Normalizzazione

Le **ridondanze** sui dati possono essere di due tipi:

- **Ridondanza concettuale**: non ci sono duplicazioni dello stesso dato, ma sono memorizzate informazioni che possono essere ricavate da altre già contenute nel database
- **Ridondanza fisica**: esistono duplicazioni sui dati, che possono generare anomalie nelle operazioni sui dati

Si osservi il seguente esempio. Cosa c'è di strano in questa tabella?

Docente	Livello	Salario	Dipartimento	Direttore	Corso
Rossi	4	15000	Fisica	Neri	Mat. Discreta
Rossi	4	15000	Chimica	Rossini	Analisi I
Bianchi	3	10000	Informatica	Viola	Basi di Dati
Neri	4	15000	Informatica	Viola	Programmazione
Neri	4	15000	Matematica	Bruni	Inf. di base
Rossi	3	15000	Matematica	Bruni	Geometria

Lo stipendio di ciascun docente è ripetuto in tutte le tuple relative, portando ad una **ridondanza sui dati**. Stesso discorso per il direttore di un dipartimento.
Questo problema porta a diverse **anomalie**, di **aggiornamento** e di **cancellazione**. Se dovesse variare lo stipendio, bisogna modificare tutte le tuple del docente. Se invece un docente non ha corsi, bisogna eliminare tutti i suoi dati.

Queste problematiche sono causate dall'utilizzo di un'**unica tabella per rappresentare informazioni eterogenee**.
L'errore potrebbe derivare da una **traduzione non corretta nel modello logico relazionale**, oppure da **errori durante la progettazione concettuale**.

Per risolvere le anomalie viste fin qui si introduce un nuovo concetto del modello relazionale, la **Dipendenza Funzionale (DF)**:

Definizione informale
Data una tabella su uno schema $R(X)$ e due attributi Y e Z di X. Esiste la *dipendenza funzionale* $Y \rightarrow Z$ se per ogni coppia di tuple $t1$ e $t2$ di r con $t1[Y] = t2[Y]$, si ha anche che $t1[Z] = t2[Z]$.

Definizione informale
Data una tabella su uno schema $R(X)$ e due liste di attributi $Y = \{Y_0, Y_1, \ldots, Y_n\}$ e $Z = \{Z_0, Z_1, \ldots, Z_n\}$. Esiste la *dipendenza funzionale* $Y \rightarrow Z$ se per ogni coppia di tuple $t1$ e $t2$ di r con $t1[Y] = t2[Y]$, si ha anche che $t1[Z] = t2[Z]$.

Si osservi un esempio:

Impiegato	Stipendio	Progetto	Sede	Ruolo
Rossi	20000	Marte	Roma	Tecnico
Verdi	35000	Giove	Bologna	Tecnico
Verdi	35000	Venere	Milano	Progettista
Neri	55000	Venere	Milano	Direttore
Neri	55000	Giove	Bologna	Direttore
Neri	55000	Marte	Roma	Tecnico
Bianchi	48000	Venere	Milano	Consulente

Le diverse dipendenze funzionali sono:

- ***Impiegato*** –> ***Stipendio***: ogni impiegato ha un unico stipendio
- ***Progetto*** –> ***Sede***: ogni progetto ha un'unica Sede
- ***Impiegato*** –> ***Impiegato***: per definizione stessa di dipendenza funzionale
- ***Impiegato Progetto*** –> ***Ruolo***: un impiegato può coprire un solo ruolo per progetto

Nota Bene: le dipendenze funzionali sono definite a **livello di schema** e non a livello di istanza. Inoltre, hanno sempre un **verso**.

Le dipendenze funzionali sono una **generalizzazione del vincolo di chiave (e di superchiave)**.

