Dipendenze Funzionali

Progettazione

- Abbiamo ipotizzato che gli attributi vengano raggruppati per formare uno schema di relazione **usando il buon senso** del progettista di basi di dati o **traducendo** uno schema di base di dati da un modello dei dati concettuale (E-R), **presumibilmente ben fatto**
- Ma abbiamo bisogno di **misurare formalmente** perché un raggruppamento di attributi in uno schema di relazione possa essere **migliore** di un altro
- **Obiettivo**: valutare la qualità della progettazione degli schemi relazionali

Approccio seguito

- **Top-down**:
  - abbiamo iniziato individuando un certo numero di **raggruppamenti di attributi** per formare relazioni che sussistono come tali nel mondo reale, ad esempio una fattura, un form o un report.
  - Queste relazioni sono poi state analizzate portando eventualmente a **decomposizioni successive**.

Obiettivi impliciti del progetto logico

- La **conservazione dell'informazione**, cioè il mantenimento di tutti i concetti espressi precedentemente mediante il modello concettuale, inclusi tipi di attributi, tipi di entità e tipi di associazioni.
- La **minimizzazione della ridondanza**, cioè l'evitare la memorizzazione ripetuta della stessa informazione, e quindi la necessità di effettuare molteplici aggiornamenti al fine di mantenere la consistenza tra le diverse copie della medesima informazione.
- Possiamo derivare da questi obiettivi **alcune linee guida** per il progetto

Linea Guida 1: semplice è bello

- Uno schema di relazione deve essere progettato in modo che **sia semplice spiegarne il significato**.
- Non si devono raggruppare attributi provenienti da più tipi di entità e tipi di relazione in un'unica relazione.
- Intuitivamente, se uno schema di relazione corrisponde a **un solo tipo di entità** o a **un solo tipo di *relationship***, risulta **semplice spiegarne il significato**.
- In caso contrario, nascerà **un'ambiguità semantica** e quindi lo schema non potrà essere spiegato con facilità.

Linea Guida 2: no alle anomalie

- Gli schemi vanno progettati in modo che **non possano presentarsi anomalie** di inserimento, cancellazione o modifica.
- La **mancanza** di anomalie va **certificata** usando una **descrizione formale della semantica** dei fatti descritti in uno schema relazionale
- Se possono presentarsi anomalie, vanno chiaramente **rilevate** e si deve assicurare che i programmi che aggiornano la base di dati operino **correttamente**.

Esempio 1

- **Fattura**(CodFatt, CodProd, TotDaPagare,
  CostoNettoProd, IVA)
- Semantica attributi:
  - CodFatt determina CodProd e TotDaPagare
  - CodProd determina CostoNettoProd e IVA
  - CostoNettoProd e IVA determinano TotDaPagare

Esempio 1

- Ovviamente TotDaPagare deve essere consistente con la regola che lo lega al CostoNettoProd e all'IVA
- Inoltre a CodProd deve essere attribuita la giusta percentuale di IVA
- Questo secondo legame è esterno al DB, se cambia per legge l'IVA di un certo prodotto, questo attributo deve essere modificato; però la sua modifica si porta dietro un'altra modifica dell'attributo TotDaPagare il cui significato è interno al DB ma è legato ad IVA.
- Per evitare anomalie di inserimento o modifica conviene che TotDaPagare non ci sia nella tabella Fattura

Esempio 2

- **Anagrafe**(CF, NomePersona, ViaRes, NomeCittaRes, NumAb)
- Semantica attributi:
  - CF determina NomePersona, ViaRes e NomeCittaRes
  - NomeCittaRes determina NumAb

Esempio 2

- NumAb è ripetuto per lo stesso NomeCittaRes per quanti sono i residenti
- Il valore deve essere mantenuto consistente (uguale) per ogni persona di una stessa città
- Come si può evitare il problema?
  - Trasformando Anagrafe in due schemi separati
    - Persona(CF, NomePersona, ViaRes, NomeCittaRes)
    - ListaComuni(NomeCitta, NumAb)
    - Con vincolo di integrità referenziale su NomeCittaRes verso NomeCitta e un vincolo aggiuntivo su NumAb...

Linea Guida 3: evitare frequenti valori nulli

- Si eviti di porre in una relazione attributi i cui valori possono essere frequentemente nulli.
- Se i valori nulli sono inevitabili, ci si assicuri che si presentino solo in casi eccezionali rispetto al numero di n-uple di una relazione.

Dipendenza Funzionale

- Una **dipendenza funzionale** (*functional dependency,* FD) esprime un **legame semantico** tra **due gruppi di attributi** di uno schema di relazione R
- Una FD è una proprietà di R, non di un particolare stato valido r di R
- Una FD non può essere dedotta a partire da uno stato valido r, ma deve essere definita esplicitamente da qualcuno che conosce la semantica degli attributi di R

Forme Normali

- Una **forma normale** è una **proprietà** di una base di dati relazionale che ne **garantisce la "qualità"**, cioè **l'assenza di determinati difetti**
- Quando una relazione non è normalizzata:
  - presenta **ridondanze**
  - si presta a **comportamenti poco desiderabili** durante gli aggiornamenti
- Le forme normali sono di solito definite sul **modello relazionale**, ma hanno senso in altri contesti, ad esempio il modello E-R

Normalizzazione

- Procedura che permette di **trasformare schemi** non normalizzati in schemi che soddisfano una forma normale
- La normalizzazione va utilizzata come **tecnica di verifica** dei risultati della progettazione di una base di dati
- **Non costituisce una metodologia di progettazione**

Relazione con anomalie

<u>Impiegato</u>	Stipendio	<u>Progetto</u>	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

Anomalie

- Lo stipendio di ciascun impiegato è ripetuto in tutte le n-uple relative
  - **ridondanza**
- Se lo stipendio di un impiegato varia, è necessario andarne a modificare il valore in diverse n-uple
  - **anomalia di aggiornamento**
- Se un impiegato interrompe la partecipazione a tutti i progetti, dobbiamo cancellarlo
  - **anomalia di cancellazione**
- Un nuovo impiegato senza progetto non può essere inserito
  - **anomalia di inserimento**

Causa dei problemi

- Abbiamo usato **un'unica relazione** per rappresentare **informazioni eterogenee**
  - gli impiegati con i relativi stipendi
  - i progetti con i relativi bilanci
  - le partecipazioni degli impiegati ai progetti con le relative funzioni
- Ora useremo il concetto di dipendenza funzionale per studiare meglio questi problemi

