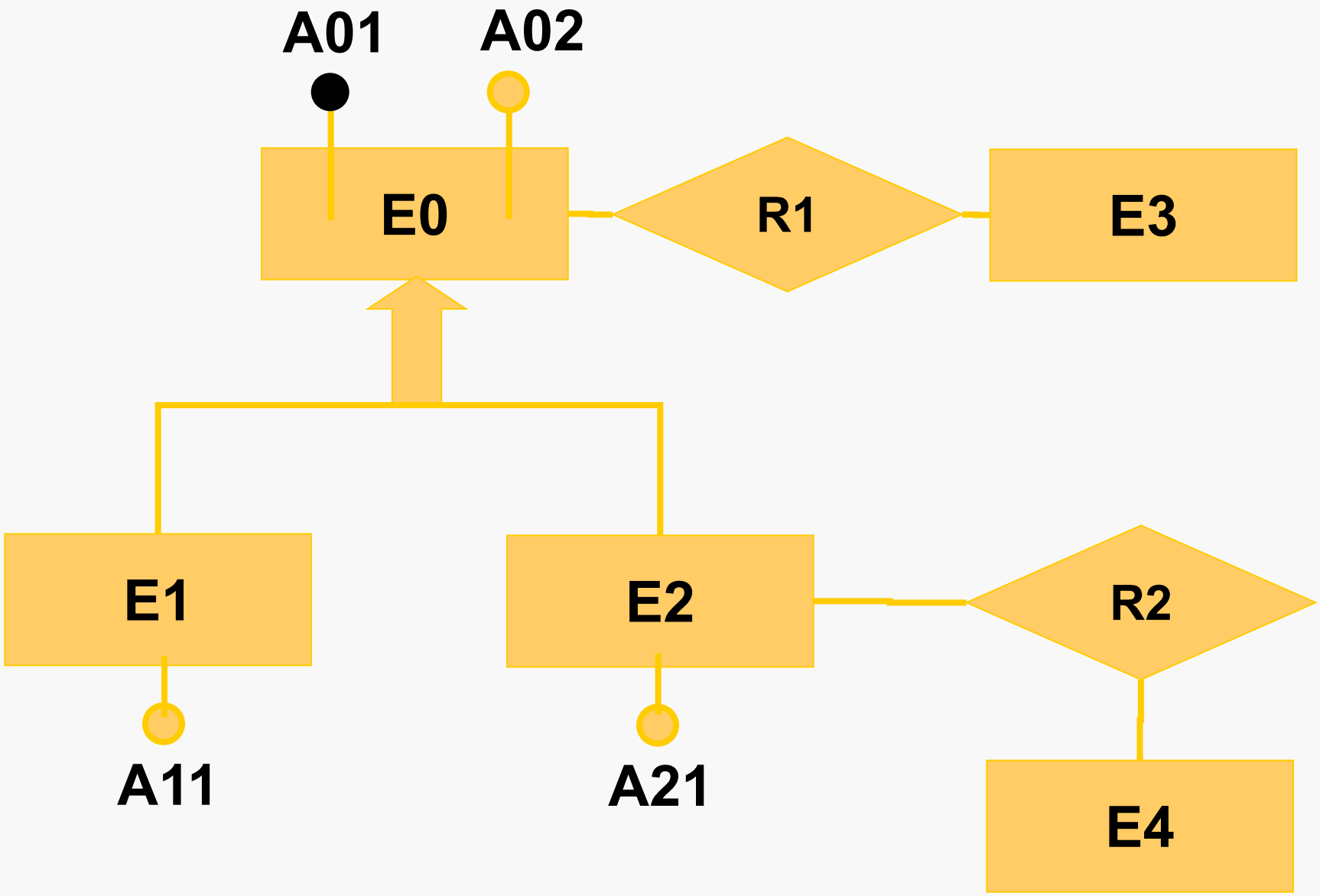
Eliminazione delle generalizzazioni

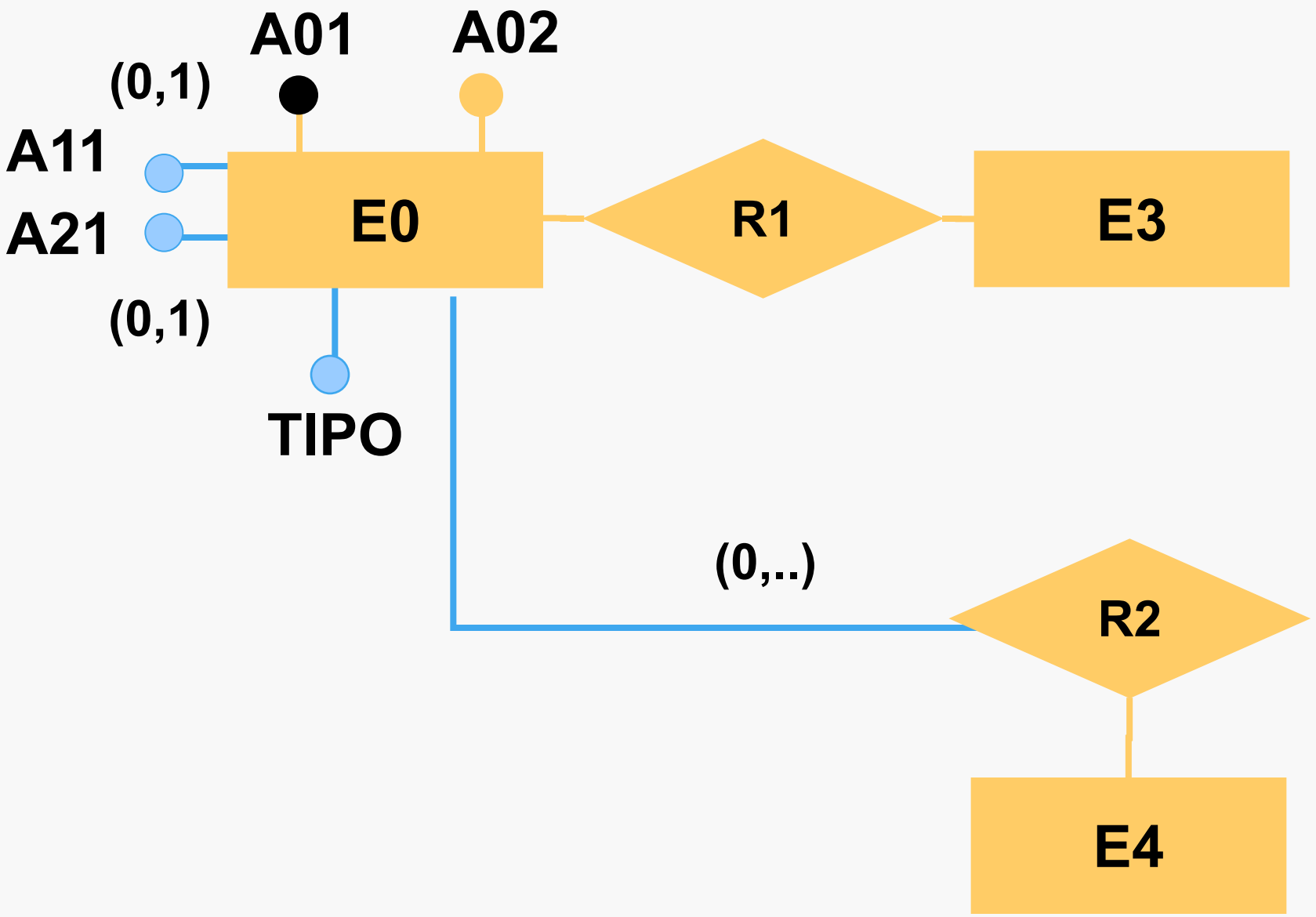
- il modello relazionale non può rappresentare direttamente le generalizzazioni
- Entità, relationship (ed attributi semplici) sono invece ’direttamente’ rappresentabili

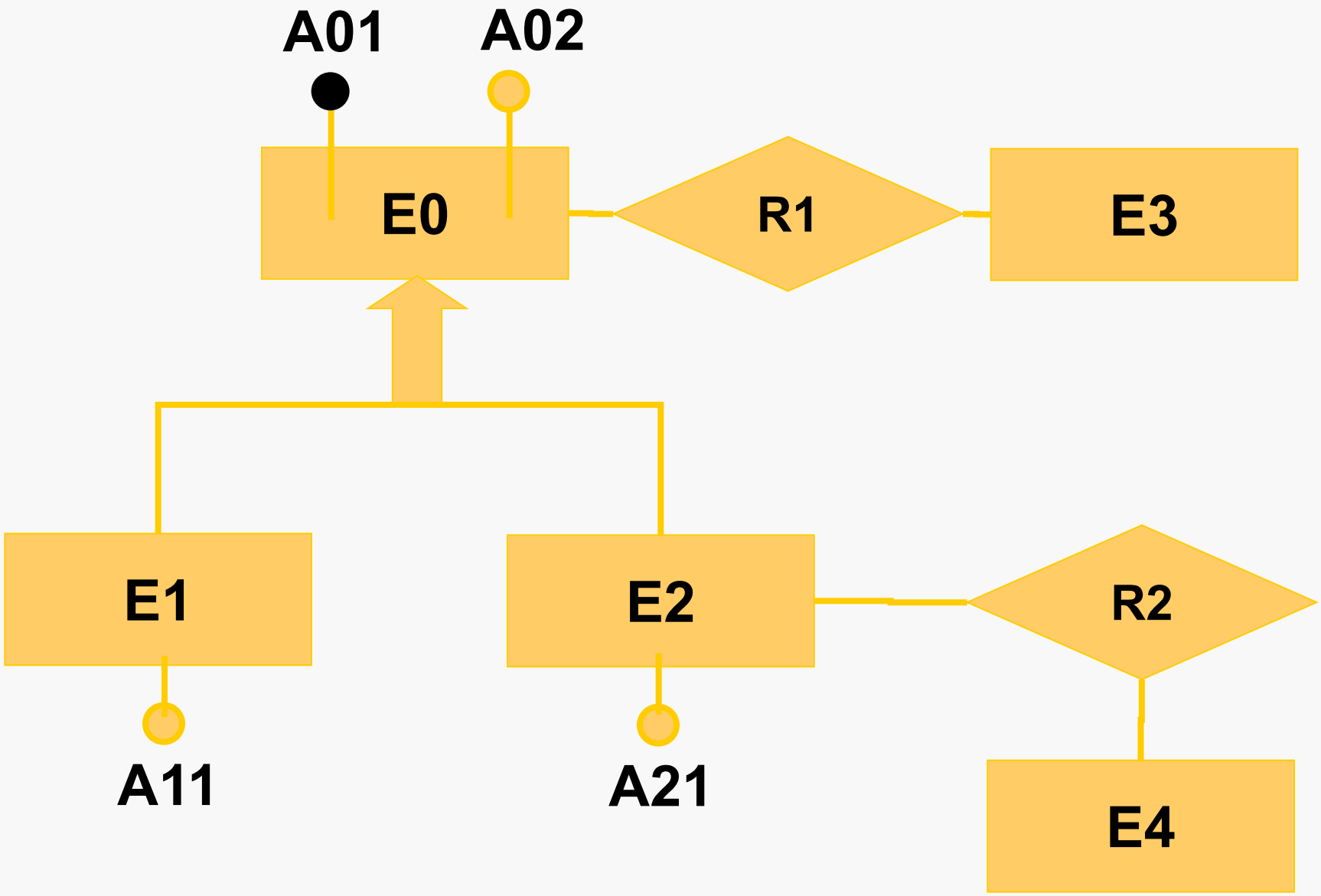
- si eliminano perciò le generalizzazioni, sostituendole con entità e relationship