Definizione informale
Data una tabella con schema *R(X)*, con superchiave **K**. Esiste un vincolo di dipendenza funzionale tra K e qualsiasi attributo della tabella o combinazione degli stessi

$$K \text{ -> } X_1, \quad X_1 \subseteq X$$

Si osservi un esempio:

Impiegato	Stipendio	Progetto	Sede	Ruolo

Impiegato, Progetto è una **(super)chiave della relazione**, non possono quindi esistere due tuple con lo stesso valore della coppia <Impiegato, Progetto>.
Le dipendenze funzionali sono:

- ***Impiegato Progetto*** –> ***Stipendio***
- ***Impiegato Progetto*** –> ***Sede***
- ***Impiegato Progetto*** –> ***Ruolo***
- ***Impiegato Progetto*** –> ***Sede Ruolo***
- ...
- ***Impiegato Progetto*** –> ***Impiegato Stipendio Progetto Sede Ruolo***

Impiegato	Stipendio	Progetto	Sede	Ruolo
Rossi	20000	Marte	Roma	Tecnico
Verdi	35000	Giove	Bologna	Tecnico
Verdi	35000	Venere	Milano	Progettista
Neri	55000	Venere	Milano	Direttore
Neri	55000	Giove	Bologna	Direttore
Neri	55000	Marte	Roma	Tecnico
Bianchi	48000	Venere	Milano	Consulente

In questo caso le dipendenze funzionali sono:

- **DF1:** ***Impiegato*** $->$ ***Stipendio***
- **DF2:** ***Progetto*** $->$ ***Sede***
- **DF3:** ***Impiegato Progetto*** $->$ ***Ruolo***

Le dipendenze funzionali **DF1** e **DF2** sono *cattive*, portando ridondanza sui dati, possibili anomalie(aggiornamento, cancellazione, ecc) nelle operazioni sui dati.
La dipendenza funzionale **DF3** invece è una dipendenza *buona*, non determina ridondanza sui dati.

Il motivo per il quale le prime due dipendenze funzionali sono *cattive* è la non presenza di una (super)chiave. Come si può notare, **DF3** ha sulla sinistra una **(super)chiave**.

Definizione informale Forma Normale di BOYCE-CODD (FNBC)
Uno schema $R(X)$ si dice in **forma normale di Boyce e Codd** se per ogni dipendenza funzionale (non ovvia) $Y -> Z$ definita su di esso, Y è una **superchiave** di $R(X)$.

Se una tabella è in **FNBC**, non presenta le anomalie e le ridondanze viste fin qui. Al contrario, se una tabella **non** è in *FNBC*, bisogna trasformarla (**normalizzarla**), se possibile, in *FNBC*. Due esempi possono essere:

Localita	Stato	Abitanti
Roma	Italia	60000000
Cambridge	UK	50000
Cambridge	US	200000
Bologna	Italia	400000
NY	US	15000000

DF: Localita Stato → Abitanti
Rispetta la FNBC!

Localita	Stato	Prefisso
Roma	Italia	0039
Cambridge	US	001
Cambridge	UK	0044
Bologna	Italia	0039
NY	US	001

DF: Stato → Prefisso
NON rispetta la FNBC!

Per **normalizzare** una tabella, si creano **tabelle separate** per ogni dipendenza funzionale, come è possibile osservare nel seguente esempio:

IMPIEGATO→ STIPENDIO

Impiegato	Stipendio
Rossi	20000
Verdi	35000
Neri	55000
Bianchi	48000

IMPIEGATO, PROGETTO → RUOLO

Impiegato	Progetto	Ruolo
Rossi	Marte	Tecnico
Verdi	Giove	Tecnico
Verdi	Venere	Progettista
Neri	Venere	Direttore
Neri	Giove	Direttore
Neri	Marte	Tecnico
Bianchi	Venere	Consulente

PROGETTO → SEDE

Progetto	Sede
Marte	Roma
Giove	Bologna
Venere	Milano

Seguendo questa logica, ci si deve chiedere se tutte le decomposizioni sono giuste, o se ci possono essere dei problemi. Si osservino degli esempi:

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

IMPIEGATO→ SEDE

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

PROGETTO→ SEDE

Progetto	Sede
Marte	Roma
Giove	Milano
Venere	Milano
Saturno	Milano

In questo caso, **DF1** implica che ogni impiegato lavora in una sola sede. **DF2** invece indica che ogni progetto ha la stessa sede.