Definizione di dipendenza funzionale

- Dati:
  - una relazione r su $R(X)$,
  - due sottoinsiemi **non vuoti** Y e Z di X,
- esiste in r una **dipendenza funzionale** da Y a Z se, per ogni coppia di n-uple t_1 e t_2 di r con gli stessi valori su Y, risulta che t_1 e t_2 hanno gli stessi valori anche su Z
- Notazione: $Y \rightarrow Z$
  - Nota: Se $Y \rightarrow Z$, non è detto che esista $Z \rightarrow Y$

Dipendenze funzionali particolari

- Una dipendenza funzionale è **completa** quando $Y \to Z$ e, per ogni $W \subset Y$, non vale $W \to Z$
- Se Y è una **superchiave** di $R(X)$, allora Y determina ogni altro attributo della relazione, i.e., $Y \to X$
- Se Y è una **chiave**, allora $Y \to X$ è una dipendenza funzionale completa
- Una dipendenza funzionale è **banale** se è sempre soddisfatta
  - $Y \to Y$ è banale
  - $Y \to A$ è non banale se $A \notin Y$
  - $Y \to Z$ è non banale se nessun attributo di Z appartiene a Y

Esempio 1

- Caratterizziamo in termini di dipendenze le informazioni semantiche che abbiamo
  - Ogni impiegato ha un solo stipendio
    - Impiegato → Stipendio
  - Ogni progetto ha un solo bilancio
    - Progetto → Bilancio
  - Ogni impiegato ha una sola funzione per progetto
    - Impiegato, Progetto → Funzione

Esempio 1

<u>Impiegato</u>	Stipendio	<u>Progetto</u>	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

Legami tra dipendenze funzionali e anomalie

<u>Impiegato</u>	Stipendio	<u>Progetto</u>	Bilancio	Funzione

- Impiegato → Stipendio
  - Ci sono ripetizioni
- Progetto → Bilancio
  - Ci sono ripetizioni
- Impiegato, Progetto → Funzione
  - Non ci sono ripetizioni
- Impiegato non è una chiave
- Progetto non è una chiave
- Impiegato, Progetto è una chiave

Legami tra dipendenze funzionali e anomalie

- Le dipendenze funzionali sono usate per **verificare l'eventuale presenza di anomalie** in un progetto
  - Vedremo che sono usate anche per "normalizzare" uno schema
- Data la loro importanza, quando necessario indicheremo con $R(X, F)$ uno schema di relazione $R(X)$ che **verifica un insieme di dipendenze funzionali** F

Implicazione

- Sia F un insieme di dipendenze funzionali definite su $R(Z)$ e sia $X \rightarrow Y$
  - Si dice che F **implica (logicamente)** $X \rightarrow Y$, in simboli $F \models X \rightarrow Y$, se, **per ogni possibile istanza** r di R che verifica tutte le dipendenze funzionali in F, risulta verificata **anche** la dipendenza funzionale $X \rightarrow Y$
  - Si dice anche che $X \rightarrow Y$ **è implicata (logicamente) da** F
- Esempio:
  - R(Impiegato, Categoria, Stipendio)
  - Le dipendenze funzionali
    - Impiegato → Categoria e
    - Categoria → Stipendio
  - implicano la dipendenza funzionale
    - Impiegato → Stipendio

Problema

- La definizione di implicazione **non è direttamente utilizzabile** nella pratica
  - Essa prevede una **quantificazione universale** sulle istanze della base di dati (“per ogni istanza ...”)
  - **Non abbiamo un algoritmo** per calcolare tutte le dipendenze funzionali implicate da un insieme F
- **Armstrong** (1974) ha fornito delle **regole di inferenza** che permettono di **derivare costruttivamente** tutte le dipendenze funzionali che sono implicate da un dato insieme iniziale

Regole di inferenza di Armstrong

1. **Riflessività**:

$$\text{Se } Y \subseteq X \text{ allora } X \rightarrow Y$$

2. **Additività** (o **arricchimento**):

$$\text{Se } X \rightarrow Y \text{ allora } XZ \rightarrow YZ \text{ per qualunque } Z$$

3. **Transitività**:

$$\text{Se } X \rightarrow Y \text{ e } Y \rightarrow Z \text{ allora } X \rightarrow Z$$

Derivazione

- Dati:
  - un **insieme** di **regole di inferenza** RI,
  - un **insieme** di **dipendenze funzionali** F e
  - una **dipendenza funzionale** f,
- una **derivazione di** f **da** F **secondo RI** è una **sequenza** finita $f_1, \ldots, f_m$ dove
  - $f_m = f$
  - ogni f_i è un elemento di F oppure è ottenuta dalle precedenti dipendenze $f_1, \ldots, f_{i-1}$ della derivazione usando una regola di inferenza RI
- Indichiamo con $F \vdash X \rightarrow Y$ il fatto che la **dipendenza funzionale** $X \rightarrow Y$ sia **derivabile** da F **usando** RI

Regole di derivazione comuni

- **Unione**:

$$\{X \to Y, X \to Z\} \vdash X \to YZ$$

- **Decomposizione**:

$$\{X \to YZ\} \vdash X \to Y$$

- **Indebolimento**:

$$\{X \to Y\} \vdash XZ \to Y$$

- **Identità**:

$$\{\} \vdash X \to X$$

Unione: dimostrazione

- **Unione**:

$$\{X \to Y, X \to Z\} \vdash X \to YZ$$

- *Dimostrazione*:

1. $X \to Y$ per ipotesi
2. $X \to XY$ per additività da 1
3. $X \to Z$ per ipotesi
4. $XY \to YZ$ per additività da 3
5. $X \to YZ$ per transitività da 2 e 4

Decomposizione: dimostrazione

- **Decomposizione**:

$$\{X \to YZ\} \vdash X \to Y$$

- *Dimostrazione*:

1. $X \to YZ$ per ipotesi
2. $YZ \to Y$ per riflessività
3. $X \to Y$ per transitività da 1 e 2

Indebolimento: dimostrazione

- **Indebolimento**:

$$\{X \to Y\} \vdash XZ \to Y$$

- *Dimostrazione*:

1. $XZ \to X$ per riflessività
2. $X \to Y$ per ipotesi
3. $XZ \to Y$ per transitività da 1 e 2

