Normalizzazione

Eliminare le anomalie

- Abbiamo sviluppato la teoria delle dipendenze funzionali per **identificare le anomalie** in uno schema mal definito
- Adesso siamo in grado di affrontare il passaggio da **schemi "con anomalie"** a **schemi "ben fatti"**
- Per fare ciò definiremo un nuovo concetto, le **forme normali**, intese come proprietà che devono essere soddisfatte dalle dipendenze fra attributi di schemi "ben fatti"
- Vedremo solo la **forma normale di Boyce-Codd** (BCNF) e la **terza forma normale** (3NF)

Forma Normale di Boyce-Codd

- Uno schema $R(T, F)$ è in **forma normale di Boyce-Codd** (**BCNF**) se e solo se per ogni dipendenza funzionale non banale $X \to Y \in F^+$, X è una **superchiave** di R

- L'idea su cui si basa la BCNF è che una dipendenza funzionale $X \to A$, in cui X non contiene attributi estranei, indica che, nella realtà che si modella, esiste una collezione di entità omogenee che sono univocamente identificate da X

Forma Normale di Boyce-Codd

- Dalla definizione, il fatto che uno schema sia in **BCNF dipende dalla chiusura** F^+, non dalla specifica copertura F
- Purtroppo per **calcolare** F^+ abbiamo solo algoritmi di **complessità esponenziale**, che costano troppo
- Tuttavia possiamo facilmente **stabilire** se uno schema è in BCNF con un algoritmo di **complessità polinomiale**

Forma Normale di Boyce-Codd

- **Teorema**:
  - Uno schema $R(T, F)$ è in BCNF se e solo se per **ogni dipendenza funzionale non banale** $X \to Y \in F$, X è una **superchiave**

- **Corollario**:
  - Uno schema $R(T, F)$ con F copertura minimale è in BCNF se e solo se per **ogni dipendenza funzionale elementare** $X \to A \in F$, X è una **superchiave**.

Verifica BCNF

Input: schema $R(T, F)$

Output: **true** se R è in BCNF, **false** altrimenti

for each $X \to Y \in F$ **do**

  if $Y \not\subseteq X$ **and** $T \not\subseteq X^+$ **then**

    return false

return true

Verifica BCNF

Input: schema $R(T, F)$

Output: **true** se R è in BCNF, **false** altrimenti

for each $X \to Y \in F$ **do**

Controlliamo ogni dipendenza funzionale della relazione

  if $Y \not\subseteq X$ **and** $T \not\subseteq X^+$ **then**

    return false

return true

Verifica BCNF

Input: schema $R(T, F)$

Output: **true** se R è in BCNF, **false** altrimenti

for each $X \to Y \in F$ **do**

Controlliamo ogni dipendenza funzionale della relazione

  if $Y \not\subseteq X$ **and** $T \not\subseteq X^+$ **then**

Se X non è superchiave

    return false

return true

Verifica BCNF

Input: schema $R(T, F)$

Output: **true** se R è in BCNF, **false** altrimenti

for each $X \to Y \in F$ **do**

Controlliamo ogni dipendenza funzionale della relazione

if $Y \not\subseteq X$ **and** $T \not\subseteq X^+$ **then**

Se X non è superchiave

return false

Se la dipendenza funzionale è non banale

return true

Esempio 7

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

- Impiegato $\rightarrow$ Stipendio
- Progetto $\rightarrow$ Bilancio
- Impiegato, Progetto $\rightarrow$ Funzione

Esempio 7

- Proviamo a normalizzare il precedente schema in BNCF con una "procedura intuitiva"
- Questa procedura non è valida in generale, ma solo in alcuno "casi semplici"
- Per ogni dipendenza $X \to Y$ che viola la BCNF, definiamo una nuova relazione su XY ed eliminiamo Y dalla relazione originaria

Esempio 7

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20	Marte	2	tecnico
Verdi	35	Giove	15	progettista
Verdi	35	Venere	15	progettista
Neri	55	Venere	15	direttore
Neri	55	Giove	15	consulente
Neri	55	Marte	2	consulente
Mori	48	Marte	2	direttore
Mori	48	Venere	15	progettista
Bianchi	48	Venere	15	progettista
Bianchi	48	Giove	15	direttore

- Impiegato → Stipendio
- Progetto → Bilancio
- Impiegato, Progetto → Funzione

Esempio 7

Impiegato	Stipendio
Rossi	20
Verdi	35
Neri	55
Mori	48
Bianchi	48

Progetto	Bilancio
Marte	2
Giove	15
Venere	15

Impiegato	Progetto	Funzione
Rossi	Marte	tecnico
Verdi	Giove	progettista
Verdi	Venere	progettista
Neri	Venere	direttore
Neri	Giove	consulente
Neri	Marte	consulente
Mori	Marte	direttore
Mori	Venere	progettista
Bianchi	Venere	progettista
Bianchi	Giove	direttore

- Impiegato → Stipendio
- Progetto → Bilancio
- Impiegato, Progetto → Funzione

Esempio 8

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

- Impiegato → Sede
- Progetto → Sede

Esempio 8

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

- Impiegato $\rightarrow$ Sede
- Progetto $\rightarrow$ Sede

Esempio 8

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

- Impiegato $\rightarrow$ Sede
- Progetto $\rightarrow$ Sede

Ricostruiamo la relazione di partenza

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Verdi	Saturno	Milano
Neri	Giove	Milano

Ricostruiamo la relazione di partenza

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

Progetto	Sede
Marte	Roma
Giove	Milano
Saturno	Milano
Venere	Milano

Impiegato	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Venere	Milano
Neri	Saturno	Milano
Neri	Venere	Milano
Verdi	Saturno	Milano
Neri	Giove	Milano

Decomposizione di schemi

- Dato uno schema $R(T)$, l'insieme di schemi $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$ è una **decomposizione** di R se e solo se $\cup_i T_i = T$
- Si noti che la precedente definizione non richiede che gli schemi R_i siano disgiunti
- Come caratterizzare **l'equivalenza tra schema originario e sua decomposizione**? In generale la decomposizione deve:
  - **preservare i dati**
  - **preservare le dipendenza**