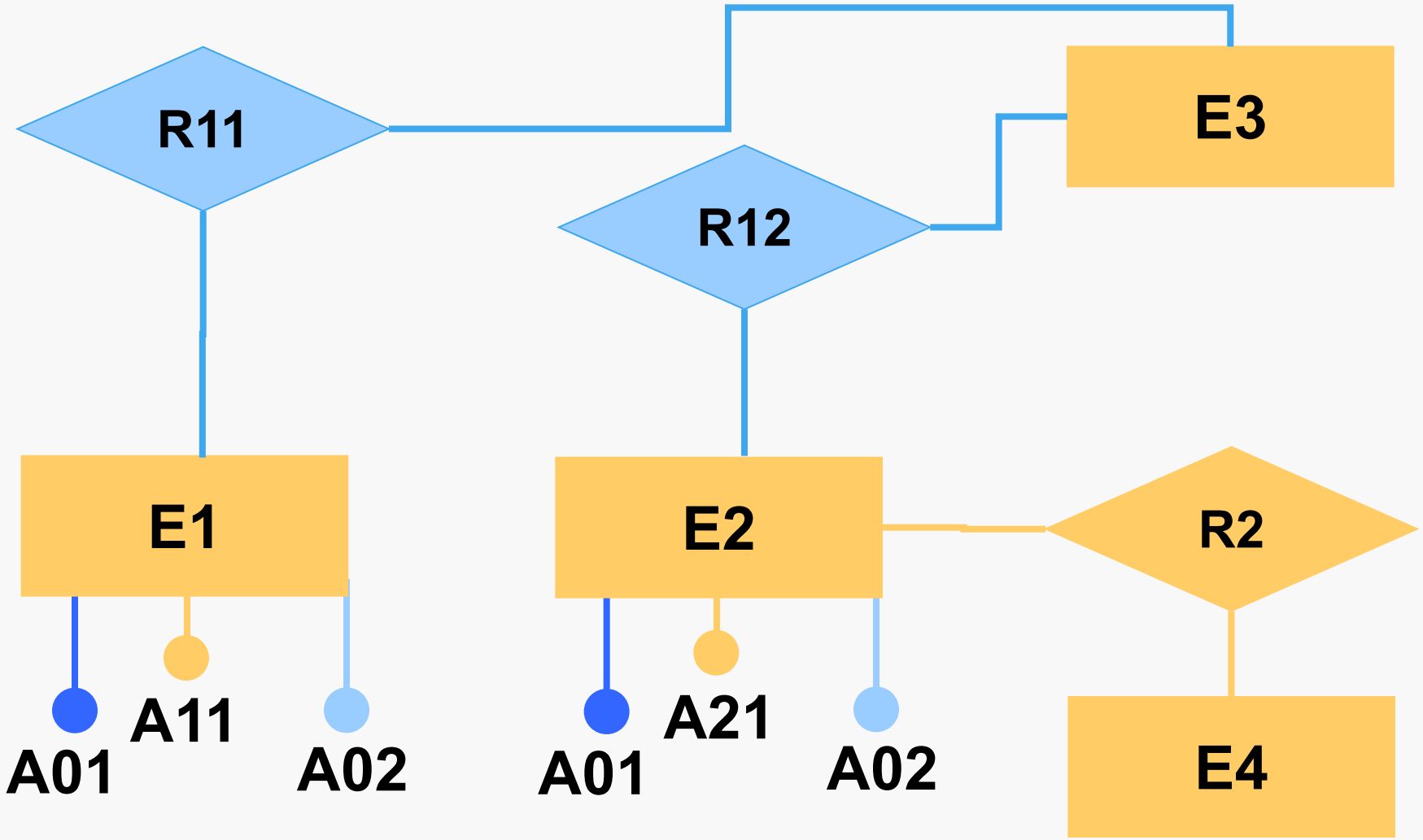
Tre possibilità

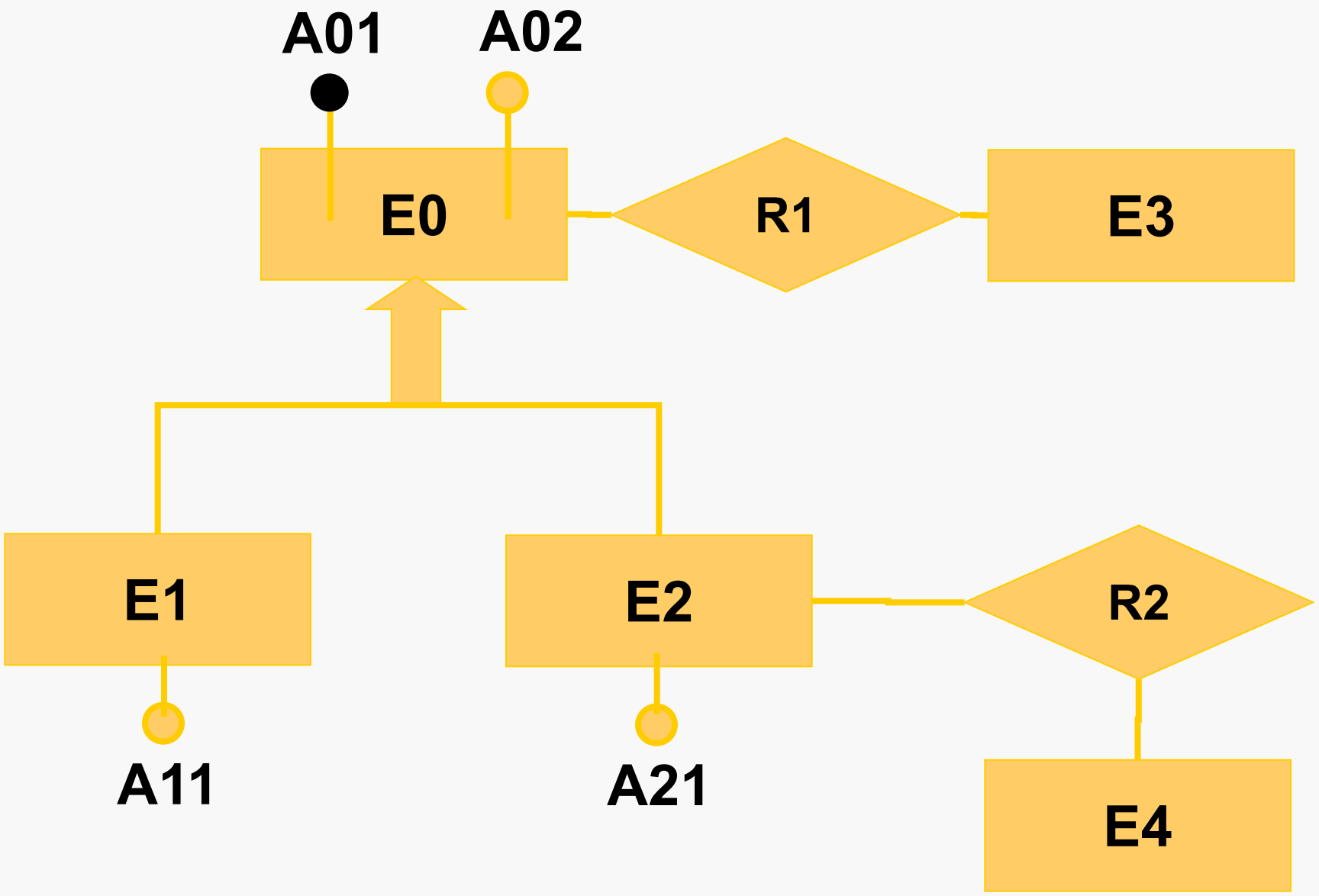
1. accorpamento delle figlie della generalizzazione nel genitore
2. accorpamento del genitore della generalizzazione nelle figlie
3. sostituzione della generalizzazione con relationship