Chiusura di un insieme di attributi

- Dato uno schema $R(T,F)$ con $X \subseteq T$, **la chiusura di** X **rispetto a** F, indicata col simbolo X_F^+, è definita come

$$X_F^+ = \{A \in T | \, F \vdash X \to A\}$$

- Se non vi sono ambiguità scriveremo semplicemente X^+
- *Ritorneremo avanti su questa definizione, con qualche esempio*

Teorema della chiusura degli attributi

- **Teorema**:

$$F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^+$$

Correttezza e Completezza

- Dato un qualche insieme di regole di inferenza RI e un insieme di dipendenze funzionali F
  - RI è **corretto** se

$$F \vdash X \to Y \Rightarrow F \models X \to Y$$

  - *Applicando RI a un insieme F di dipendenze funzionali, si ottengono solo dipendenze logicamente implicate da F*
  - RI è **completo** se

$$F \models X \to Y \Rightarrow F \vdash X \to Y$$

  - *Applicando RI a un insieme F di dipendenze funzionali, si ottengono tutte le dipendenze logicamente implicate da F*

Teorema

- **Le regole di inferenza di Armstrong sono corrette e complete**

Teorema

- **Le regole di inferenza di Armstrong sono corrette e complete**
- Questo teorema ci permette di scambiare $\models$ con $\vdash$ ovunque. In particolare nella definizione di chiusura degli attributi, cioè
$$X_F^+ = \{A \in T | F \models X \rightarrow A\}$$
- Si può dimostrare che le regole di inferenza di Armstrong sono **minimali**, cioè **ignorando** anche una sola di esse, l'insieme di regole che rimangono **non è più completo.**
- Le regole di inferenza di Armstrong non sono l'unico insieme di regole corretto e completo!

Chiusura di un insieme di dipendenze funzionali

- Sia F un insieme di dipendenze funzionali definite su $R(Z)$
  - La **chiusura di** F è l'insieme F^+ di **tutte** le dipendenze funzionali implicate da F:

$$F^+ = \{X \rightarrow Y | F \Rightarrow X \rightarrow Y\}$$

- Dato un insieme di dipendenze funzionali F definite su $R(Z)$, un'istanza r di R che soddisfa F soddisfa anche le dipendenze funzionali di F^+

Calcolo di F^+

- Possiamo usare le regole di Armstrong per calcolare F^+

  Input: $R(T, F)$

  Output: F^+

  $F^+ \leftarrow F$

  while (F^+ non cambia) **do**

      for each $f \in F^+$ **do**

          applicare riflessività e additività a f e aggiungere a F^+ le dipendenze ottenute

      for each $f_1, f_2 \in F^+$ **do**

          se possibile, applicare transitività a f_1 e f_2 e aggiungere a F^+ la dipendenza ottenuta

  return F^+

Esempio 2

- Dati
  - $R(ABCGHI)$
  - $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- Alcuni membri di F^+ sono:
  - $A \rightarrow H$
    - per transitività da $A \rightarrow B$ e $B \rightarrow H$
  - $AG \rightarrow I$
    - arricchendo $A \rightarrow C$ con G e per transitività con $CG \rightarrow I$
  - $CG \rightarrow HI$
    - arricchendo $CG \rightarrow I$ con CG, arricchendo $CG \rightarrow H$ con I e per transitività

F^+ e X^+

- Il calcolo di F^+ è **molto costoso**
  - **complessità esponenziale** nel numero di attributi dello schema nel caso peggiore
- Spesso però quello che ci interessa è **verificare** se F^+ contiene una certa dipendenza e **non generare** l'intera chiusura
- Per fare ciò basta calcolare X^+ per il teorema di chiusura degli attributi

$$F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^+$$

$$(F \models X \rightarrow Y \Leftrightarrow Y \subseteq X^+)$$

Calcolo di X^+

Input: $R(T, F)$, $X \subseteq T$

Output: X^+

$X^+ \leftarrow X$

while (X^+ non cambia) **do**

  for each $W \rightarrow V \in F$ **do**

    if $W \subseteq X^+$ **and** $V \nsubseteq X^+$ **then**

      $X^+ \leftarrow X^+ \cup V$

return X^+

Esempio 3

- Dati
  - $R(ABCDE)$
  - $F = \{A \to B, BC \to D, B \to E, E \to C\}$
- Calcoliamo A^+:
  - $A^+ \leftarrow A$
  - $A^+ \leftarrow AB$ perché $A \to B$ e $A \subseteq A^+$
  - $A^+ \leftarrow ABE$ perché $B \to E$ e $B \subseteq A^+$
  - $A^+ \leftarrow ABEC$ perché $E \to C$ e $E \subseteq A^+$
  - $A^+ \leftarrow ABECD$ perché $BC \to D$ e $BC \subseteq A^+$
- Possiamo concludere che A è superchiave (e anche chiave)

Chiavi

- Dato uno schema $R(T, F)$
  - Un insieme di attributi $K \subseteq T$ si dice **superchiave** di R se la dipendenza funzionale $K \rightarrow T$ è implicata da F, ovvero se $K \rightarrow T \in F^+$
  - Un insieme di attributi $K \subseteq T$ si dice **chiave** di R se K è una superchiave di R e se non esiste alcun sottoinsieme proprio di K che sia superchiave di R
- Dato che in uno schema ci possono essere più chiavi, di solito ne viene scelta una, detta **chiave primaria**, come identificatore delle n-uple delle istanze dello schema

Trovare tutte le chiavi

- Il problema di **trovare tutte le chiavi** di una relazione $R(Z)$ richiede un algoritmo di **complessità esponenziale** nel caso pessimo
- Cosa si deve fare:
  - Gli attributi che stanno solo a sinistra stanno in tutte le chiavi, chiamiamo N questo insieme
  - Gli attributi che stanno solo a destra non stanno in nessuna chiave
  - Si aggiunge a N un attributo alla volta tra quelli che non stanno solo a destra, poi una coppia di attributi e così via, chiamiamo X_i questo sottoinsieme di attributi, ogni volta si controlla se la dipendenza $N \cup X_i \rightarrow Z$ esiste