Esempio di perdita di dati

R

P	T	C
p1	t1	c1
p1	t2	c2
p1	t3	c2

Esempio di perdita di dati

R

P	T	C
p1	t1	c1
p1	t2	c2
p1	t3	c2

$R_1 = \pi_{PT}(R)$

P	T
p1	t1
p1	t2
p1	t3

$R_2 = \pi_{PC}(R)$

P	C
p1	c1
p1	c2

Esempio di perdita di dati

R

P	T	C
p1	t1	c1
p1	t2	c2
p1	t3	c2

$R_1 = \pi_{PT}(R)$

P	T
p1	t1
p1	t2
p1	t3

$R_2 = \pi_{PC}(R)$

P	C
p1	c1
p1	c2

$R_1 \bowtie R_2$

P	T	C
p1	t1	c1
p1	t1	c2
p1	t2	c1
p1	t2	c2
p1	t3	c1
p1	t3	c2

Esempio di perdita di dipendenze

R

P	T	C
p1	t1	c1
p1	t2	c2
p1	t3	c2

$$T \to C$$

$$C \to P$$

questa decomposizione
preserva i dati

Esempio di perdita di dipendenze

R

P	T	C
p1	t1	c1
p1	t2	c2
p1	t3	c2

$R_1 = \pi_{PT}(R)$

P	T
p1	t1
p1	t2
p1	t3

$R_2 = \pi_{TC}(R)$

T	C
t1	c1
t2	c2
t3	c2

$$T \to C$$

$$C \to P$$

questa decomposizione preserva i dati

questa decomposizione non preserva la dipendenza

$$C \to P$$

perché gli attributi sono in relazioni diverse

Teorema della perdita di dati

- **Teorema**:
  - Se $\rho = \left\{R_1(T_1), \ldots, R_k(T_k)\right\}$ è una decomposizione di $R(T, F)$, allora per ogni istanza r di $R(T)$ si ha

$$r \subseteq \pi_{T_1}(r) \bowtie \cdots \bowtie \pi_{T_k}(r)$$

- *Dimostrazione*:
  - Per esercizio 😅

- Questo teorema ci dice che perdiamo informazione quando, **ricostruendo una relazione**, **otteniamo più n-uple che nella relazione originaria**

Decomposizione che preserva i dati

- Dato uno schema $R(T, F)$ e una decomposizione $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$, ρ è una **decomposizione** di $R(T, F)$ **che preserva i dati** se e solo se, per ogni relazione r che soddisfa $R(T, F)$, si ha:

$$r = \pi_{T_1}(r) \bowtie \cdots \bowtie \pi_{T_k}(r)$$

- Questa definizione ci dice che, per una decomposizione che preserva i dati, ogni istanza valida r della relazione di partenza **deve essere uguale al join naturale delle sue proiezioni** sui vari T_i

Teorema di preservazione dei dati

- Sia $\rho = \left\{R_1(T_1), R_2(T_2)\right\}$ una decomposizione di $R(T, F)$; essa preserva i dati se e solo se $T_1 \cap T_2 \rightarrow T_1 \in F^+$ oppure $T_1 \cap T_2 \rightarrow T_2 \in F^+$.

- In altre parole, gli **attributi comuni** alle due relazioni **devono essere chiave in una delle due** tabelle

- Nel nostro esempio, Sede è l'attributo a comune tra le due tabelle, ma non è chiave per nessuna delle due
  - Non c'è nessuna dipendenza con Sede come parte sinistra

Proiezioni di un insieme di dipendenze

- Dato $R(T, F)$ e $T_i \subseteq T$, **la proiezione dell'insieme di dipendenze F sull'insieme di attributi T_i** è

$$\pi_{T_i}(F) = \{X \to Y \in F^+ \,|\, X, Y \subseteq T_i\}$$

- Nota bene che la proiezione è costruita considerando le dipendenze in F^+, non quelle in F
- Esempio:
  - $R(ABC, \{A \to B, B \to C, C \to A\})$
  - $\pi_{AB}(F) = \{A \to B, B \to A\}$
  - $\pi_{AC}(F) = \{A \to C, C \to A\}$

Algoritmo per il calcolo di $\pi_{T_i}(F)$

Input: $R(T, F)$ e $T_i \subseteq T$

Output: $\pi_{T_i}(F)$

$Z \leftarrow \{\}$

for each $Y \subset T_i$ **do**

$W \leftarrow Y^+ - Y$

$Z \leftarrow Z \cup \left\{ Y \rightarrow (W \cap T_i) \right\}$

return Z

Calcolo di $\pi_{T_i}(F)$

- L'algoritmo precedente ha **complessità esponenziale** nel caso pessimo
- Consideriamo
  - $R(A_1, \ldots, A_n, B_1, \ldots, B_n, C_1, \ldots, C_n, D)$
  - $F = \left(\cup_i \{A_i \to C_i, B_i \to C_i\} \right) \cup \{C_1 \cdots C_n \to D\}$
- La proiezione di F su $A_1 \cdots A_n B_1 \cdots B_n D$ è pari a $\{X_1 \cdots X_n \to D$ dove $X_i = A_i$ oppure $X_i = B_i\}$
- La sua dimensione è esponenziale rispetto al numero di attributi e di dipendenze funzionali
- Si può dimostrare che nessun altro insieme "equivalente" ha cardinalità inferiore

Decomposizione che preserva le dipendenze

- Dato uno schema $R(T, F)$ e una decomposizione $\rho = \left\{R_1(T_1), \ldots, R_k(T_k)\right\}$, ρ è una **decomposizione** di $R(T, F)$ **che preserva le dipendenze** se e solo se:

$$\cup_i \pi_{T_i}(F) \equiv F$$

- Si noti il simbolo di equivalenza $\equiv$
- La decomposizione di $R(T, F)$ in due relazioni con attributi X e Y è una decomposizione che preserva le dipendenze se $\pi_X(F) \cup \pi_Y(F) \equiv F$, cioè se

$$\left(\pi_X(F) \cup \pi_Y(F)\right)^+ = F^+$$

Verificare una decomposizione

- Per **verificare** se una decomposizione di $R(T, F)$ in due relazioni con attributi X e Y preserva le dipendenze bisogna verificare che

$$\Big(\pi_X(F) \cup \pi_Y(F)\Big)^+ = F^+$$

- Per fare ciò:
  - è **necessario** saper **calcolare la proiezione** di un insieme di dipendenze funzionali su un insieme di attributi
  - è **necessario** saper **determinare l'equivalenza** di due insiemi di dipendenze funzionali