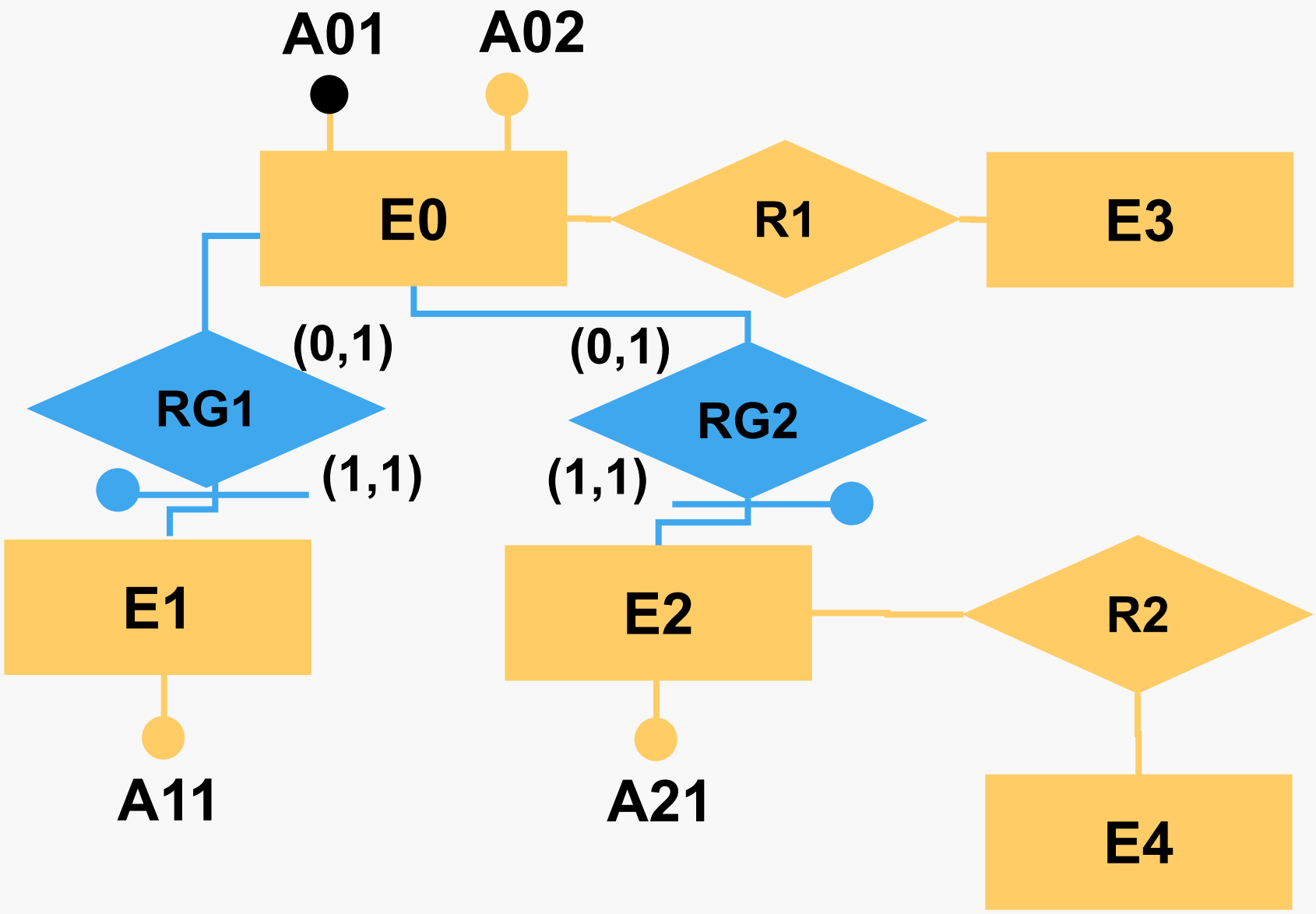










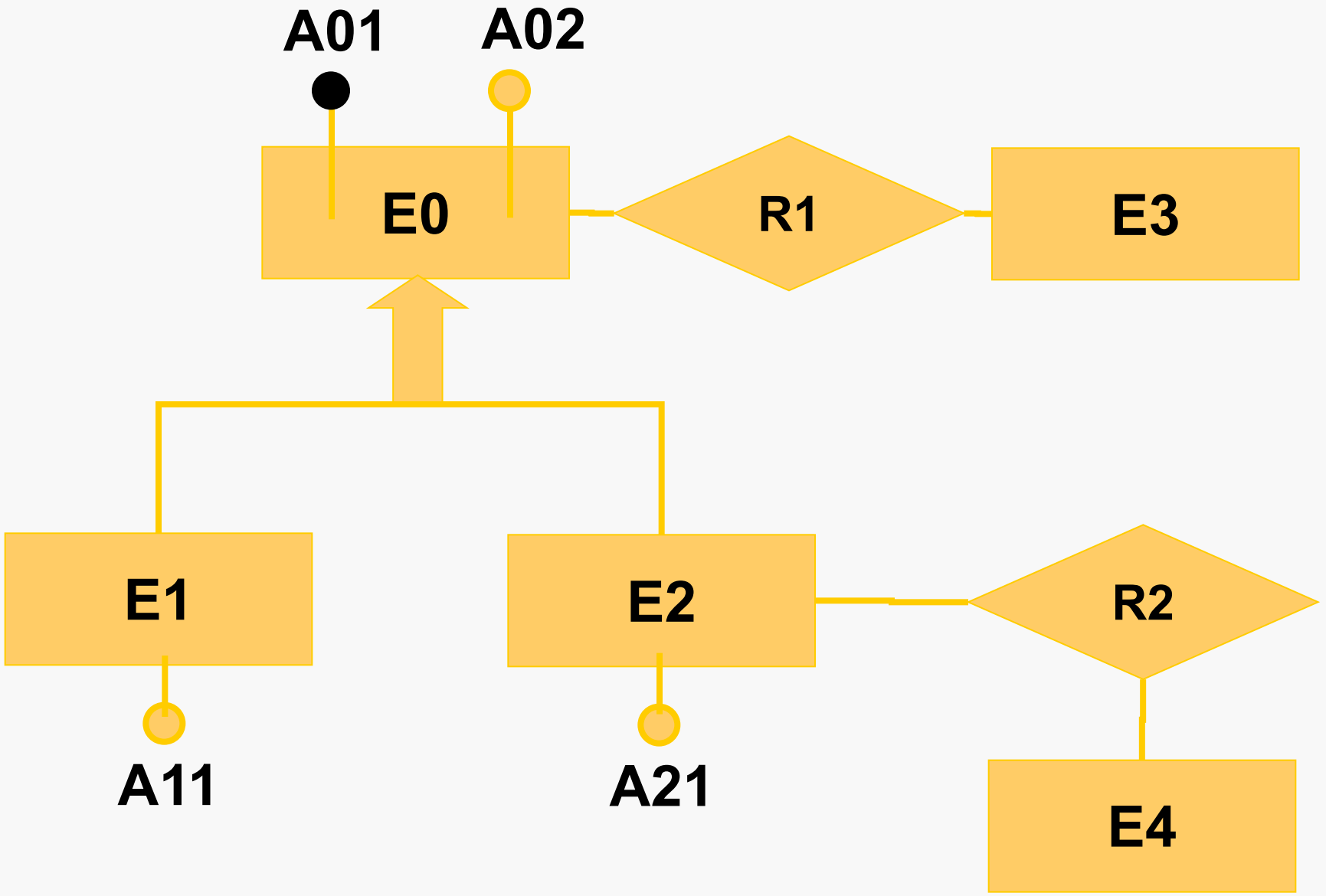


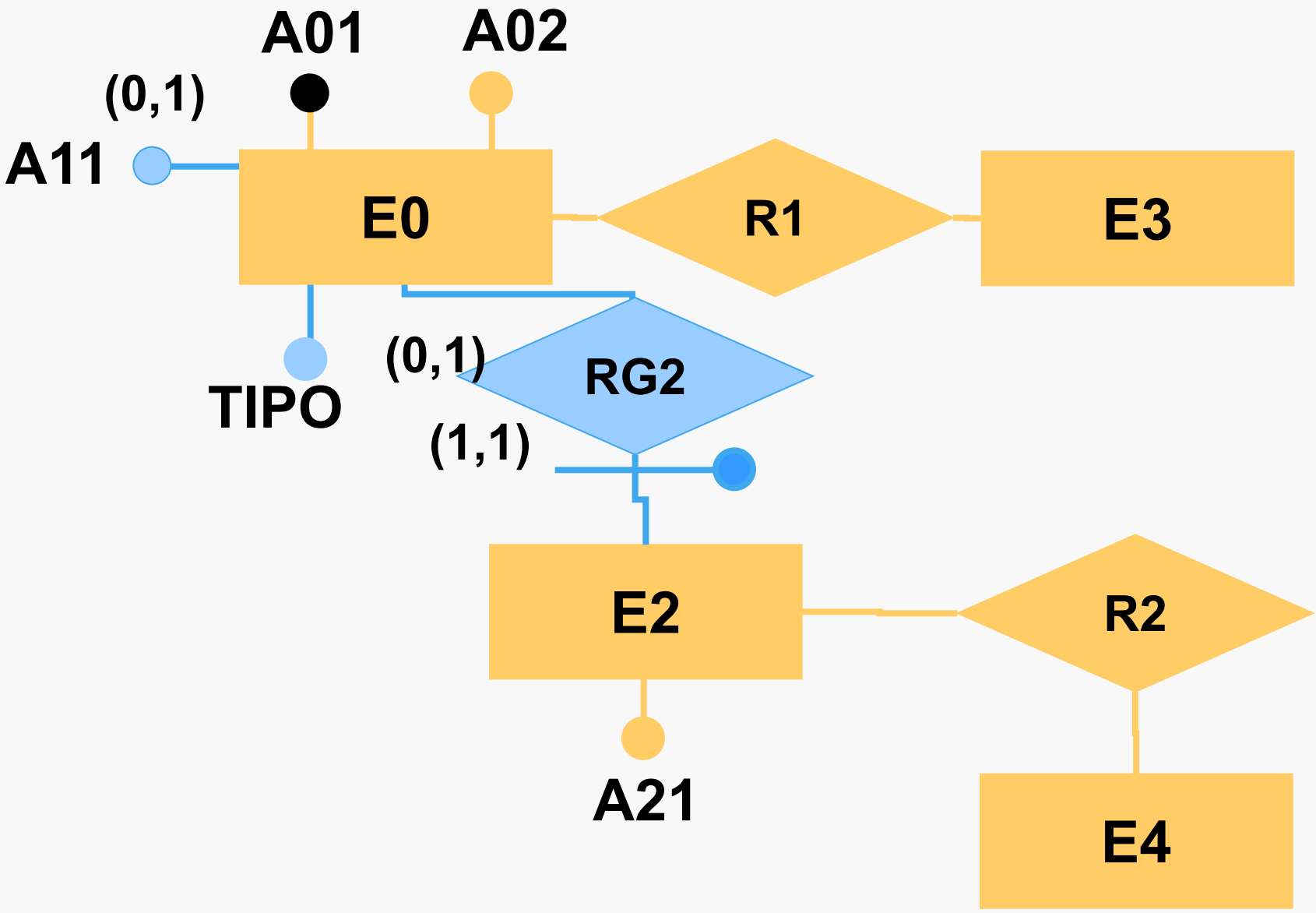
- la scelta fra le alternative con un metodo simile a quello visto per l'analisi delle ridondanze non sarebbe sufficiente, perché altri parametri entrano in gioco

E’ possibile seguire alcune semplici regole generali

1. conviene se gli accessi al padre e alle figlie sono contestuali
2. conviene se gli accessi alle figlie sono distinti, ed è usabile sono per generalizzazioni totali
3. conviene se gli accessi alle entità figlie sono separati dagli accessi al padre

(sono anche possibili soluzioni “ibride”, soprattutto in gerarchie a più livelli, ma di cui vediamo solo un esempio)





Attività della ristrutturazione

- Analisi delle ridondanze
- Eliminazione delle generalizzazioni
- Partizionamento/accorpamento di entità e relationship
- Scelta degli identificatori primari

- Ristrutturazioni effettuate per rendere più efficienti le operazioni in base a un semplice principio.

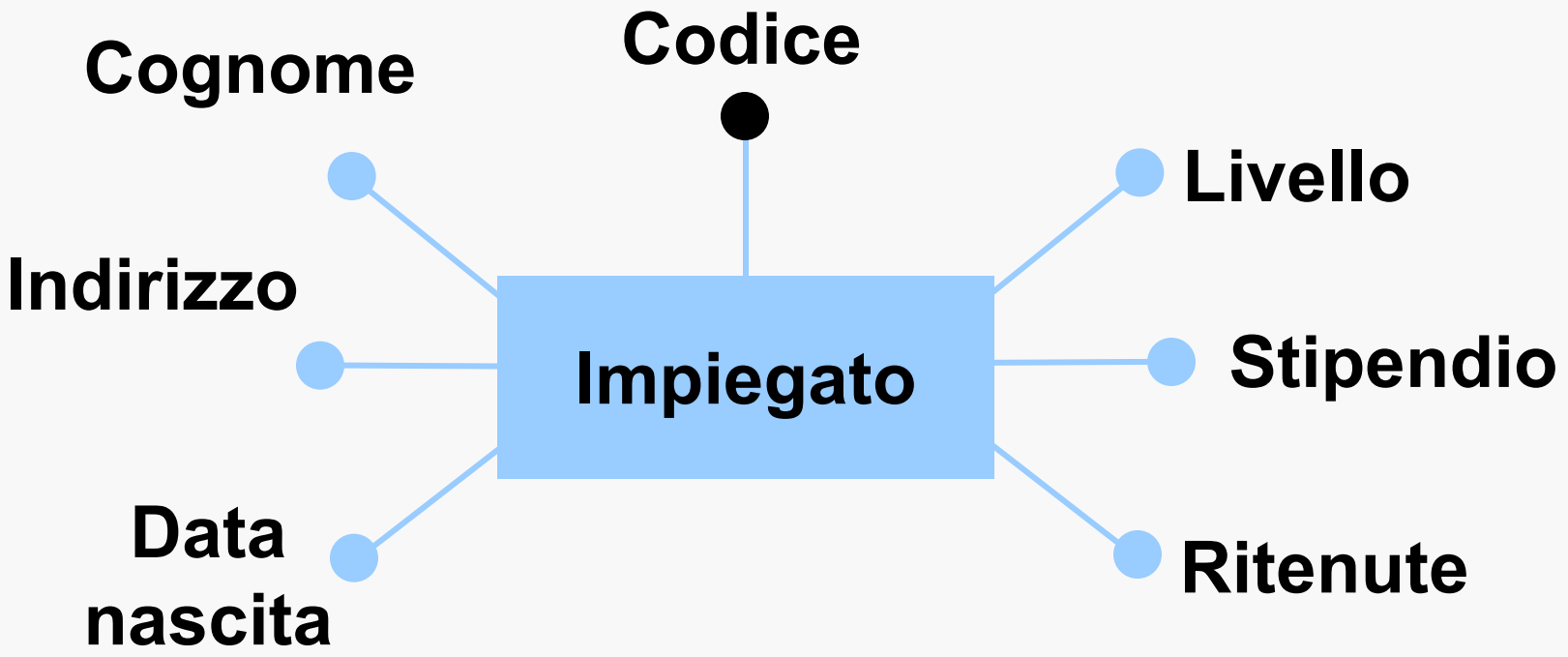
- Gli accessi si riducono:
  - separando attributi di un concetto che vengono acceduti separatamente
  - raggruppando attributi di concetti diversi acceduti insieme