Verificare una chiave

- L'algoritmo per il calcolo della chiusura di un insieme di attributi può essere usato per **verificare se un insieme di attributi è chiave o superchiave**
- $X \subseteq T$ è superchiave di $R(T, F)$
  - se e solo se $X \rightarrow T \in F^+$, ovvero
  - se e solo se $T \subseteq X^+$
- $X \subseteq T$ è chiave di $R(T, F)$
  - se e solo se $T \subseteq X^+$, e non esiste $Y \subset X$ tale che $T \subseteq Y^+$

Equivalenza

- **Due insiemi di dipendenze funzionali** F e G sugli attributi T di una relazione $R(T)$ sono **equivalenti**, in simboli $F \equiv G$, se e solo se $F^+ = G^+$
  - Se $F \equiv G$ allora F è una **copertura** di G e viceversa
- La relazione di equivalenza permette di stabilire se due schemi di relazione rappresentano **gli stessi fatti**
  - Basta che abbiano gli stessi attributi e dipendenze funzionali equivalenti
- Per **verificare l'equivalenza** è sufficiente che
  - tutte le dipendenze di F appartengano a G^+
  - tutte le dipendenze di G appartengano a F^+

Esempio 4

- Verificare se F e G sono equivalenti:
  - $F = \{A \to C, AC \to D, E \to AD, E \to H\}$
  - $G = \{A \to CD, E \to AH\}$
- Verificare che tutte le dipendenze di F appartengano a G^+
  - $A \to CD \Rightarrow A \to C$
  - $A \to CD \Rightarrow AC \to CD \Rightarrow AC \to D$
  - $E \to AH \Rightarrow E \to H$
  - $E \to AH \Rightarrow E \to A \Rightarrow E \to AE$
  - $A \to CD \Rightarrow A \to D \Rightarrow A \to AD \Rightarrow AE \to ADE$
  - $E \to ADE \Rightarrow E \to AD$

Esempio 4

- Verificare se F e G sono equivalenti:
  - $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
  - $G = \{A \rightarrow CD, E \rightarrow AH\}$
- Verificare che tutte le dipendenze di G appartengano a F^+
  - $A \rightarrow C \Rightarrow A \rightarrow AC, AC \rightarrow D \Rightarrow A \rightarrow D \Rightarrow A \rightarrow CD$
  - $E \rightarrow AD \Rightarrow E \rightarrow A, E \rightarrow H \Rightarrow E \rightarrow AH$

Esempio 5

- Verificare se F e G sono equivalenti:
  - $F = \{A \to C, AC \to D, E \to AD, E \to H\}$
  - $G = \{A \to CD, E \to AH\}$
- Invece di verificare se $X \to Y \in F$ è anche in G^+ e viceversa, possiamo verificare se $Y \subseteq X_G^+$ e, viceversa, per ogni dipendenza funzionale
- Per esempio (verifichiamo F su X_G^+):
  - $A \to C$: $A_G^+ = ACD$, quindi $C \in A_G^+$
  - $AC \to D$: $AC_G^+ = ACD$, quindi $D \in AC_G^+$
  - $E \to AD$: $E_G^+ = EADCH$, quindi $AD \in E_G^+$
  - $E \to H$: $E_G^+ = EADCH$, quindi $H \in E_G^+$

Esempio 5

- Verificare se F e G sono equivalenti:
  - $F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
  - $G = \{A \rightarrow CD, E \rightarrow AH\}$
- Invece di verificare se $X \rightarrow Y \in F$ è anche in G^+ e viceversa, possiamo verificare se $Y \subseteq X_G^+$ e, viceversa, per ogni dipendenza funzionale
- Per esempio (verifichiamo G su X_F^+):
  - $A \rightarrow CD$: $A_F^+ = ACD$, quindi $CD \in A_F^+$
  - $E \rightarrow AH$: $E_F^+ = EADHC$, quindi $AH \in E_F^+$

Ridondanza

- Sia F un insieme di dipendenze funzionali
- Data $X \to Y \in F$, X contiene un **attributo estraneo** $A \in X$ se e solo se $\big(F - \{X \to Y\}\big) \cup \big(X - \{A\} \to Y\big) \equiv F$.
- $X \to Y$ è una **dipendenza ridondante** se e solo se $\big(F - \{X \to Y\}\big) \equiv F$, o, in altre parole, se e solo se $X \to Y \in \big(F - \{X \to Y\}\big)^+$
- Le dipendenze che **non contengono attributi estranei** e la cui **parte destra è un unico attributo** sono dette **dipendenze elementari**

Esempio 6

- Sia $F = \{AB \to C, B \to A, C \to A\}$
  - L'unica dipendenza che può avere un attributo estraneo è $AB \to C$
    - $A^+ = \{A\}$ e $B^+ = \{B, A, C\}$, quindi A è un **attributo estraneo** in $AB \to C$
- Quindi $F \equiv G_1 = \{B \to C, B \to A, C \to A\}$
  - $\{B \to C, C \to A\}^+ = G$, quindi $B \to A$ è una **dipendenza ridondante**
- Quindi $F \equiv G_1 \equiv G_2 = \{B \to C, C \to A\}$
  - Se tentiamo di eliminare la dipendenza ridondante prima di eliminare l'attributo estraneo, non ci riusciamo, quindi **l'ordine** delle due attività **è importante**

Copertura minimale

- Sia F un insieme di dipendenze funzionali
- F è una **copertura minimale** se e solo se:
  - ogni parte destra di una dipendenza ha un unico attributo;
  - le dipendenze non contengono attributi estranei;
  - non esistono dipendenze ridondanti.
- In alcuni testi una copertura minimale è chiamata:
  - *insieme minimale*
  - *copertura canonica*
- Nell'esempio precedente, G_2 è una copertura minimale.