Verificare una decomposizione

- Per **calcolare la proiezione** di un insieme di dipendenze funzionali su un insieme di attributi abbiamo un algoritmo con **complessità esponenziale**
- Per **verificare l'equivalenza** di due insiemi di dipendenze funzionali F e G abbiamo un algoritmo con **complessità polinomiale**
  - Per ogni $X \to Y \in F$, calcoliamo X_G^+ e verifichiamo se $Y \in X_G^+$
  - Per ogni $X \to Y \in G$, calcoliamo X_F^+ e verifichiamo se $Y \in X_F^+$

Algoritmo per decomposizione in BCNF

Input: $R(T, F)$ (per semplicità gli elementi di F sono nella forma $X \to A$)

Output: ρ che preserva i dati

$\rho \leftarrow \{R(T, F)\}$

while esiste $R_i(T_i, F_i) \in \rho$ che non è in BCNF **do**

  for each $X \to A \in F_i$ **do**

    if $A \notin X$ **and** $T_i \not\subseteq X^+$ **then**

      $R_1 \leftarrow R_i\left(T_i - A, \pi_{T_i - A}(F_i)\right)$

      $R_2 \leftarrow R_i\left(X + A, \pi_{X+A}(F_i)\right)$

      $\rho \leftarrow \rho - \{R_i\} \cup \{R_1, R_2\}$

      break

return ρ

Algoritmo per decomposizione in BCNF

- **Teorema**:
  - Qualunque sia la relazione, l'esecuzione dell'algoritmo per decomposizione in BCNF su tale relazione termina e produce una decomposizione della relazione tale che:
    - la decomposizione prodotta è in BCNF
    - la decomposizione prodotta preserva i dati

- Non è garantito che la decomposizione generata preservi le dipendenze

Esempio 9

- Sia $R =$ Telefoni
- Sia $T =$ {Prefisso, Numero, Località}
- Sia $F =$ {Prefisso, Numero $\rightarrow$ Località,
  Località $\rightarrow$ Prefisso}

- Inizialmente $\rho =$ {Telefoni}
- La dipendenza Località $\rightarrow$ Prefisso viola la BCNF
- Rimpiazziamo $R =$ Telefoni in ρ con
  - $R_1\left(\{\text{Numero}, \text{Località}\}, \{\,\}\right)$
  - $R_2\left(\{\text{Località}, \text{Prefisso}\}, \{\text{Località} \rightarrow \text{Prefisso}\}\right)$

Esempio 9

- La decomposizione $\rho = \{R_1, R_2\}$ con
  - $R_1\left(\{\text{Numero, Località}\}, \{\,\}\right)$
  - $R_2\left(\{\text{Località, Prefisso}\}, \{\text{Località} \rightarrow \text{Prefisso}\}\right)$
- è in BCNF e quindi l'algoritmo termina.

- La decomposizione ρ preserva i dati, ma non preserva le dipendenze funzionali
  - Prefisso, Numero $\rightarrow$ Località è perduta

Qualità delle decomposizioni

- Una decomposizione dovrebbe sempre garantire
  - di **essere in BCNF**
  - **l'assenza di perdite sui dati**, in modo da poter ricostruire le informazioni originarie tramite join naturali
- la **conservazione delle dipendenze funzionali**, in modo da mantenere i vincoli di integrità originari

Esempio 10

- "*Ogni dirigente ha una sede, e un progetto può essere diretto da più persone, ma in sedi diverse*"

Dirigente	**Progetto**	**Sede**
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Dirigente → Sede

Progetto, Sede → Dirigente

- Questa relazione è in BNCF?

Esempio 10

- Applichiamo l'algoritmo di verifica!

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto, Sede $\rightarrow$ Dirigente

Dirigente $\rightarrow$ Sede

Esempio 10

- Applichiamo l'algoritmo di verifica!

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto, Sede $\rightarrow$ Dirigente ✓

Dirigente $\rightarrow$ Sede

Esempio 10

- Applichiamo l'algoritmo di verifica!

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto, Sede → Dirigente ✓

Dirigente → Sede ✗

Esempio 10

- Come decomponiamo la relazione?
  - La dipendenza Progetto, Sede → Dirigente coinvolge tutti gli attributi e quindi nessuna decomposizione potrà preservarla
  - Possiamo calcolare una decomposizione in BCNF, ma non potrà preservare questa dipendenza

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Dirigente → Sede

Progetto, Sede → Dirigente

- Quando non si può raggiungere una BCNF di buona qualità, spesso si tratta di una cattiva progettazione...

- Quando non si può raggiungere una BCNF di buona qualità, spesso si tratta di una cattiva progettazione...
- ...tuttavia possiamo “abbandonare” la BNCF...

- Quando non si può raggiungere una BCNF di buona qualità, spesso si tratta di una cattiva progettazione...
- ...tuttavia possiamo “abbandonare” la BNCF...
- ...e adottare una nuova forma normale “meno restrittiva” della BCNF

Terza Forma Normale

- Una relazione $R(T, F)$ è in **terza forma normale** (3NF) se e solo se, per ogni **dipendenza funzionale non banale** $X \to A \in F^+$, è verificata almeno una delle seguenti condizioni:
  - X è una **superchiave** di R
  - A è **contenuto in almeno una chiave** di R (in questo caso si dice che A è un attributo primo)

- Come si vede dalla definizione, se R è in BCNF allora R è in 3NF, i.e., BCNF $\Rightarrow$ 3NF

Esempio 11

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto, Sede → Dirigente

Dirigente → Sede

- L'attributo Sede è contenuto in una chiave, quindi la relazione è in 3NF

Esempio 11

Dirigente	Progetto	Sede
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Progetto, Sede → Dirigente

Dirigente → Sede

- Tuttavia c'è una ridondanza nella ripetizione della sede del dirigente per i vari progetti che dirige

Verifica di 3NF

- Il problema di **decidere** se uno schema di relazione è in 3NF è **NP-completo**
  - Il miglior algoritmo deterministico noto ha **complessità esponenziale** nel caso peggiore
    - Per stabilire se uno schema è in 3NF occorre conoscere gli attributi primi, cioè le chiavi
    - L'algoritmo per calcolare le chiavi ha complessità esponenziale
- Tuttavia **si può sempre ottenere una decomposizione in 3NF** che preserva dati e dipendenze funzionali

Algoritmo per decomposizione in 3NF

- **Intuizione**:
  - Dato un insieme di attributi T e una **copertura minimale** G, si divide G in gruppi G_i in modo che tutte le dipendenze funzionali di ogni gruppo G_i abbiano **la stessa "parte" sinistra**.
  - Da ogni gruppo G_i si definisce uno schema di relazione composto da tutti gli attributi che appaiono in G_i, la cui chiave, detta **chiave sintetizzata**, è la parte sinistra comune.