Solo una ristrutturazione è obbligatoria (*)

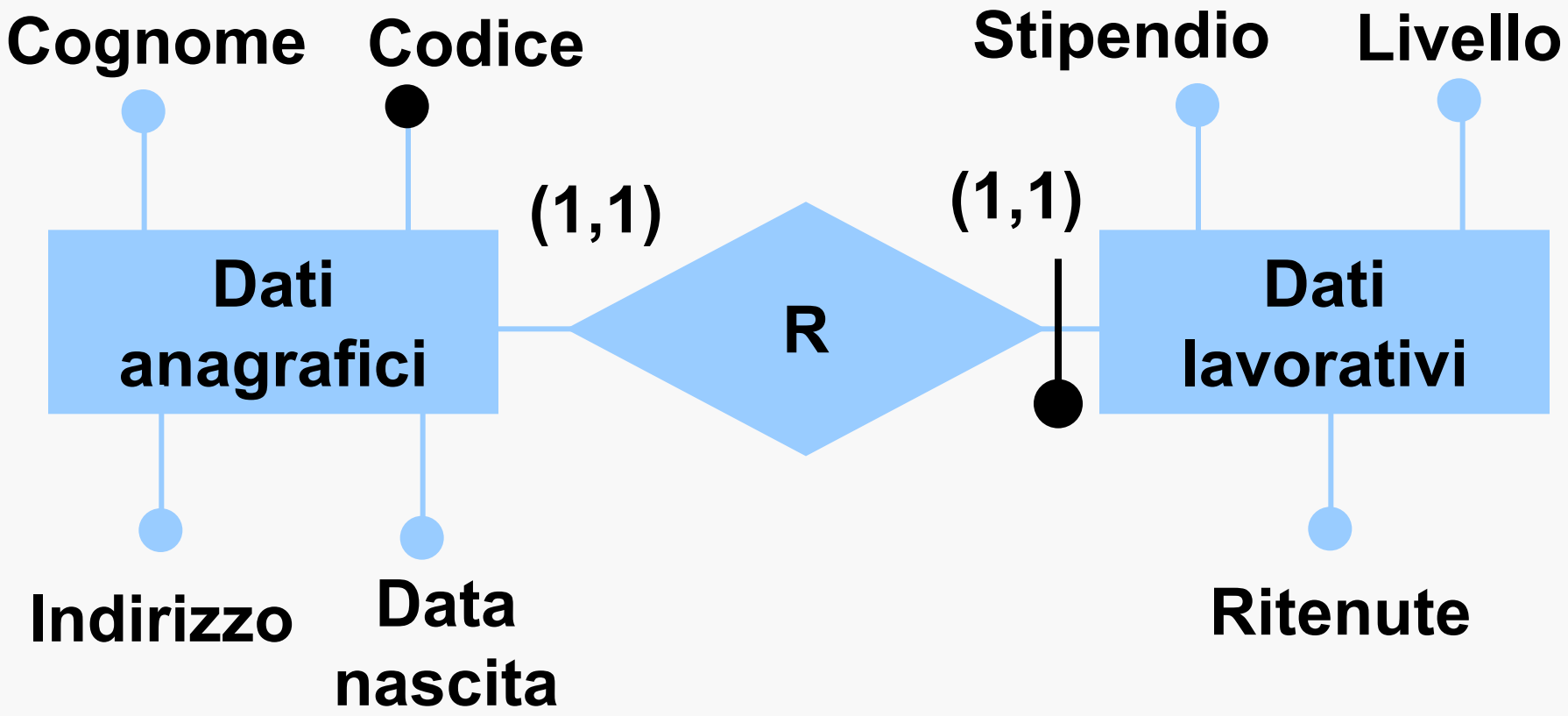
Ristrutturazioni, casi principali

- partizionamento verticale di entità
- partizionamento orizzontale di relationship
- eliminazione di attributi multivalore (*)
- accorpamento di entità/ relationship

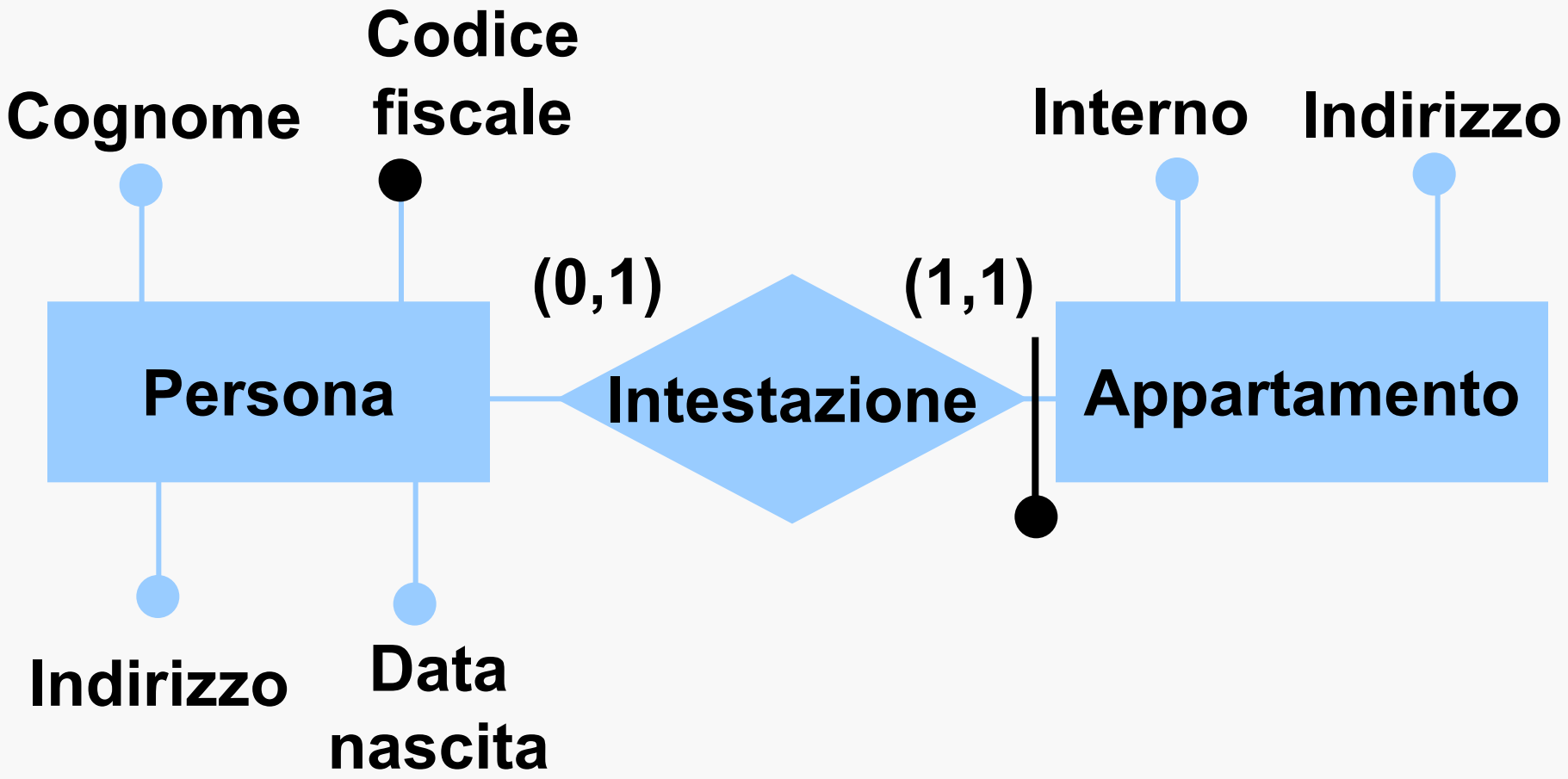
Partizionamento verticale di entità: schema di partenza



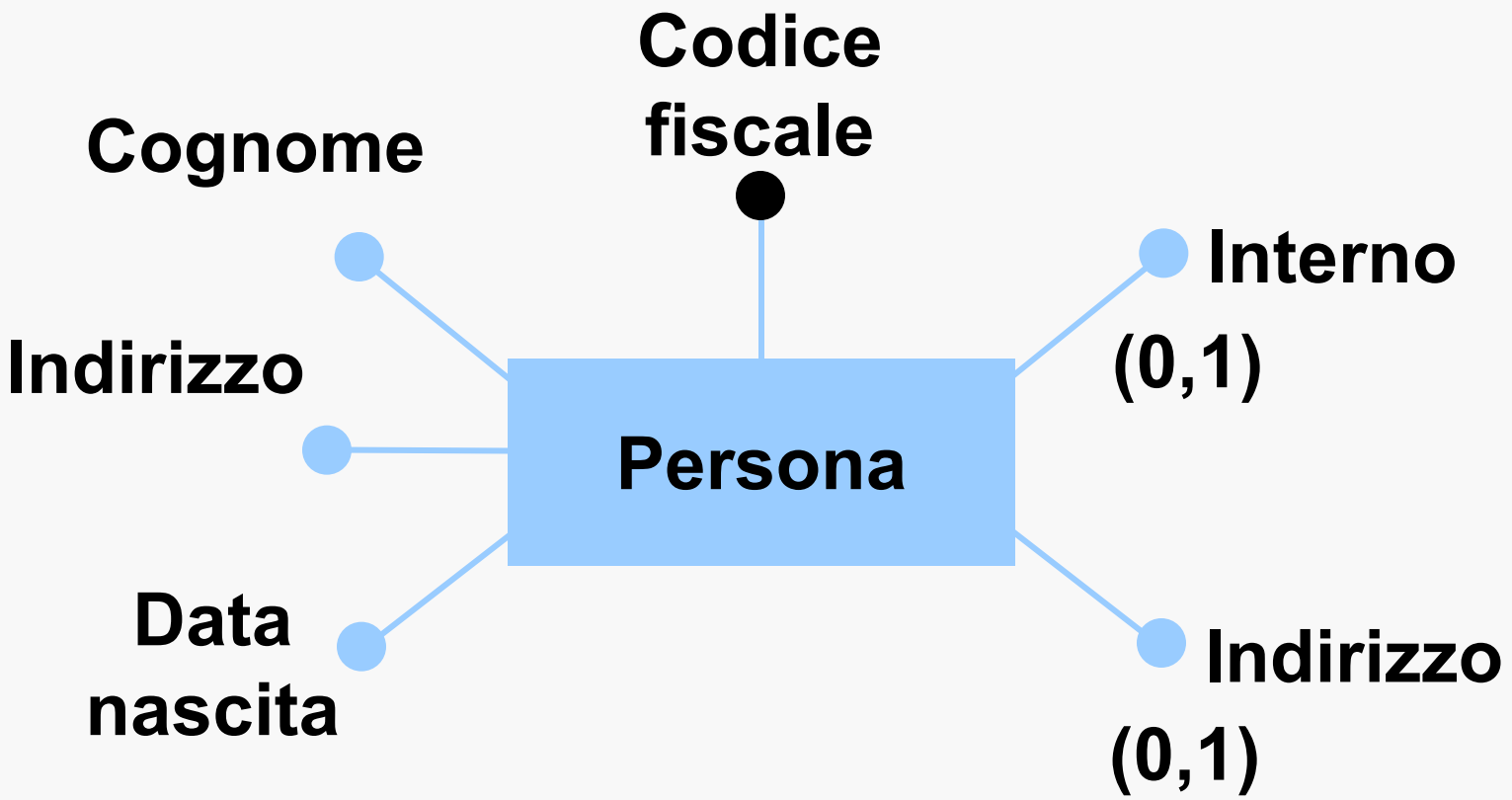
Partizionamento verticale di entità: schema ristrutturato