Calcolo della copertura minimale

Input: insieme di dipendenze funzionali F

Output: copertura minimale G di F

$G \leftarrow F$

for each $X \rightarrow Y \in G$ **do**

  $Z \leftarrow X$

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

    if $Y \in \left(Z - (A)\right)^+_F$ **then**

      $Z \leftarrow Z - \{A\}$

  $G \leftarrow \left(G - \{X \rightarrow Y\}\right) \cup \{Z \rightarrow Y\}$

for each $f \in G$ **do**

  if $f \in \left(G - \{f\}\right)^+$ **then**

    $G \leftarrow G - \{f\}$

return G

Calcoliamo gli attributi estranei delle dipendenze

Eliminiamo gli attributi estranei delle dipendenze

Eliminiamo le dipendenze ridondanti

Copertura minimale

- Il precedente algoritmo dimostra il seguente teorema.
- **Teorema**:
  - Per ogni insieme di dipendenze funzionali F esiste una copertura minimale
- Si noti che il teorema nulla dice sull'**unicità** della copertura minimale
- Infatti, per $F = \{AB \to C, A \to B, B \to A\}$,
  - $\{A \to C, A \to B, B \to A\}$ è una copertura minimale
  - $\{B \to C, A \to B, B \to A\}$ è una copertura minimale

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** $X \rightarrow Y \in F$, 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** $X \rightarrow A \in F$, 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**

Controlliamo ogni dipendenza funzionale della relazione

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

Se X non è superchiave

Se la dipendenza funzionale è non banale

    return false

return true

Esempio 7

<u>Impiegato</u>	Stipendio	<u>Progetto</u>	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 $X \rightarrow Y$ 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 $\rightarrow$ Stipendio
- Progetto $\rightarrow$ Bilancio
- Impiegato, Progetto $\rightarrow$ 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 $\rightarrow$ 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 → Sede
- Progetto → 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

Decomposizione di schemi

- Dato uno schema $R(T)$, l'insieme di schemi $\rho = \left\{R_1(T_1), \ldots, R_k(T_k)\right\}$ 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 dipendenze**

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

$$T \rightarrow C$$

$$C \rightarrow P$$

questa decomposizione preserva i dati

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

P	T
p1	t1
p1	t2
p1	t3

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

T	C
t1	c1
t2	c2
t3	c2

questa decomposizione non preserva la dipendenza

$$C \rightarrow P$$

perché gli attributi sono in relazioni diverse

Teorema della perdita di dati

- **Teorema**:
  - Se $\rho = \{R_1(T_1), \ldots, R_k(T_k)\}$ 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 $\rho = \left\{R_1(T_1), \ldots, R_k(T_k)\right\}$, 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 $\rho = \{R_1(T_1), R_2(T_2)\}$ 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

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

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

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
$$\left(\pi_X(F) \cup \pi_Y(F)\right)^+ = F^+$$
- 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 $X \rightarrow Y \in F$, calcoliamo X_G^+ e verifichiamo se $Y \in X_G^+$
  - Per ogni $X \rightarrow Y \in G$, 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 $\rho = \{\text{Telefoni}\}$
- 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
  - $R_1\left(\{\text{Numero}, \text{Località}\}, \{\,\}\right)$
  - $R_2\left(\{\text{Località}, \text{Prefisso}\}, \{\text{Località} \rightarrow \text{Prefisso}\}\right)$
- 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

$$\text{Dirigente} \rightarrow \text{Sede}$$

$$\text{Progetto, Sede} \rightarrow \text{Dirigente}$$

- Questa relazione è in BNCF?

Esempio 10

- Applichiamo l'algoritmo di verifica!

Dirigente	<u>Progetto</u>	<u>Sede</u>
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

$$\text{Progetto, Sede} \rightarrow \text{Dirigente} \quad \checkmark$$

$$\text{Dirigente} \rightarrow \text{Sede} \quad ✗$$

Esempio 10

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

Dirigente	<u>Progetto</u>	<u>Sede</u>
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

Dirigente $\rightarrow$ Sede

Progetto, Sede $\rightarrow$ Dirigente

- 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** $X \rightarrow A \in F^{+}$, 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	<u>Progetto</u>	<u>Sede</u>
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

$$\text{Progetto, Sede} \rightarrow \text{Dirigente}$$

$$\text{Dirigente} \rightarrow \text{Sede}$$

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

Esempio 11

Dirigente	<u>Progetto</u>	<u>Sede</u>
Rossi	Marte	Roma
Verdi	Giove	Milano
Verdi	Marte	Milano
Neri	Saturno	Milano
Neri	Venere	Milano

$$\text{Progetto, Sede} \rightarrow \text{Dirigente}$$

$$\text{Dirigente} \rightarrow \text{Sede}$$

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

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
  - $$\begin{aligned} F \equiv F_1 = \{&ABC \to D, ABC \to E, ABC \to G, BD \to A, \\ &BD \to C, BD \to E, C \to B, C \to H, \\ &H \to B, H \to D, H \to E\} \end{aligned}$$
  - 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
  - $$\begin{aligned} F_2 \equiv F_1 = \{&C \to D, C \to E, C \to G, BD \to A, \\ &BD \to C, BD \to E, C \to B, C \to H, \\ &H \to B, H \to D, H \to E\} \end{aligned}$$

Esempio 13

- $$\begin{aligned} F_2 \equiv F_1 = \{C \to D, C \to E, C \to G, BD \to A, \\ BD \to C, BD \to E, C \to B, C \to H, \\ H \to B, H \to D, H \to E\} \end{aligned}$$
- 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 $C \to H \to B$
- $$\begin{aligned} G \equiv F_2 = \{C \to G, BD \to A, BD \to C, \\ C \to H, H \to B, H \to D, H \to E\} \end{aligned}$$