Algoritmo per decomposizione in 3NF

Input: $R(T, F)$

Output: ρ che preserva i dati e le dipendenze e con ogni elemento in 3NF

1. **Trovare una copertura minimale** G di F e porre $\rho \leftarrow \{\}$
2. **Sostituire** in G ogni insieme di dipendenze $\{X \rightarrow A_1, \ldots, X \rightarrow A_h\}$ con la dipendenza $X \rightarrow A_1 \cdots A_h$
3. **Per ogni dipendenza** $X \rightarrow Y \in G$ creare uno schema con attributi XY in ρ
4. **Eliminare** da ρ ogni schema che sia contenuto in un altro schema di ρ
5. Se ρ non contiene nessuno schema i cui attributi costituiscono una superchiave di R, aggiungere a ρ uno schema con attributi W, dove W è una **chiave** di R

Algoritmo per decomposizione in BCNF

- **Teorema**:
  - Qualunque sia la relazione, l'esecuzione dell'algoritmo per decomposizione in 3NF su tale relazione termina e produce una decomposizione della relazione tale che:
    - la decomposizione prodotta è in 3NF
    - la decomposizione prodotta preserva i dati e le dipendenze funzionali

- La **complessità** dell'algoritmo è **polinomiale**

Esempio 12

- Dato $R(ABCD, F)$ con
  $F = \{AB \to C, C \to D, D \to B\}$
- F è una copertura minimale
- $AB \to C$: $R_1(ABC)$ con chiave sintetizzata AB
- $C \to D$: $R_2(CD)$ con chiave sintetizzata C
- $D \to B$: $R_3(BD)$ con chiave sintetizzata D
- $\pi_{R_2}(F) = \{C \to D\}$
- $\pi_{R_3}(F) = \{D \to B\}$
- $\pi_{R_1}(F) = \{AB \to C, C \to B\}$

Esempio 13

- Dato $R(ABCDEGH, F)$ con
  $F = \{ABC \to DEG, BD \to ACE, C \to BH, H \to BDE\}$
- Per prima cosa, calcoliamo la copertura minimale
  $$F \equiv F_1 = \{ABC \to D, ABC \to E, ABC \to G, BD \to A,$$
  $$BD \to C, BD \to E, C \to B, C \to H,$$
  $$H \to B, H \to D, H \to E\}$$
- ABC contiene attributi estranei?
  - $C^+ = CBHDEAG$, quindi A, B sono estranei in ABC
- BD contiene attributi estranei?
  - $B^+ = B$, $D^+ = D$ quindi non ci sono attributi estranei in BD
- $F_2 \equiv F_1 = \{C \to D, C \to E, C \to G, BD \to A,$
  $$BD \to C, BD \to E, C \to B, C \to H,$$
  $$H \to B, H \to D, H \to E\}$$

Esempio 13

- $F_2 \equiv F_1 = \{C \to D, C \to E, C \to G, BD \to A,$
  $BD \to C, BD \to E, C \to B, C \to H,$
  $H \to B, H \to D, H \to E\}$
- F_2 contiene dipendenze ridondanti?
  - $C \to D$ perché $C \to H \to D$
  - $C \to E$ perché $C \to H \to E$
  - $BD \to E$ perché $BD \to C \to H \to E$
  - $C \to B$ perché $C \to H \to B$
- $G \equiv F_2 = \{C \to G, BD \to A, BD \to C,$
  $C \to H, H \to B, H \to D, H \to E\}$

Esempio 13

- $G \equiv F_2 = \{C \to G, BD \to A, BD \to C, C \to H, H \to B, H \to D, H \to E\}$
- Prima di eseguire le sostituzioni previste, controlliamo se le parti sinistre delle dipendenze in G sono superchiavi
  - $C^+ = CBHDEAG$, quindi C è chiave
  - $BD^+ = BDACGHE$, quindi BD è superchiave
  - $H^+ = HBDEACG$, quindi H è chiave
- Possiamo concludere che il nostro schema è in BNCF, e quindi in 3NF, e non va decomposto

Esempio 14

- Dato $R(ABCDEGH, F)$ con
  $F = \{AB \to CDE, CE \to AB, A \to G, G \to BD\}$
- Per prima cosa, calcoliamo la copertura minimale
  $F \equiv F_1 = \{AB \to C, AB \to D, AB \to E, CE \to A,$
  $CE \to B, A \to G, G \to B, G \to D\}$
- AB contiene attributi estranei?
  - $A^+ = AGBDCE$, quindi B è estraneo in AB
- CE contiene attributi estranei?
  - $C^+ = C$, $E^+ = E$ quindi non ci sono attributi estranei in CE
  $F_2 \equiv F_1 = \{A \to C, A \to D, A \to E, CE \to A,$
  $CE \to B, A \to G, G \to B, G \to D\}$

Esempio 14

- $F_2 \equiv F_1 = \{A \to C, A \to D, A \to E, CE \to A, CE \to B, A \to G, G \to B, G \to D\}$
- F_2 contiene dipendenze ridondanti?
  - $A \to D$ perché $A \to G \to D$
  - $CE \to B$ perché $CE \to A \to G \to B$
- $G \equiv F_2 = \{A \to C, A \to E, CE \to A, A \to G, G \to B, G \to D\}$
- Controllo superchiavi
  - In G nessuna dipendenza funzionale include H, se le quindi nessuna delle parti sinistre delle dipendenze in G sono superchiavi