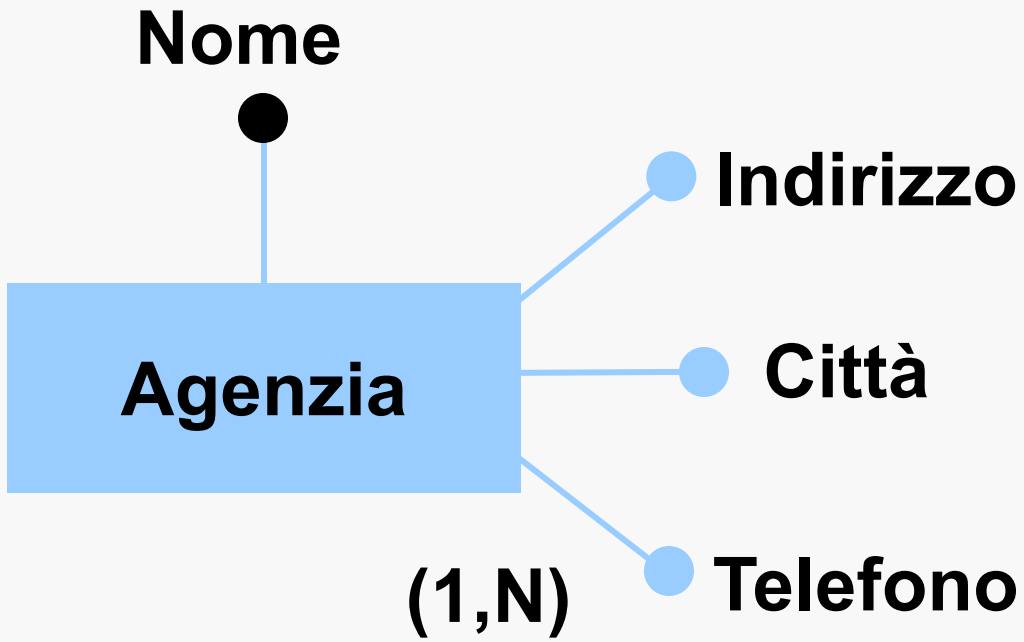
Accorpamento di entità: schema di partenza



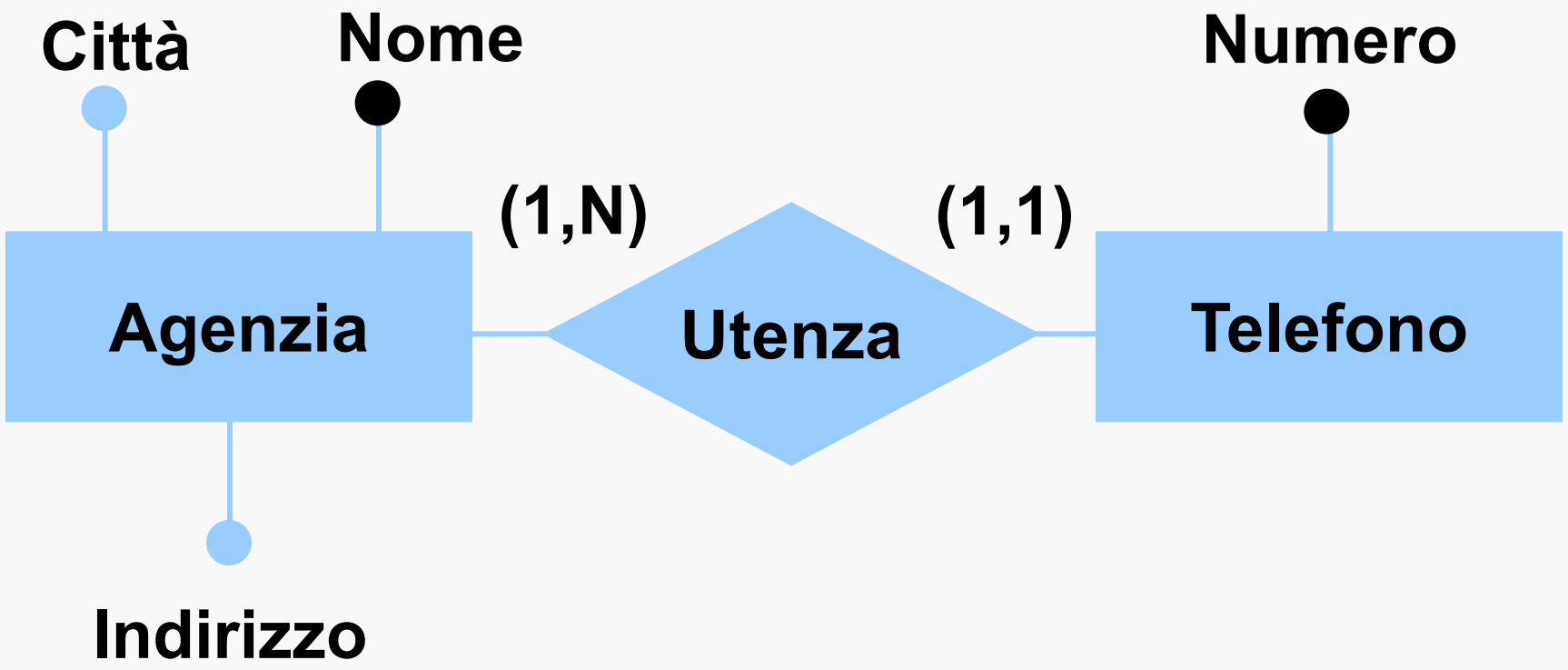
Accorpamento di entità: schema ristrutturato



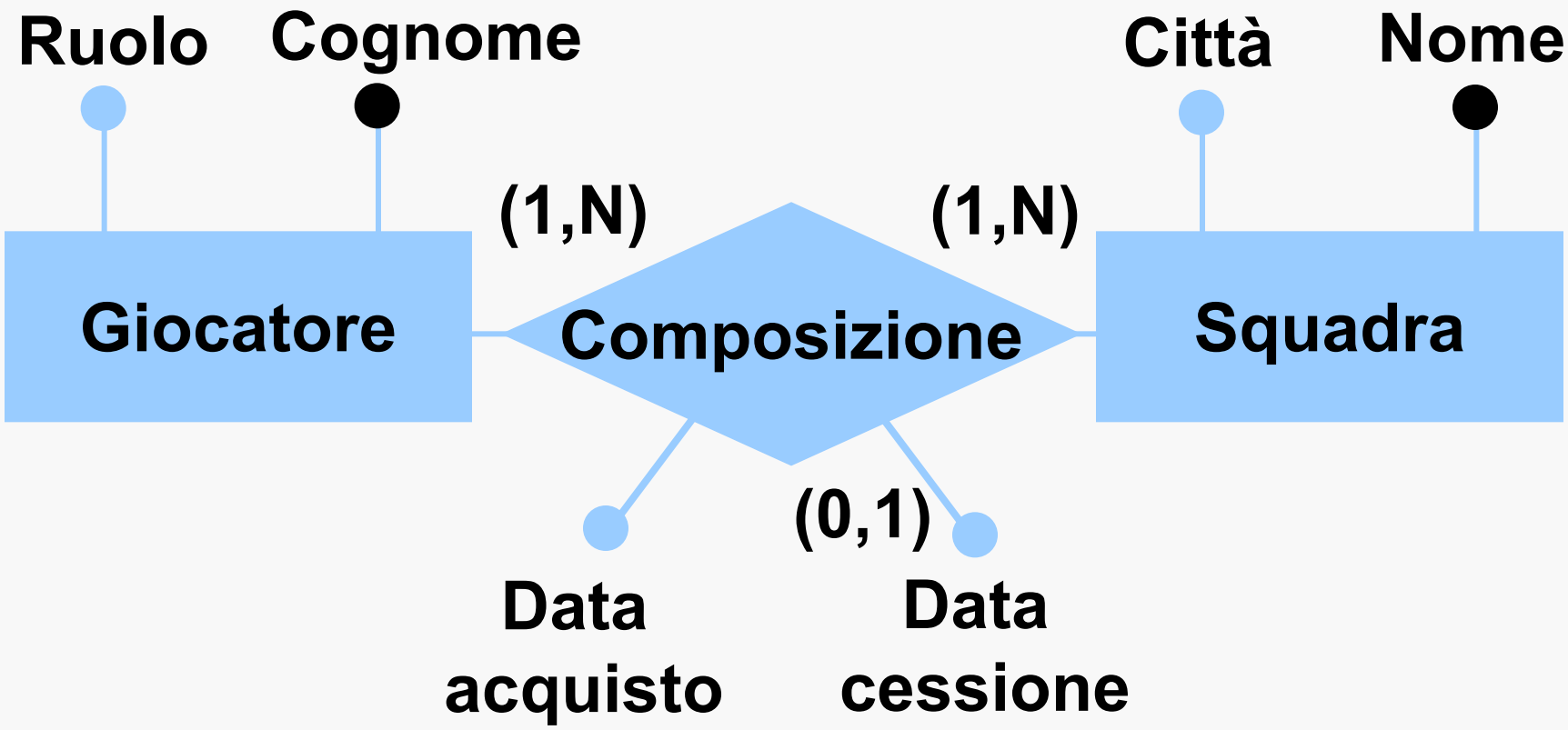
Eliminazione di attibuti multivalore (*): schema di partenza



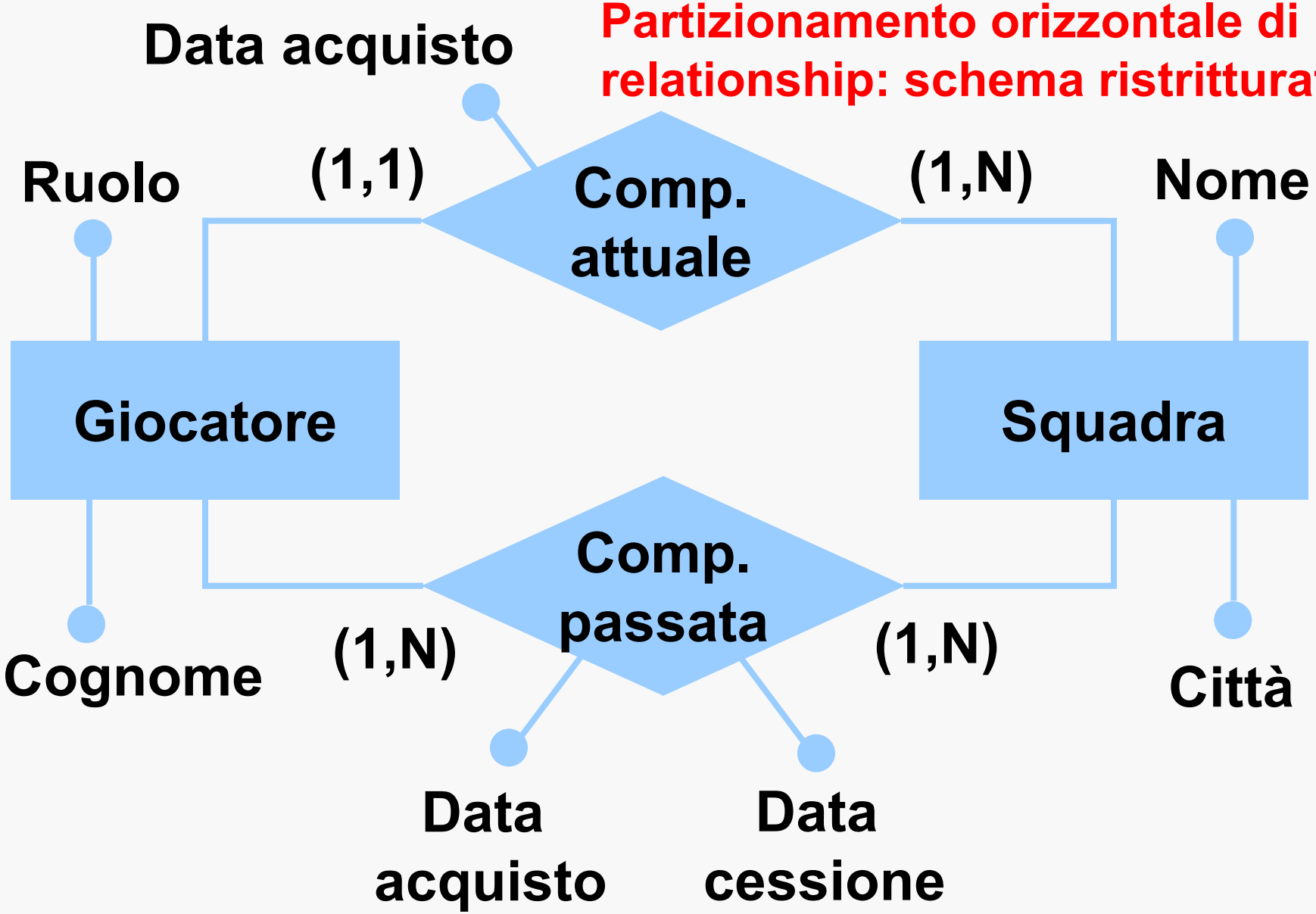
Eliminazione di attibuti multivalore (*): schema ristrutturato