Impiegato	Sede
Rossi	Roma
Verdi	Milano
Neri	Milano

$\triangleright\triangleleft$

Progetto	Sede
Marte	Roma
Giove	Milano
Venere	Milano
Saturno	Milano

=

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

ESEMPIO DI TUPLA SPURIA

Se combino le due tabelle della decomposizione tramite operatore di join, non ottengo la tabella di partenza! Si ha quindi una **decomposizione con perdita/aggiunta**.

Definizione informale Decomposizione senza perdita
Uno schema $R(X)$ si **decompone senza perdita** negli schemi $R1(X1)$ ed $R2(X2)$ se, per ogni possibile istanza r di $R(X)$, il join naturale delle $X1$ ed $X2$ produce la tabella di partenza

$$\pi X1(r) \triangleright\triangleleft \pi X2(r) = r$$

In caso di **decomposizione con perdite/aggiunte**, possono generarsi delle **tuple spurie dopo il join**.

Anche se una decomposizione è senza aggiunte, può comunque presentare dei problemi di

conservazione delle dipendenze.

Si ci pone quindi la seguente domanda: **tutte le decomposizioni vanno bene?**
La risposta è **No**. La decomposizione deve soddisfare **tre proprietà**:

- **Soddisfacimento della FNBC**: ogni tabella **deve essere in FNBC**
- **Decomposizione senza perdita**: il join delle tabelle decomposte **deve produrre la relazione originaria**
- **Conservazione delle dipendenze**: il join delle tabelle decomposte **deve rispettare tutte le DF dello schema originario**

Ci si pone quindi una domanda molto importante. Data una relazione non in *FNBC*, **è sempre possibile** ottenere una **decomposizione in FNBC**?
La risposta è **No**. Si consideri un controesempio:

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

- **DF1.** Progetto Sede → Dirigente
- **DF2.** Dirigente → Sede

PROBLEMA: DF1 coinvolge tutti gli attributi, nessuna decomposizione può preservare la dipendenza!

Per risolvere casi come questo, si introduce una **nuova definizione di forma normale** meno restrittiva della forma di *Boyce e Codd*.

Definizione informale Terza Forma Normale (TFN)
Una tabella r è in **terza forma normale** se per ogni dipendenza funzionale X –> A dello schema, **almeno una delle seguenti condizioni è verificata**:

- X è una superchiave di r
- A appartiene ad almeno una chiave K di r

Quindi, la tabella considerata fin qui rispetta la terza forma normale.

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

- **DF1.** Progetto Sede → Dirigente
- **DF2.** Dirigente → Sede

DF1: Progetto Sede è una chiave → **Condizione 1 soddisfatta!**

DF2: Sede è parte di una chiave → **Condizione 2 soddisfatta!**

Se la tabella è gia in *TFN*, **non è necessaria alcuna normalizzazione**. Tuttavia, le ridondanze sui dati restano.

Confrontando la *TFN* con *FNBC*, si possono notare alcune cose:

- (Svantaggi) La *TFN* è **meno restrittiva** della *FNBC*, infatti **tollera alcune ridondanze** ed anomalie sui dati e **certifica meno la qualità** dello schema ottenuto

- (Vantaggi) La *TFN* è **sempre ottenibile**, qualsiasi sia la tabella. Questo si può raggiungere attraverso l'**algoritmo di normalizzazione** in *TFN*

Algoritmo di normalizzazione in Terza Forma Normale (TFN)

Definizione informale Terza Forma Normale (TFN)
Una tabella r è in **terza forma normale** se per ogni dipendenza funzionale X –> A (non banale) dello schema, **almeno una delle seguenti condizioni è verificata**:

- X è una superchiave di r
- A appartiene ad almeno una chiave K di r

Definizione informale Dipendenza Funzionale Banale
Una dipendenza funzionale X –> Y si dice **banale** se Y è contenuto in X. Alcuni esempi possono essere:

- *Impiegato Progetto –> Impiegato*
- *Impiegato Progetto Sede –> Impiegato Progetto*

Questo genere di dipendenze funzionali non ci interessano, e non le consideriamo come tali nel resto della trattazione!