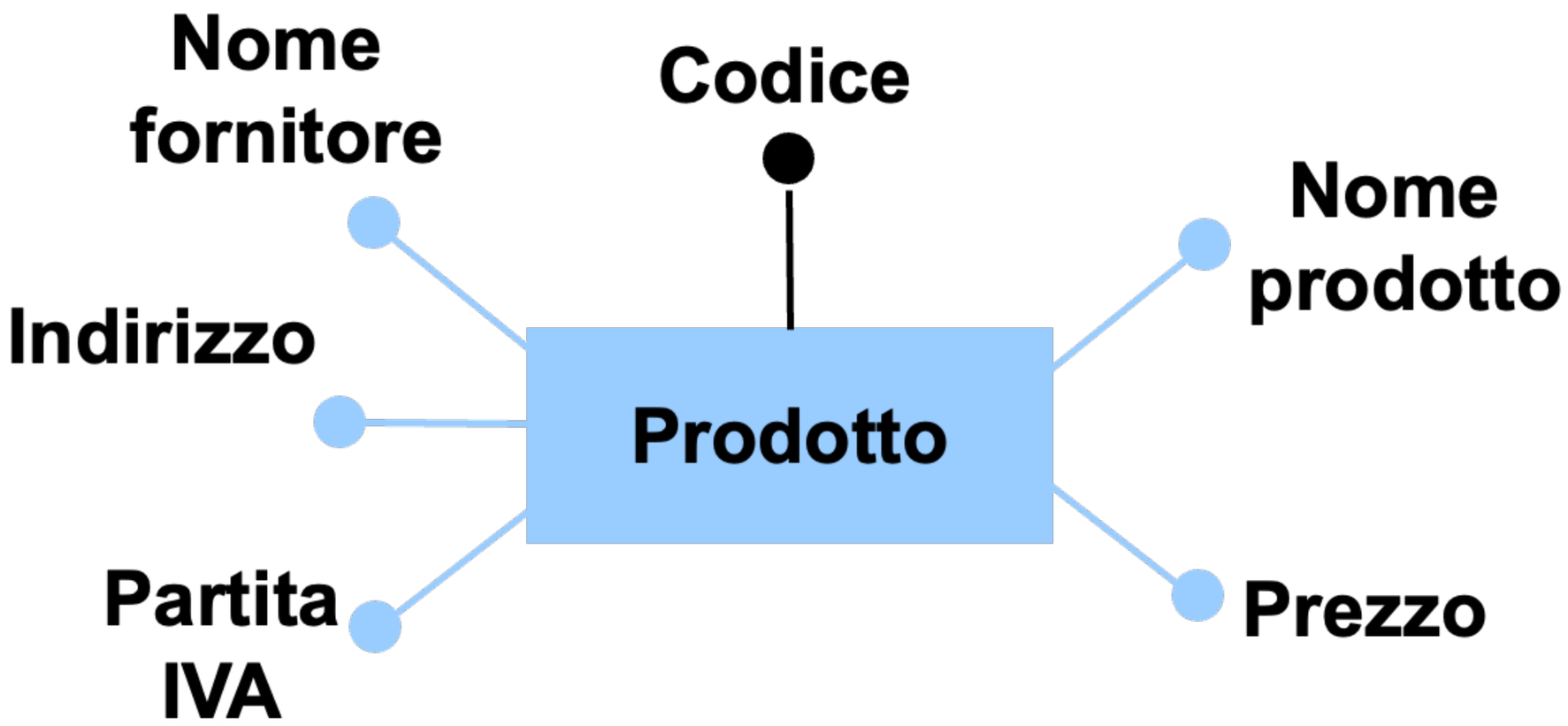
Esempio 14

- $G \equiv F_2 = \{A \to C, A \to E, CE \to A, A \to G, G \to B, G \to D\}$
- Decomponiamo!
  - $A \to C, A \to E, A \to G$, quindi creiamo $R_1(ACEG)$
  - $CE \to A$, quindi creiamo $R_2(CEA)$
  - $G \to B, G \to D$, quindi creiamo $R_3(GBD)$
- Eliminiamo!
  - $R_2(CEA)$ è contenuta in $R_1(ACEG)$, quindi la eliminiamo
- Controllo superchiave!
  - Nè $R_1(ACEG)$ nè $R_3(GBD)$ contengono H
  - Siccome AH è chiave, aggiungiamo $R_0(AH)$ alla decomposizione
- $\rho = \{R_1(ACEG), R_3(GBD), R_0(AH)\}$ è in 3NF