Partizionamento orizzontale di relationship: schema di partenza



Partizionamento orizzontale di relationship: schema ristrutturato



Attività della ristrutturazione

- Analisi delle ridondanze
- Eliminazione delle generalizzazioni
- Partizionamento/accorpamento di entità e relationship
- Scelta degli identificatori primari

Scelta degli identificatori primari

- operazione indispensabile per la traduzione nel modello relazionale, perché diventeranno chiavi primarie

- Criteri
  - assenza di opzionalità
  - semplicità (meglio interni che esterni, meglio un attributo)
  - utilizzo nelle operazioni più frequenti o importanti
  - occupazione di memoria

Se nessuno degli identificatori soddisfa i requisiti visti?

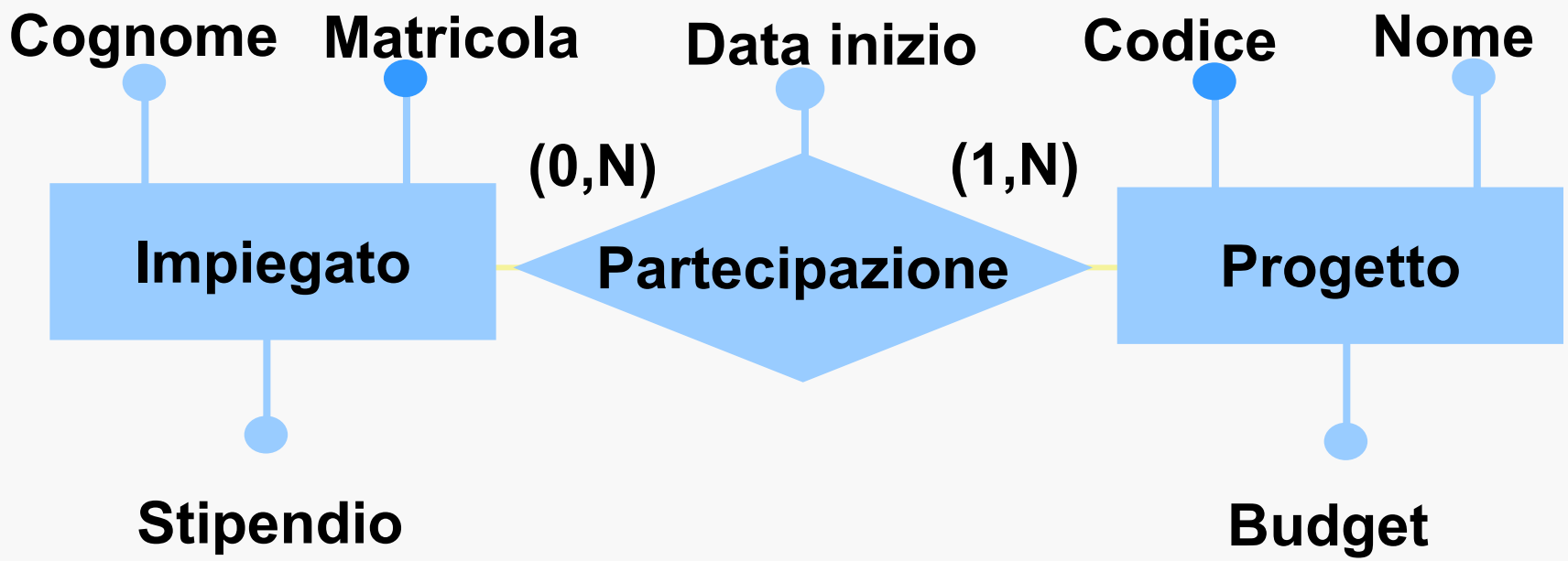
Si introducono nuovi attributi (codici) contenenti valori speciali generati appositamente per questo scopo

Traduzione verso il modello relazionale

- idea di base:
  - le entità con determinati attributi diventano tabelle sugli stessi attributi, con identificatore primario dell'entità che diventa chiave primaria
  - per quanto riguarda le relationships, dipende dalla molteplicità (N-N, 1-N, 1-1)

Ricordate che i diagrammi ER non rappresentano le occorrenze, quindi tradurremo in uno schema di relazioni.

Entità e relationship molti a molti



Impiegato(<u>Matricola</u>, Cognome, Stipendio)

Progetto(<u>Codice</u>, Nome, Budget)

Partecipazione(<u>Matricola</u>, <u>Codice</u>, DataInizio)

Entità e relationship molti a molti

Impiegato(Matricola, Cognome, Stipendio)
Progetto(Codice, Nome, Budget)
Partecipazione(Matricola, Codice, DataInizio)

- con vincoli di integrità referenziale fra
  - Matricola in Partecipazione e (la chiave di) Impiegato
  - Codice in Partecipazione e (la chiave di) Progetto

Nomi più espressivi per gli attributi della chiave della relazione che rappresenta la relationship

Impiegato(Matricola, Cognome, Stipendio)

Progetto(Codice, Nome, Budget)

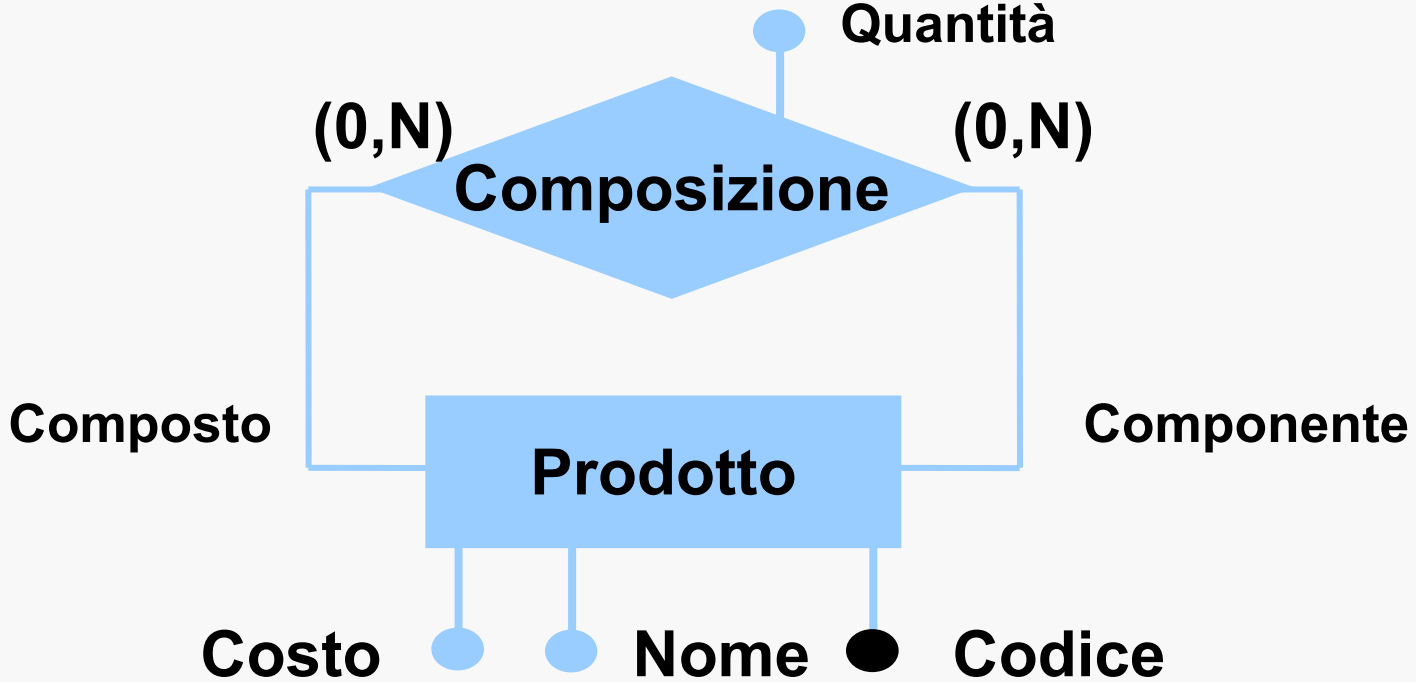
Partecipazione(Impiegato, Progetto, DataInizio)

Nota

- La traduzione non tiene conto delle cardinalità minime delle relationship molti a molti, ovvero, tutte le relationship molti a molti si traducono così.

- Ovvero, per le occorrenze che non partecipano alla relationship nello schema di partenza, non ci saranno righe corrispondenti nella tabella che rappresenta la relationship.