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

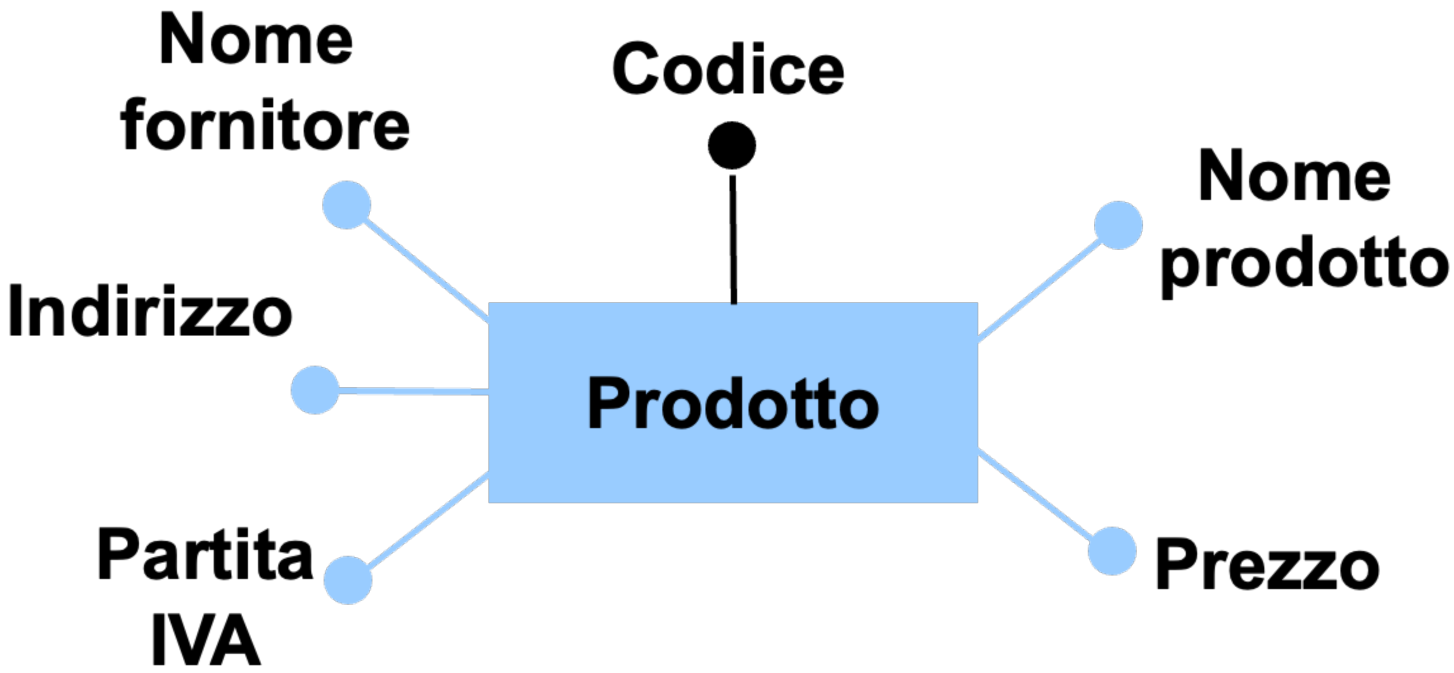
Esempio 14

- $$G \equiv F_2 = \{A \to C, A \to E, CE \to A,$$
  $$A \to G, G \to B, G \to D\}$$
- 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
- $\rho = \left\{R_1(ACEG), R_3(GBD), R_0(AH)\right\}$ è in 3NF

Progettazione e Normalizzazione

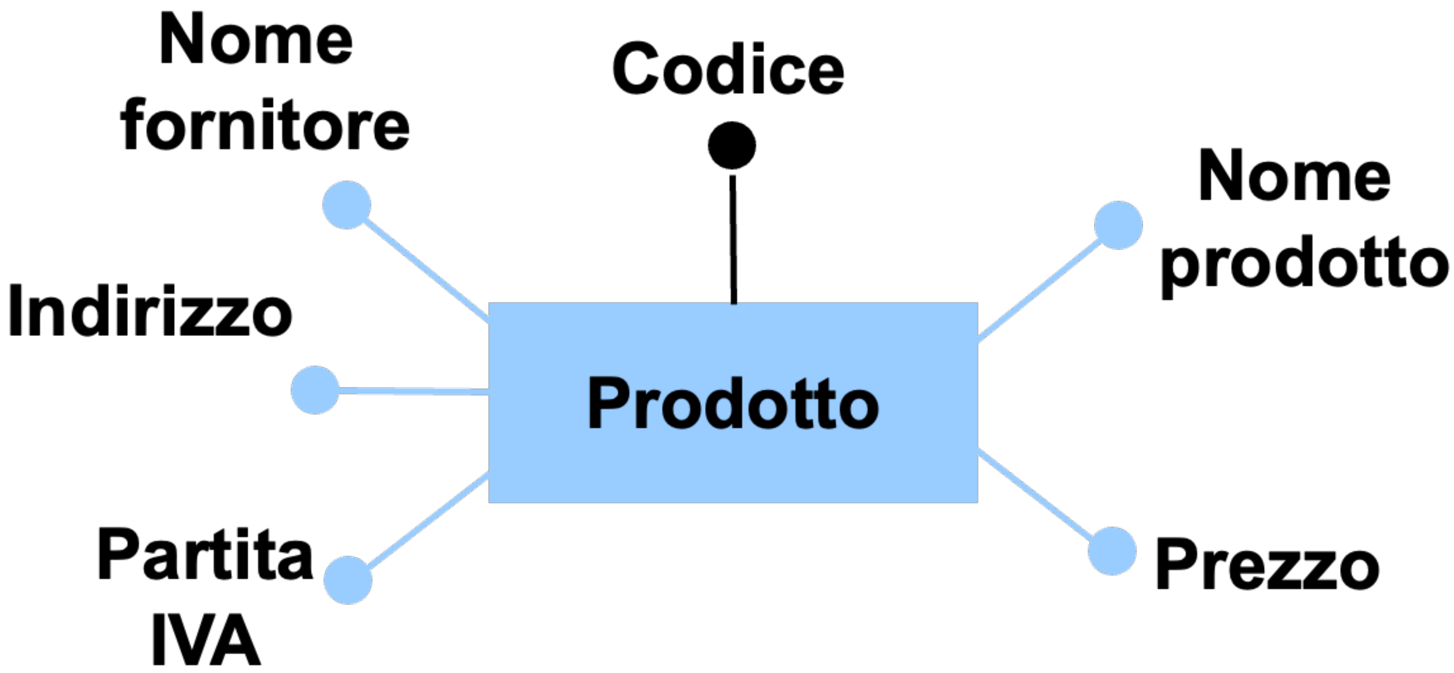
- La teoria della normalizzazione serve per verificare la qualità dello schema logico
- Ma si può usare anche durante la progettazione concettuale per ottenere uno schema di buona qualità (verifica ridondanze, partizionamento di entità/ relazioni)