Progettazione e Normalizzazione

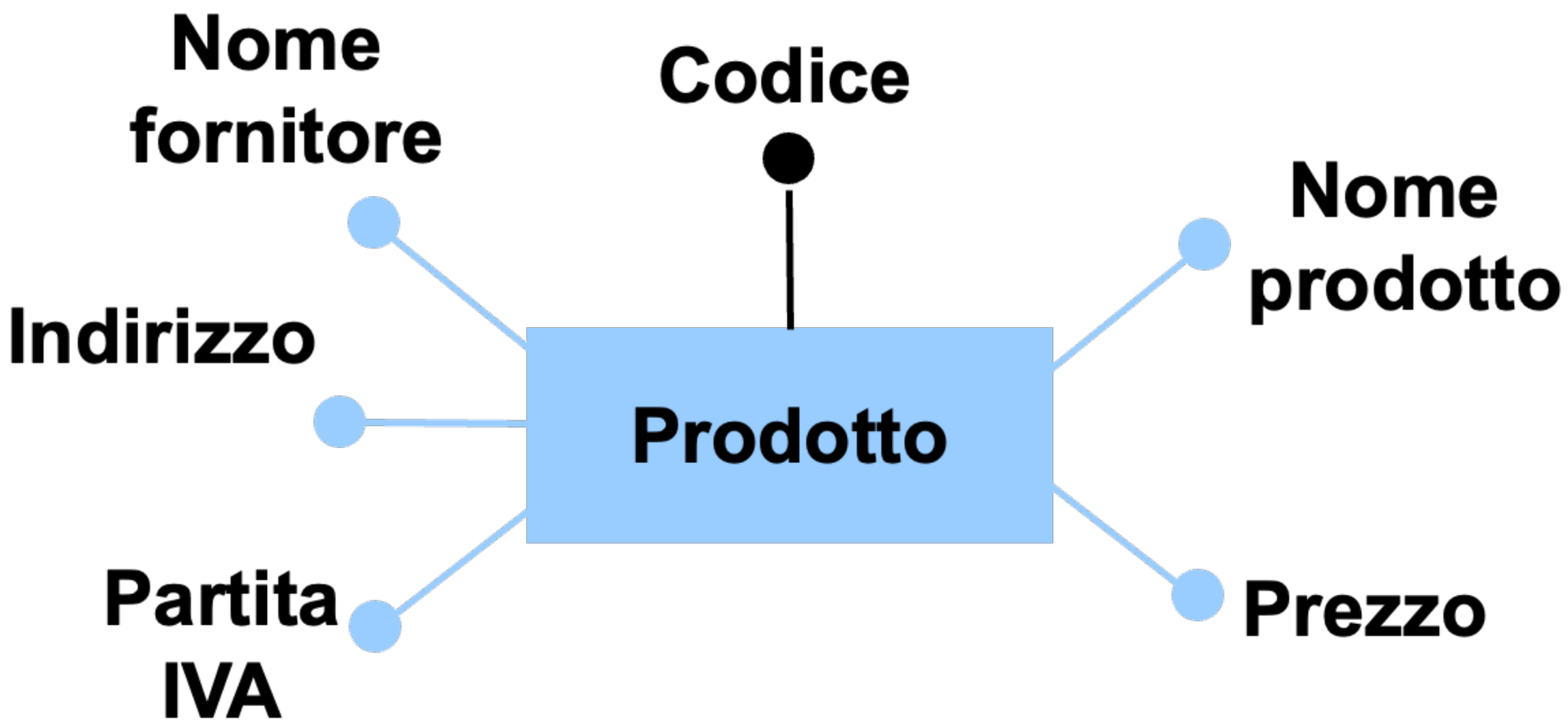
- La teoria della normalizzazione serve per verificare la qualità dello schema logico
- Ma si può usare anche durante la progettazione concettuale per ottenere uno schema di buona qualità (verifica ridondanze, partizionamento di entità/ relazioni)

Verifica di normalizzazione su entità



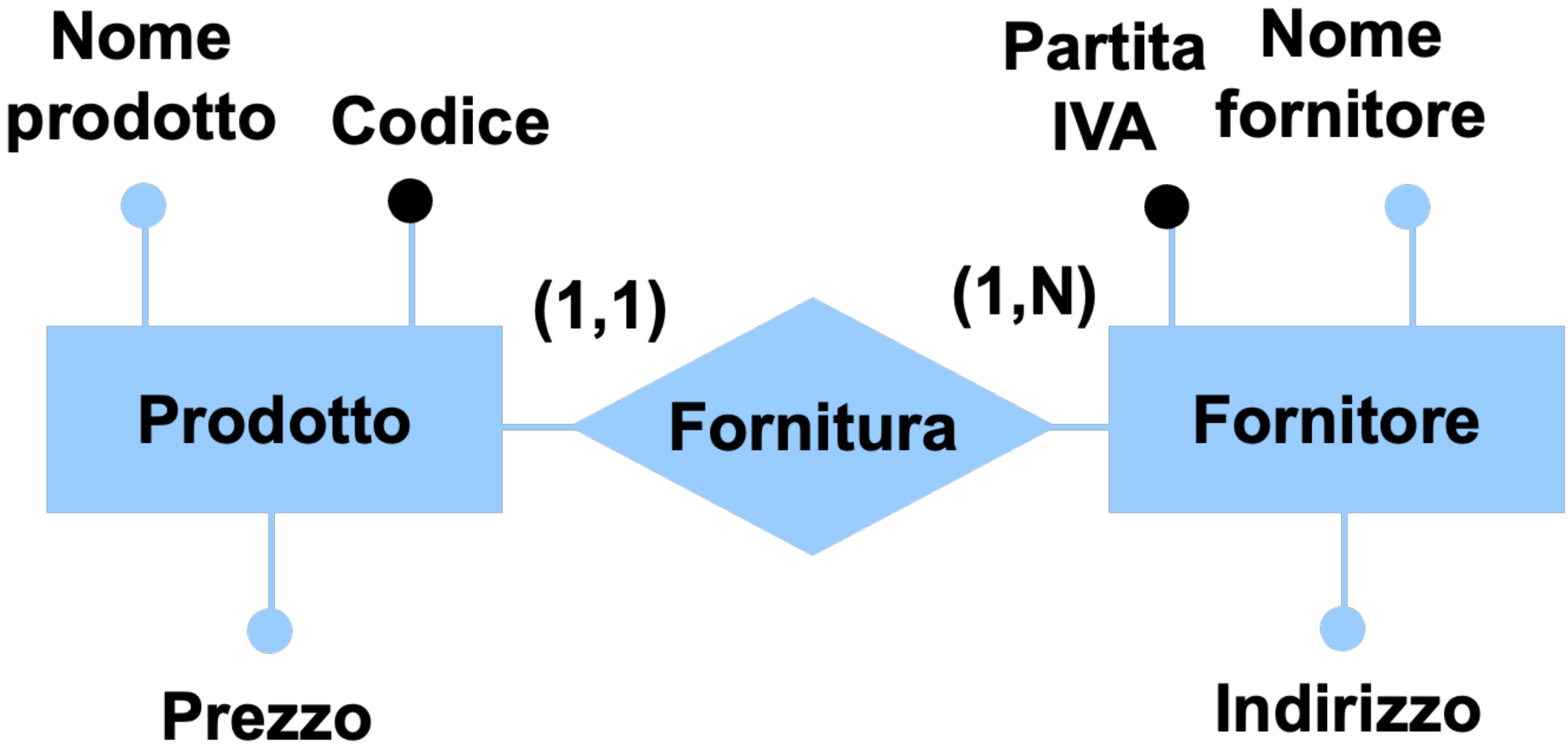
- Abbiamo la dipendenza funzionale
  - Partita IVA → Nome fornitore, Indirizzo
- Codice è chiave