Relationship ricorsive (con ruoli)

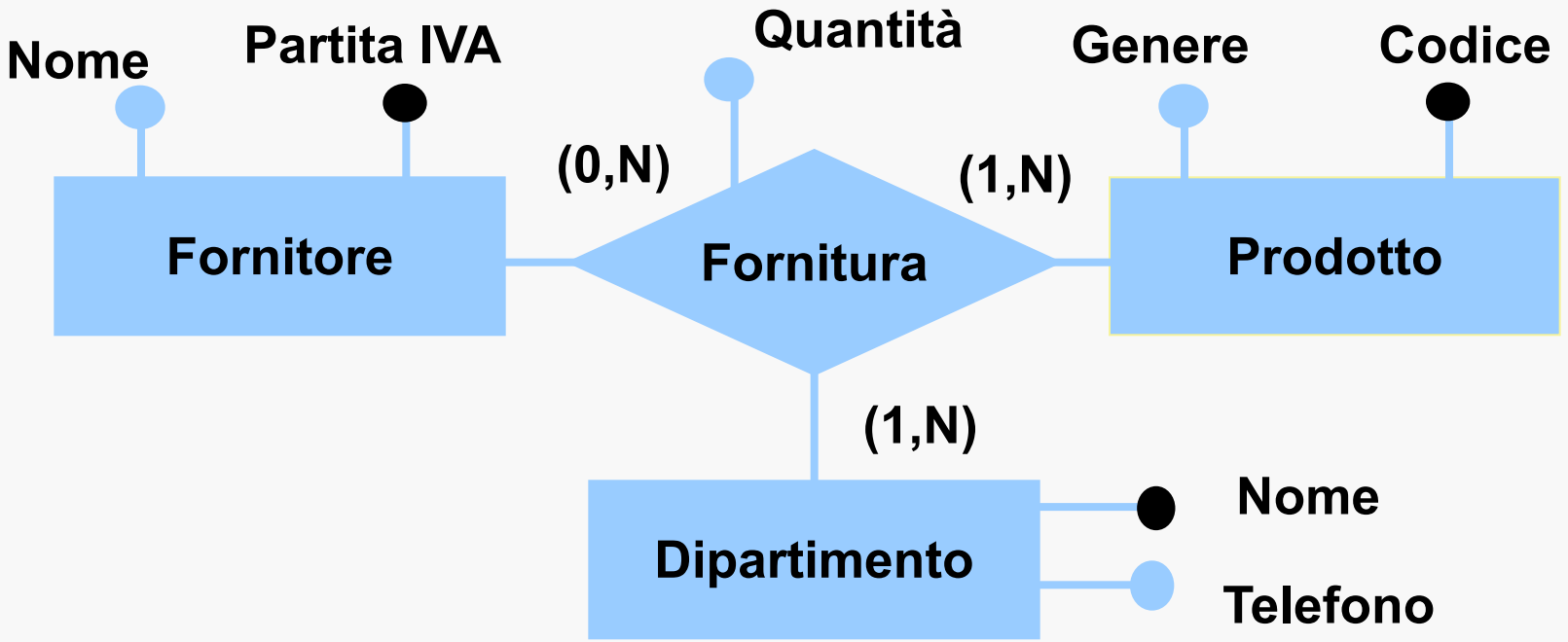


Prodotto(<u>Codice</u>, Nome, Costo)

Composizione(<u>Composto</u>, <u>Componente</u>, Quantità)

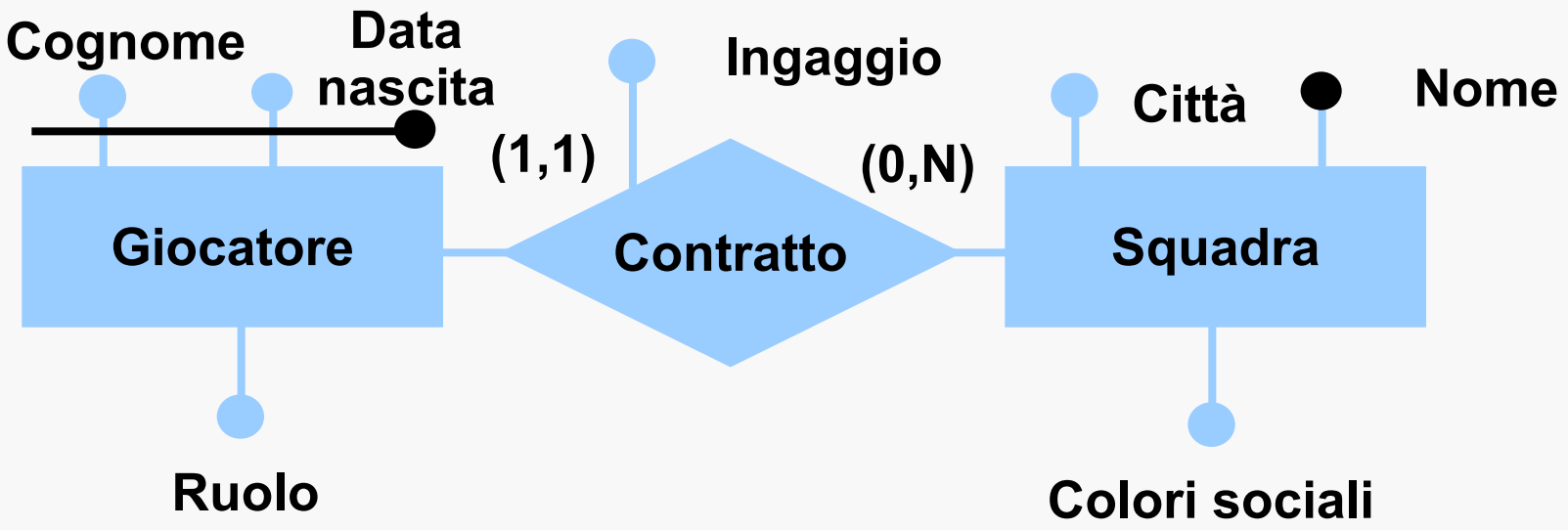
+ vincoli (Composto -> Codice; Componente -> Codice)

Relationship n-arie



Fornitore(<u>PartitaIVA</u>, Nome), Prodotto(<u>Codice</u>, Genere)
Dipartimento(<u>Nome</u>, Telefono)
Fornitura(<u>Fornitore</u>, <u>Prodotto</u>, <u>Dipartimento</u>, Quantità)
+ 3 vincoli di integrità referenziale

Relationship uno a molti



Giocatore(<u>Cognome, DataNascita</u>, Ruolo)

Contratto(<u>CognGiocatore, DataNascG, Squadra</u>, Ingaggio)

Squadra(<u>Nome</u>, Città, ColoriSociali)

- corretto?

Soluzione più compatta

Giocatore(Cognome, DataNascita, Ruolo)
Contratto(CognGiocatore, DataNascG, Squadra, Ingaggio)
Squadra(Nome, Città, ColoriSociali)

Squadra è dipendente da Giocatore, quindi la chiave di Contratto è solo CognGiocatore, DataNascG, ma a questo punto Giocatore e Contratto sono in relazione biunivoca:

Giocatore(Cognome, DataNasc, Ruolo, Squadra, Ingaggio)
Squadra(Nome, Città, ColoriSociali)

- **con vincolo di integrità referenziale fra Squadra in Giocatore e la chiave di Squadra**

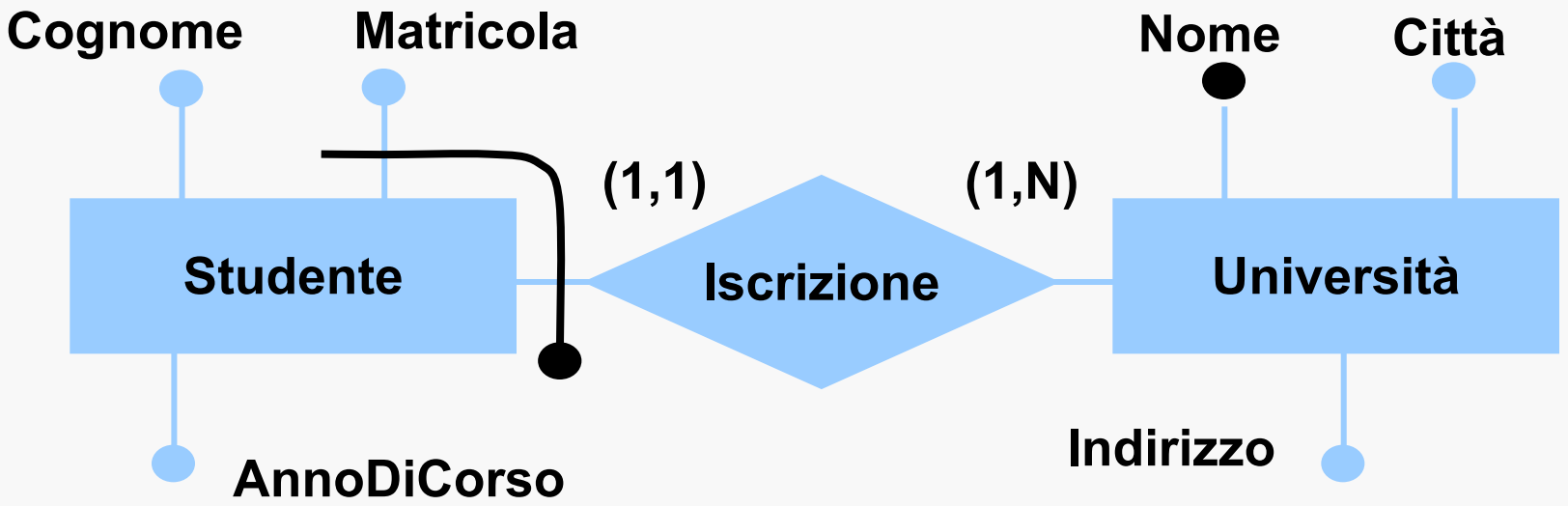
Nota

- La traduzione questa volta riesce a rappresentare efficacemente la cardinalità minima della partecipazione che ha 1 come cardinalità massima:
  - 0 : valore nullo ammesso
  - 1 : valore nullo non ammesso

  Quindi, se Giocatore partecipa alla relationship con cardinalità (0,1), la sua rappresentazione tabellare è

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

Entità con identificazione esterna

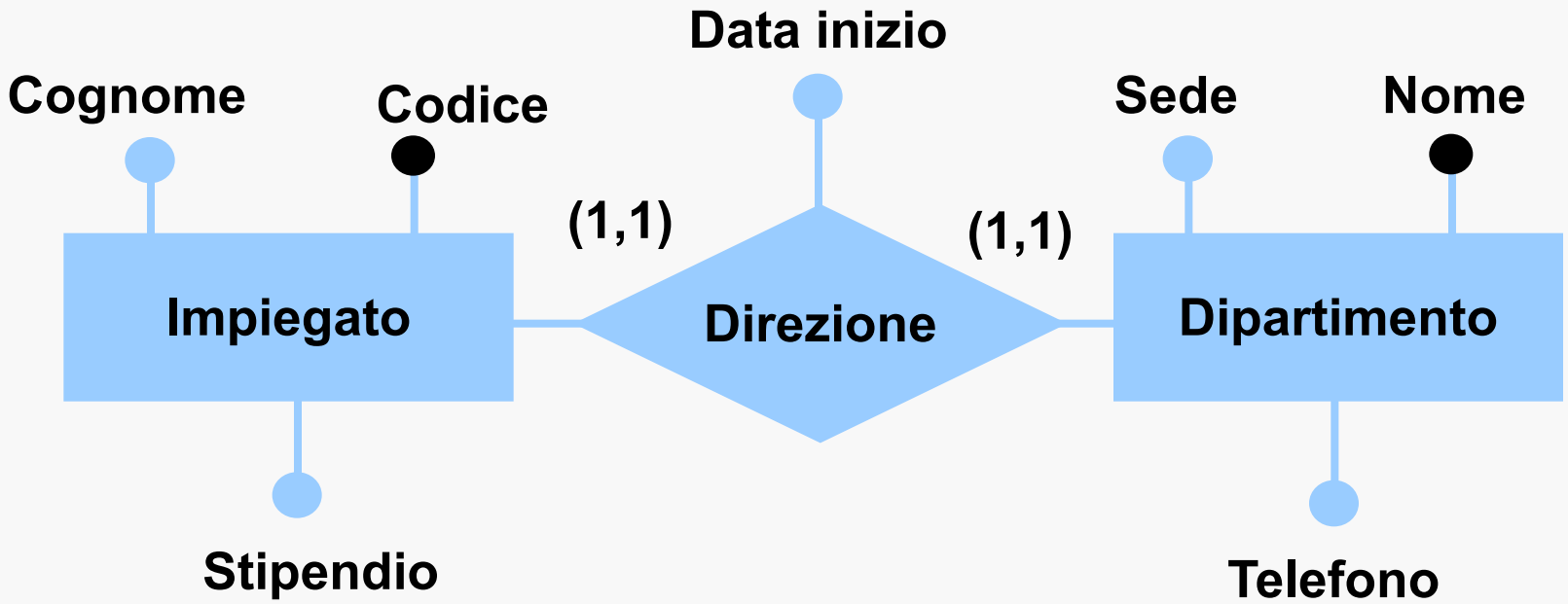


Studente(Matricola, Università, Cognome, AnnoDiCorso)