Verifica di normalizzazione su entità



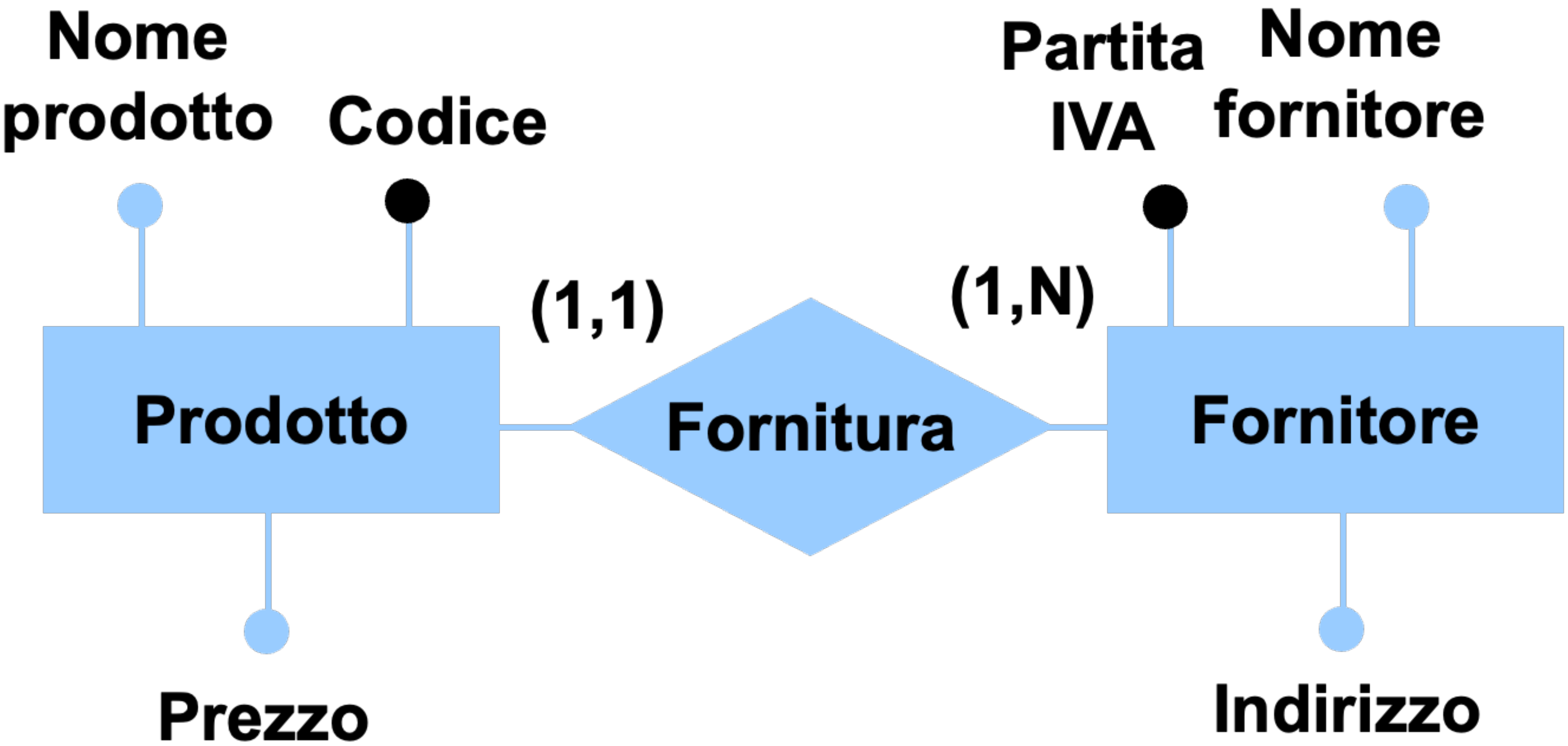
- Abbiamo la dipendenza funzionale
  - Partita IVA → Nome fornitore, Indirizzo
- Codice è chiave

Verifica di normalizzazione su entità

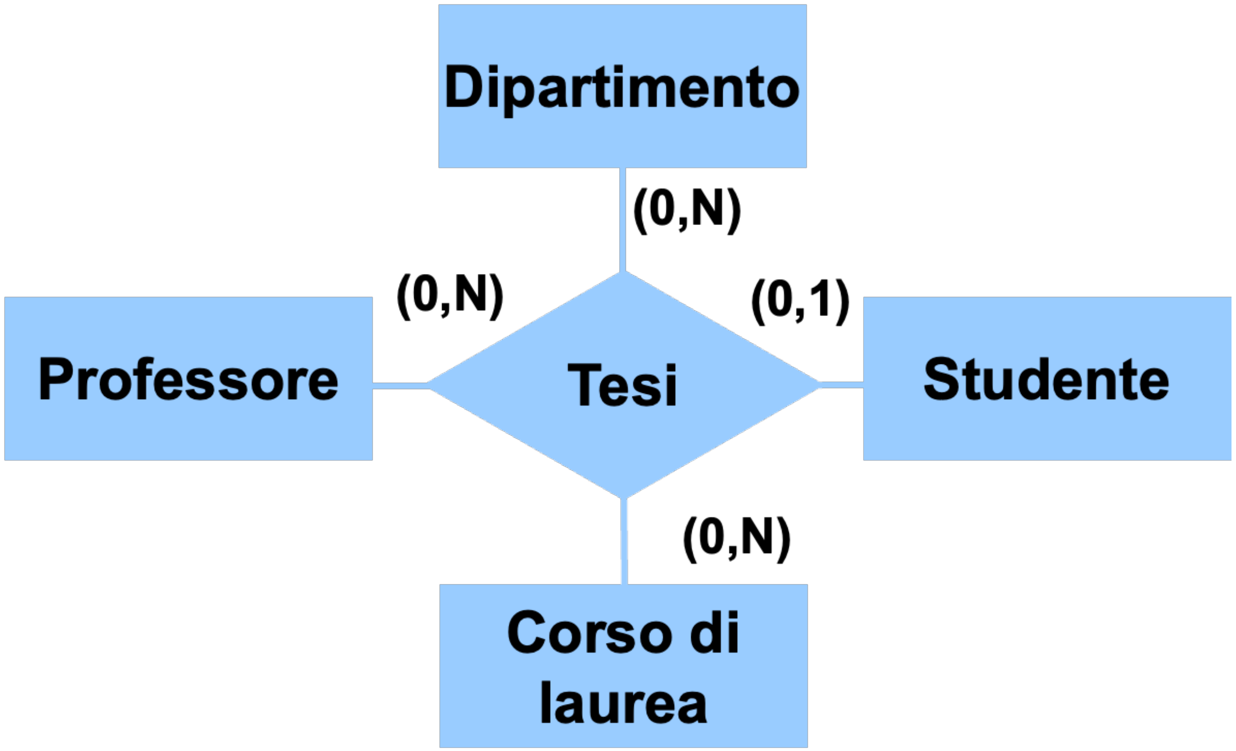


- Partita IVA → Nome fornitore, Indirizzo
  - Partita IVA non è superchiave
  - Nome fornitore e Indirizzo non fanno parte di una chiave
- L'entità viola la terza forma normale