Verifica di normalizzazione su entità

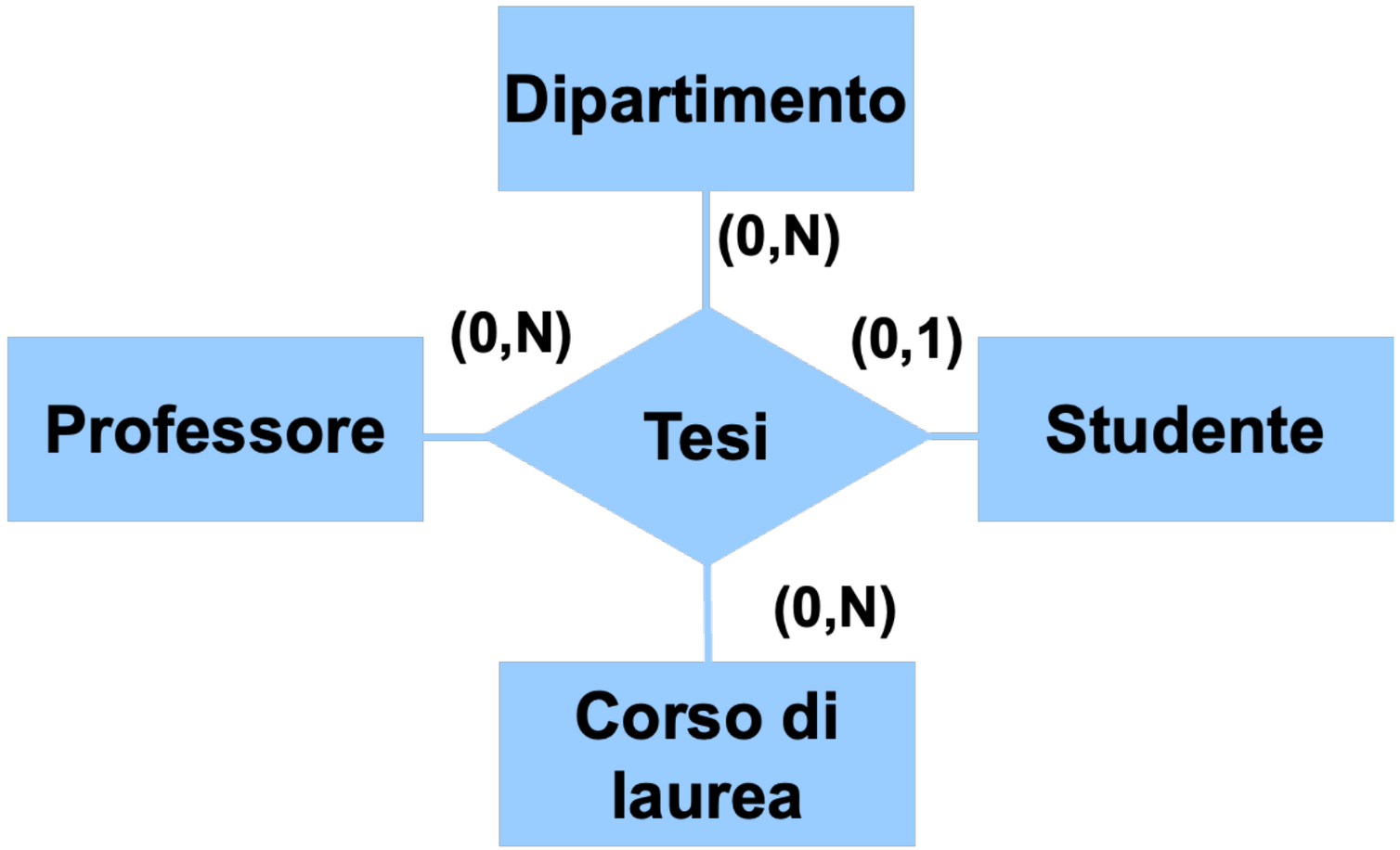


- Partita IVA → Nome fornitore, Indirizzo
  - Partita IVA non è superchiave
  - Nome fornitore e Indirizzo non fanno parte di una chiave
- L'entità viola la terza forma normale