Data una relazione r con schema *R(X)* non in *TFN*, **normalizzare in TFN** vuol dire decomporre r nelle relazioni r_1, r_2, ..., r_n, garantendo che:

- Ogni r_i ($1 <= i <= n$) è in *TFN*
- La decomposizione è senza perdite
- La decomposizione conserva tutte le dipendenze F definite sullo schema *R(X)* di partenza

L'idea alla base dell'algoritmo di normalizzazione è il seguente:

- **Semplificare l'insieme di dipendenze F**, rimuovendo quelle non necessarie, e trasformando ogni dipendenza in modo che nella parte destra compaia un singolo attributo
- **Raggruppare gli attributi coinvolti nelle stesse dipendenze**, e costruire le tabelle corrispondenti
- **Assicurarsi che almeno una delle tabella prodotta contenga la chiave** della tabella originaria

Definizione informale Implicazione Funzionale
Dato un insieme di dipendenze funzionali F, ed una dipendenza funzionale f, diremo che **F implica f** se ogni tabella che soddisfa F soddisfa anche f.

Si osservi un esempio:
F: {*Impiegato –> Livello*, *Livello –> Stipendio*}
f: *Impiegato –> Stipendio*
In questo caso, **F implica f**!

Definizione informale Chiusura di una Dipendenza Funzionale
Dato uno schema *R(U)*, con un insieme di dipendenze *F*. Sia *X* un insieme di attributi contenuti in *U*. Si definisce la **chiusura di X rispetto ad F** (X_F^+) come l'insieme degli attributi che dipendono funzionalmente da *X*

$$X_F^+ = \{A \mid A \in U \quad F \text{ implica } X \text{ -> } A\}$$

Si osservi un esempio:
R = (ABCDE)
F = (A –> B, A –> C, C –> D)
Si vuole conoscere la chiusura di A: A_F^+

A_F^+ = **{B, C, D}**

Dato quindi in input **X** (attributi) e **F** (dipendenze), si ottiene in output **la chiusura di X rispetto ad F:** X_F^+
Ad esempio, per verificare se *F implica f : X –> Y*, si può calcolare la chiusura X_F^+. Se **Y appartiene ad** X_F^+, allora **F implica f**.

Inoltre, data una tabella con schema *R(U)*, l'algoritmo per determinare la chiusura X_F^+ può essere usato anche per **verificare se X è una superchiave di R**.
Infatti, dato uno schema *R(U)*, con un insieme *F* di dipendenze funzionali, allora **un insieme di attributi K è una (super)chiave di R(U) se F implica K –> U**.

Si osservi un esempio:
R = (ABCDE)
F = (A –> B, BC –> D, B –> E, E –> C)
Se *A* è una chiave allora *F* implica *A – > ABCDE*
A_F^+ = **{A, B, E, C, D} quindi A è una chiave!**

Definizione informale Insiemi di Dipendenze Equivalenti
Dati due insiemi di dipendenze funzionali F_1 ed F_2, essi si dicono **equivalenti** se F_1 implica ciascuna dipendenza di F_2 e viceversa.

Definizione informale Insiemi di Dipendenze Non Ridondanti
Dato un insieme di dipendenze funzionali *F* definito su uno schema *R(U)*, esso si dice **non ridondante** se non esiste una dipendenza *f* di *F* tale che *F-f implica f*.

Definizione informale Insiemi di Dipendenze Ridotte
Dato un insieme di dipendenze funzionali *F* definito su uno schema *R(U)*, esso si dice **ridotto** se *(1)* non è ridondante, e *(2)* non è possibile ottenere un insieme *F'* equivalente eliminando attributi dai primi membri di una o più dipendenze di *F*.