Verifica di normalizzazione su entità

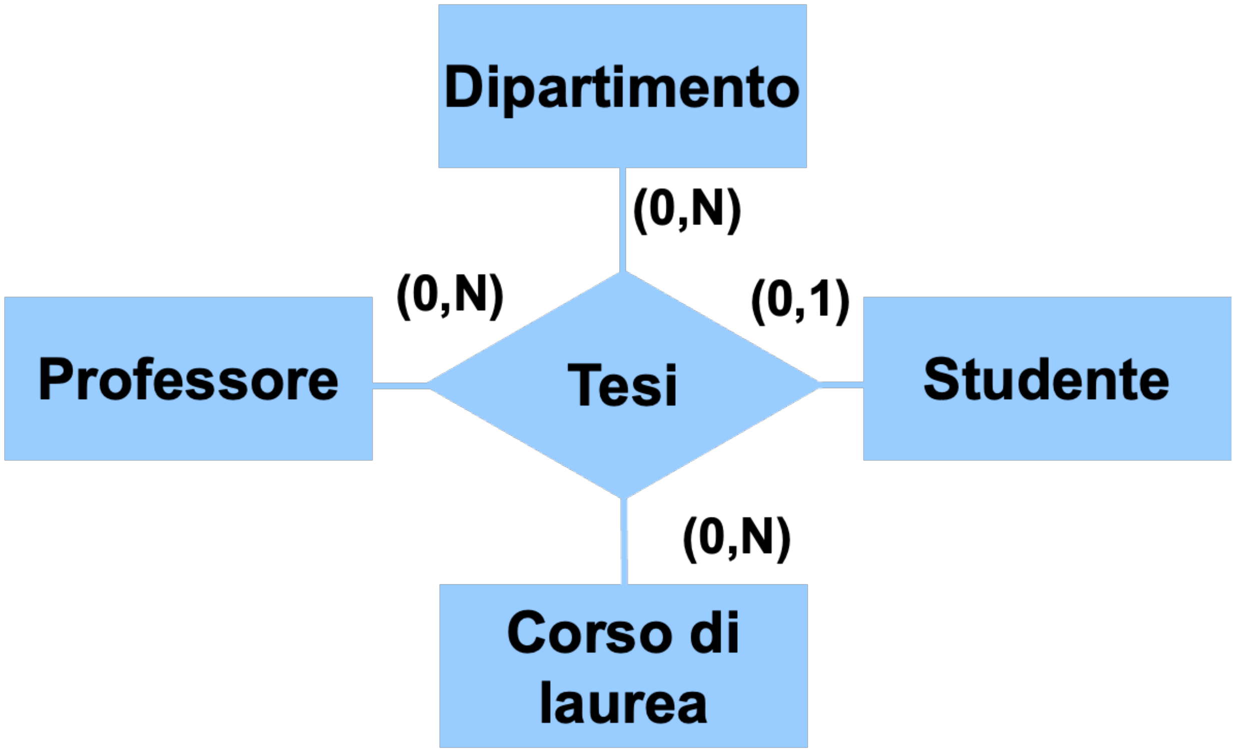


Verifica di normalizzazione su *relationship*



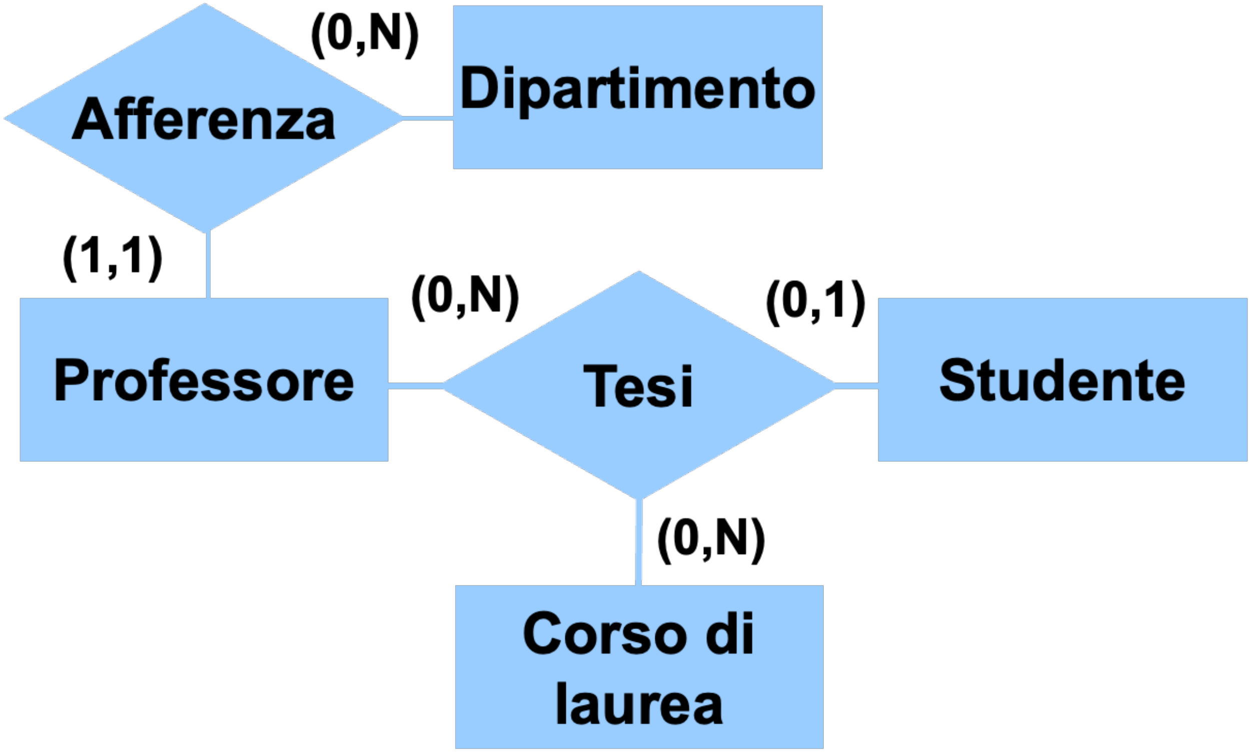
- Studente → Corso di laurea
- Studente → Professore
- Professore → Dipartimento
- Studente è chiave

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*