Verifica di normalizzazione su entità

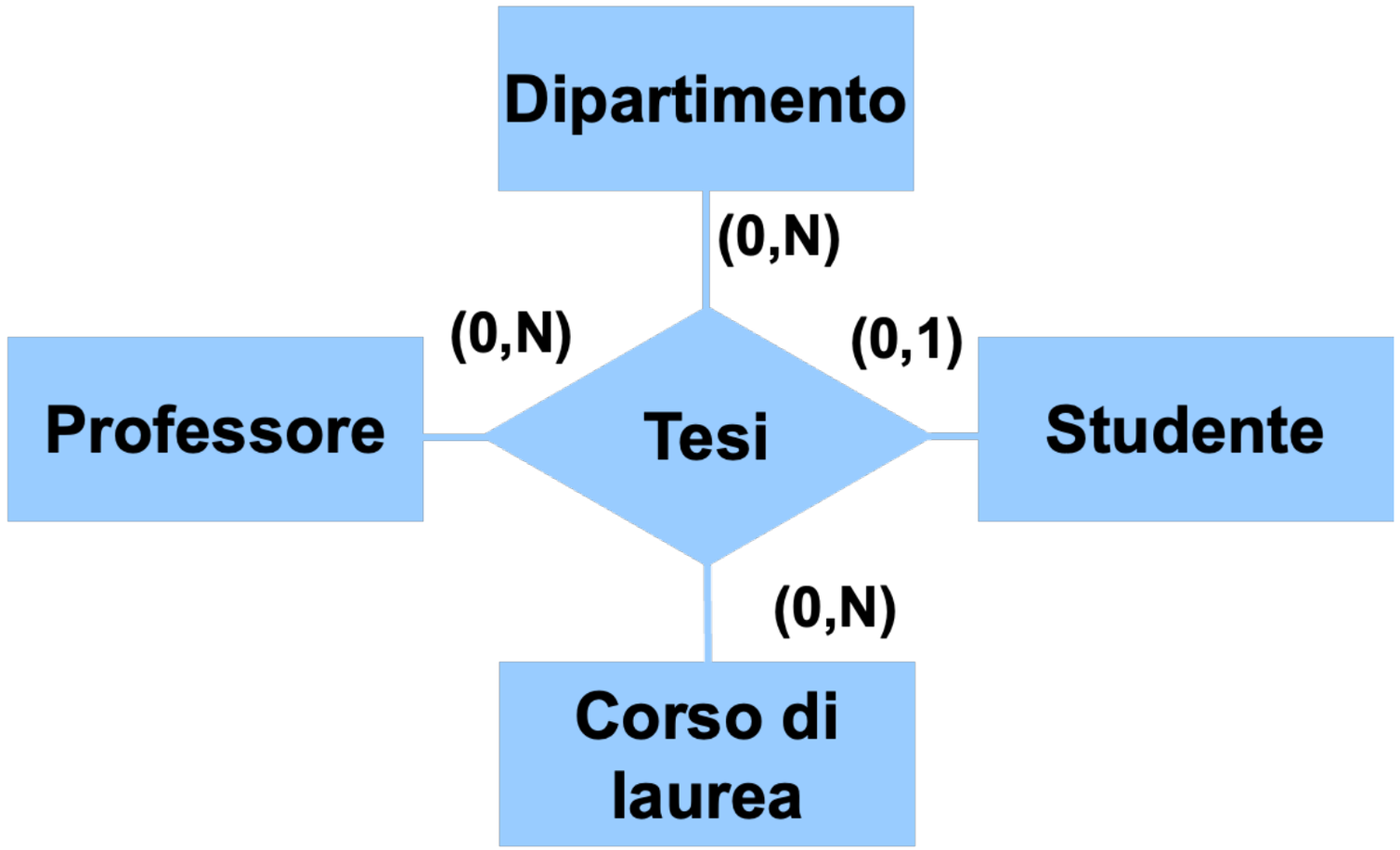


Verifica di normalizzazione su *relationship*



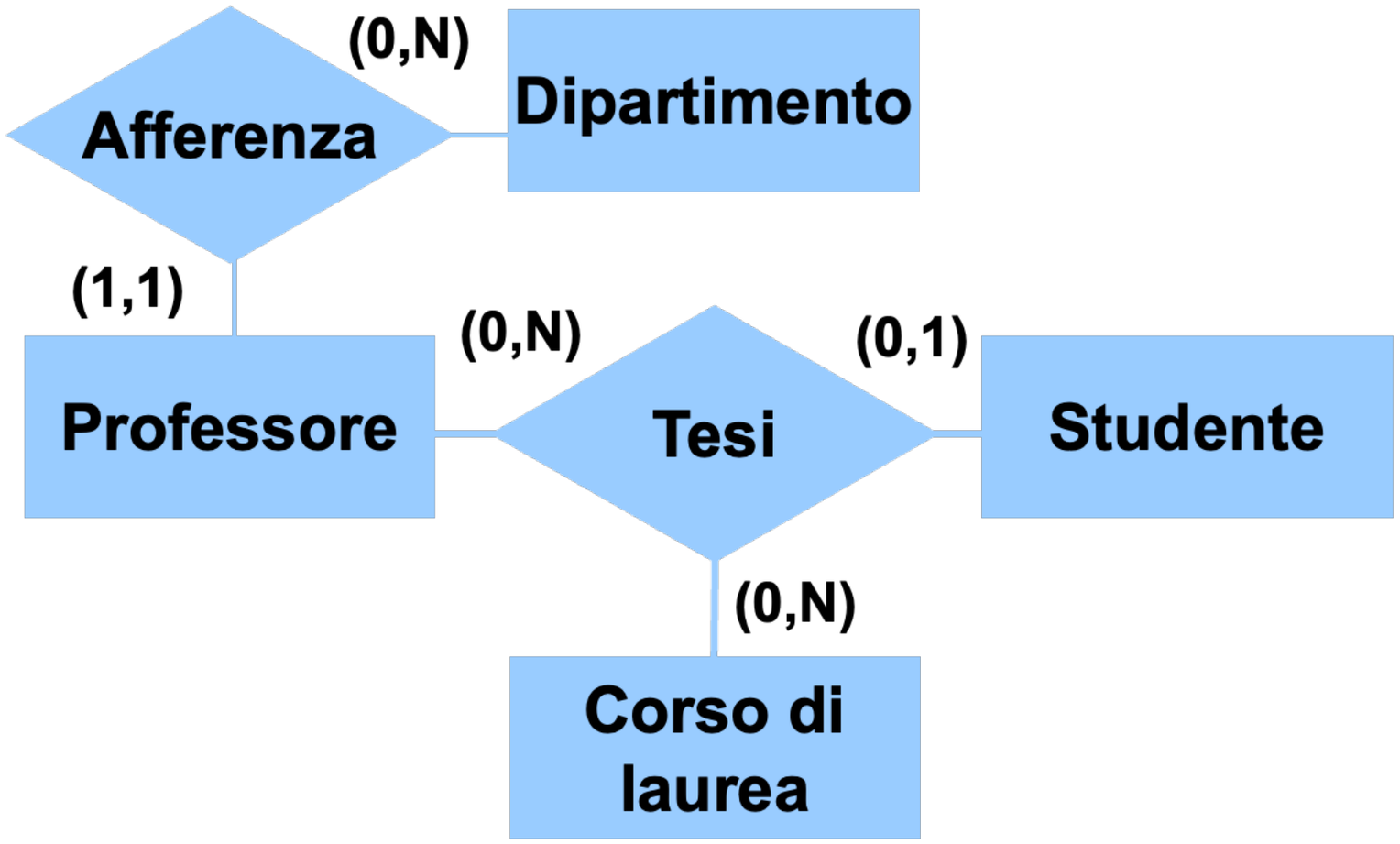
- Studente → Corso di laurea
- Studente → Professore
- Professore → Dipartimento
- Studente è chiave

Verifica di normalizzazione su *relationship*



- Studente → Corso di laurea NON VIOLA la 3NF
- Studente → Professore NON VIOLA la 3NF
- Professore → Dipartimento VIOLA la 3NF
- Studente è chiave

Verifica di normalizzazione su *relationship*



- Le due relationship Afferenza e Tesi sono in 3NF (e in BCNF)
  - Tesi lo è in virtù delle dipendenze Studente → Corso di laurea e Studente → Professore

Verifica di normalizzazione su *relationship*