Università(Nome, Città, Indirizzo)

- con vincolo tra Università in Studente ed Università (la chiave esterna funge anche da vincolo)

Relationship uno a uno



- varie possibilità:
  - fondere da una parte o dall'altra
  - fondere tutto?
  - dipendenti da cardinalità minime

Relationship uno a uno con entrambe le partecipazioni obbligatorie – due opzioni

Impiegato (Codice, Cognome, Stipendio)

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

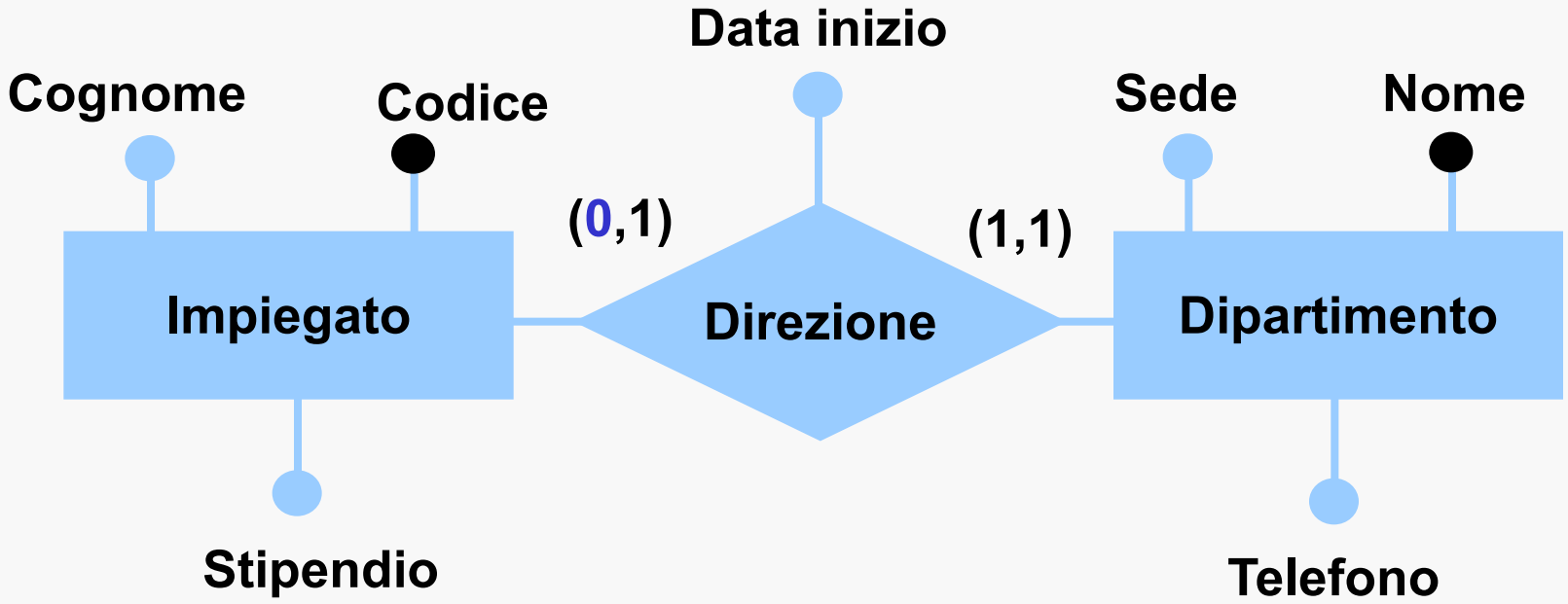
+ vincolo di integrità referenziale tra Direttore in Dipartimento ed Impiegato

Impiegato (Codice, Cognome, Stipendio, Dipartimento, DataInizio)

Dipartimento (Nome, Sede, Telefono)

+ vincolo di integrità referenziale tra Dipartimento in Impiegato e Dipartimento

Una possibilità privilegiata se cardinalità minima 0

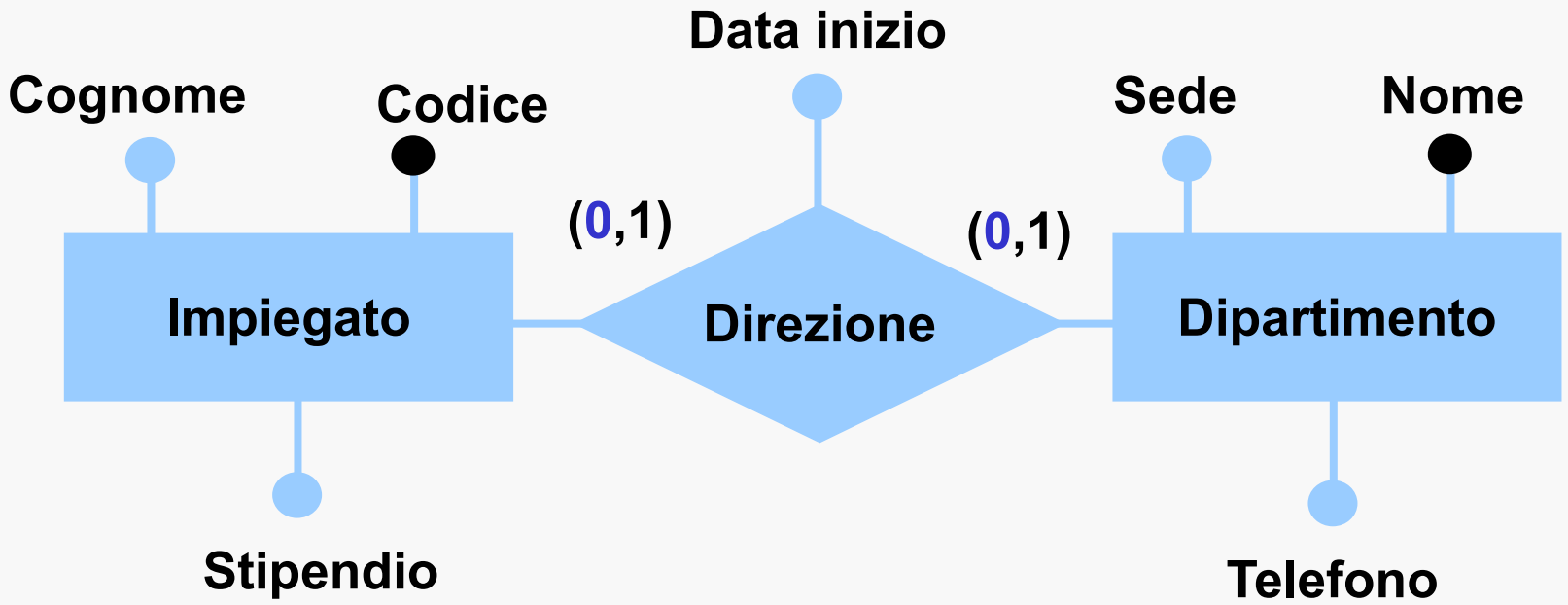


Impiegato (Codice, Cognome, Stipendio)

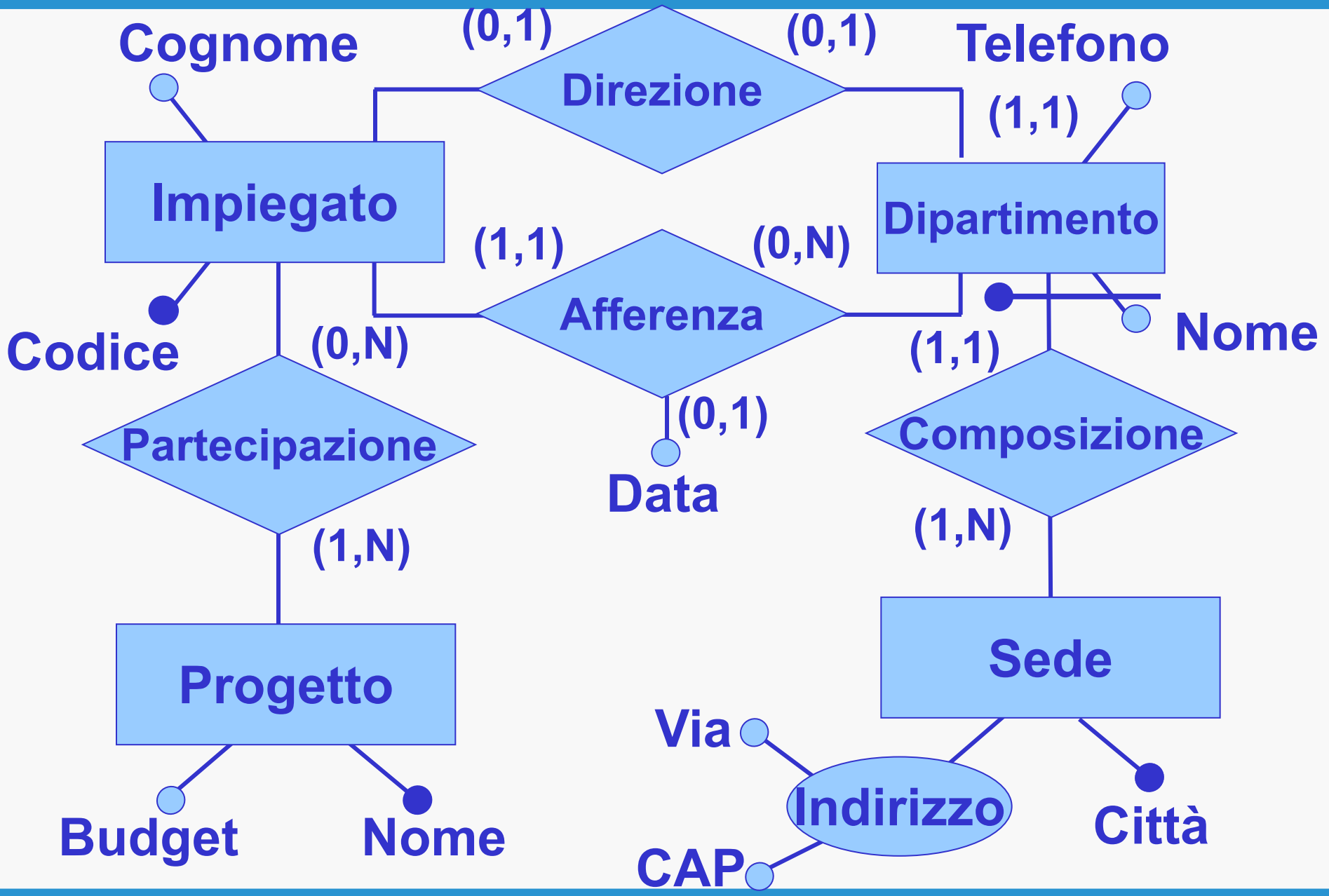
Dipartimento (Nome, Sede, Telefono, Direttore, InizioD)

- **con vincolo di integrità referenziale, senza valori nulli**

Un altro caso



Per questo caso esiste una terza opzione: rappresentarlo come tre tabelle come nel caso molti a molti, evitando valori nulli.



Schema finale tradotto: Ma come procedo?

Impiegato(<u>Codice</u>, Cognome, Dipartimento,Sede, Data*)

Dipartimento(<u>Nome</u>, <u>Città</u>, Telefono, Direttore*)

Sede(<u>Città</u>, Via, CAP)

Progetto(<u>Nome</u>, Budget)

Partecipazione(<u>Impiegato</u>, <u>Progetto</u>)
+ vincoli

Idea: prima entità, poi relantionship N-N, poi le altre relationship.