Dato uno schema *R(U)* con insieme di dipendenze *F*, per trovare una **copertura ridotta di F** si procede in **tre passi**:

STEP 1
Sostituire *F* con F_1, che ha tutti i **secondi membri composti da un singolo attributo**.

M –> RSDG, MS –> CD, G –> R, D –> S, S –> D, MPD –> AM
F_1 = **{M –> R, M –> S, M –> D, M –> G, MS –> C, MS –> D, G –> R, D –> S, S –> D, MPD –> A, MPD –> M}**

STEP 2
Eliminare gli attributi estranei.

Supponiamo di avere *F = {AB –> C, A –> B}*, e calcoliamo A_F^+.
A_F^+ = **ABC**
C dipende solo da A, quindi l'attributo *B* in *AB –> C* può essere eliminato preservando l'uguaglianza.
F_1 = **{A –> C, A –> B}**

In generale, se ho una dipendenza funzionale del tipo: **AX –> B**, per stabilire se l'attributo *A* può essere eliminato preservando l'uguaglianza **si calcola** X^+ **e si verifica se esso include *B***, nel qual caso *A* può essere eliminato dalla dipendenza.

STEP 3
Eliminare le dipendenze non necessarie.

Supponiamo di avere *F = {B –> C, B –> A, C –> A}*:
B –> A è **ridondante**, in quanto bastano le dipendenze *B –> C* e *C –> A* per capire che *A* dipende da *B*.
Formalmente, bisogna dimostrare che *F-{B –> A} implica {B –> A}, quindi verificare che:* $B^+_{F-\{B-->A\}}$ *contiene A*

In generale, per stabilire se la dipendenza del tipo *X –> A* è ridondante la si elimina da *F*. Successivaente, si calcola $X^+_{F-\{X-->A\}}$, e **si verifica se tale insieme include ancora A**. Nel caso lo includa, si elimina le dipendenza funzionale *X –> A*.

Algoritmo di normalizzazione in terza forma normale
Dati *R(U)*, ed un insieme di dipendenze *F*, l'**algoritmo di normalizzazione in terza forma normale** procede come segue:

1. **Costruire una copertura ridotta F_1 di F**
2. **Decomporre F_1 nei sottoinsiemi $F_1^{(1)}$, $F_1^{(2)}$, ..., $F_1^{(n)}$**: ad ogni sottoinsieme appartengono dipendenze con gli stessi lati sinistri
3. **Se due o più lati sinistri delle dipendenze si implicano a vicenda, si fondono i relativi insiemi**
4. **Trasformare ciascun $F_1^{(i)}$ in una tabella** $R^{(i)}$ con gli attributi contenuti in ciascuna dipendenza. Il lato sinistro diventa la chiave della relazione
5. Se nessuna relazione $R^{(i)}$ così ottenuta contiene una chiave *K* di *R(U)*, **inserire una nuova tabella** $R^{(n+1)}$ contenente gli attributi della chiave

Perchè si chiama **Terza Forma Normale (TFN)**?

- **Prima Forma Normale (PFN)**: si suppone sia rispettata
- **Seconda Forma Normale (SFN)**: variante debole della *TFN*

Procedendo per gradi, si dovrebbe **normalizzare in PFN, poi in SFN, e quindi in TFN**.

Definizione informale Seconda Forma Normale
Una relazione r con schema *R(U)* è in **Seconda Forma Normale (SFN)** quando **non presenta dipendenze parziali**, della forma: $Y -> A$, dove:

- Y è un sottoinsieme **proprio** della chiave
- A è un qualsiasi sottoinsieme di *R(U)*

Definizione informale Quarta Forma Normale
Una tabella con schema *R(U)* è in **Quarta Forma Normale (4FN)** se non presenta dipendenze multivalore non banali diverse da una chiave della tabella.

Definizione informale Quinta Forma Normale
Una tabella con schema *R(U)* è in **Quinta Forma Normale (5FN)** se non è possible decomporre ulteriormente la tabella senza perdere informazioni.