Appunti sulla Normalizzazione dei Database

Basato sulle slide del Prof. Danilo Montesi

17 maggio 2025

Indice

1 Normalizzazione nel Contesto dei Database

La **normalizzazione** è un processo fondamentale nella progettazione di database relazionali. Il suo scopo principale è organizzare i dati in modo da:

1. **Ridurre la ridondanza:** Evitare di ripetere le stesse informazioni in più punti.
2. **Eliminare le anomalie:** Prevenire problemi che possono sorgere durante l'inserimento, l'aggiornamento o la cancellazione dei dati.
3. **Garantire la qualità e l'integrità dei dati:** Assicurare che i dati siano coerenti e affidabili.

Le **Forme Normali (FN)** sono un insieme di regole che definiscono quanto "ben formata" è una tabella (relazione). Se una relazione non è in una forma normale adeguata, può presentare:

- **Ridondanze:** Dati duplicati inutilmente.
- **Comportamenti indesiderati durante gli aggiornamenti:** Ad esempio, la necessità di modificare lo stesso dato in più righe, con il rischio di dimenticarne qualcuna e creare inconsistenza.

La normalizzazione è una **tecnica di verifica** del design del database, non una metodologia di progettazione da zero. Prima progetti lo schema (magari con un modello E-R), poi lo verifichi e lo affini con la normalizzazione.

1.1 Esempio di Tabella con Anomalie

Consideriamo una tabella che traccia impiegati, progetti a cui lavorano, i loro stipendi, il budget dei progetti e il loro ruolo nel progetto:

Employee	Wage	Project	Budget	Role
Jones	20	Mars	2	Technician
Smith	35	Jupiter	15	Designer
Smith	35	Venus	15	Designer
Williams	55	Venus	15	Chief
Williams	55	Jupiter	15	Consultant
Williams	55	Mars	2	Consultant
Brown	48	Mars	2	Chief
Brown	48	Venus	15	Designer
White	48	Venus	15	Designer
White	48	Jupiter	15	Director

Questa tabella presenta diversi problemi (anomalie):

1. **Ridondanza:**
   - Lo stipendio (`Wage`) di un impiegato (es. Smith, 35) è ripetuto per ogni progetto a cui lavora.
   - Il budget (`Budget`) di un progetto (es. Jupiter, 15) è ripetuto per ogni impiegato che ci lavora.
2. **Anomalia di Aggiornamento (Update Anomaly):**
   - Se lo stipendio di Smith cambia, dobbiamo aggiornarlo in *tutte* le righe in cui Smith compare. Se ne dimentichiamo una, il database diventa inconsistente.
3. **Anomalia di Cancellazione (Deletion Anomaly):**
   - Se Jones smette di lavorare al progetto Mars (e Mars era il suo unico progetto), cancellando quella riga potremmo perdere l'informazione che Jones ha uno stipendio di 20 (se non ci sono altre tabelle che lo tracciano).
   - Similmente, se il progetto Mars viene cancellato e Jones e Brown lavoravano solo a Mars, perderemmo le informazioni su Jones e Brown.
4. **Anomalia di Inserimento (Insertion Anomaly):**

- Non possiamo inserire un nuovo impiegato con il suo stipendio se non è ancora assegnato a un progetto.
- Non possiamo inserire un nuovo progetto con il suo budget se nessun impiegato ci sta ancora lavorando.

1.2 Perché questa situazione è indesiderabile?

Perché stiamo mescolando diversi "concetti" o "pezzi di informazione" nella stessa tabella:

- Informazioni sugli impiegati e i loro stipendi.
- Informazioni sui progetti e i loro budget.
- Informazioni sul ruolo di un impiegato *all'interno di uno specifico progetto*.

2 Dipendenze Funzionali (Functional Dependencies - FD)

Per studiare e risolvere queste anomalie in modo sistematico, introduciamo il concetto di **Dipendenza Funzionale (DF)**. Una DF è un vincolo di integrità che descrive una relazione tra attributi all'interno di una tabella.

2.1 Definizione Formale

Data una relazione r con uno schema $R(X)$ (dove X è l'insieme di tutti gli attributi), e dati due sottoinsiemi non vuoti di attributi Y e Z (contenuti in X), esiste una dipendenza funzionale $Y \to Z$ (si legge "Y determina funzionalmente Z" o "Z dipende funzionalmente da Y") se e solo se: *Per ogni coppia di tuple (righe) t_1 e t_2 in r, se i valori degli attributi in Y sono uguali in t_1 e t_2 (cioè $t_1[Y] = t_2[Y]$), allora anche i valori degli attributi in Z devono essere uguali (cioè $t_1[Z] = t_2[Z]$).*

In parole povere: se conosci il valore di Y, puoi determinare *univocamente* il valore di Z.

2.2 Esempi dalla tabella precedente

- Employee $\to$ Wage
- Project $\to$ Budget
- $\{$Employee, Project$\}$ $\to$ Role

2.3 FD Triviali e Non Triviali

- Una FD $Y \to A$ è **triviale** se $A \subseteq Y$ (es. $\{$Employee, Project$\}$ $\to$ Project). Sono sempre vere e poco utili.
- Una FD $Y \to A$ è **non triviale** se $A \not\subseteq Y$. Sono queste che ci interessano per la normalizzazione.

2.4 Come le FD causano anomalie

Le anomalie sorgono principalmente quando abbiamo FD $X \to Y$ dove X **non è una superchiave** (o chiave candidata) della tabella.

- Employee $\to$ Wage: `Employee` da solo non è la chiave. Causa ridondanza.
- Project $\to$ Budget: `Project` da solo non è la chiave. Causa ridondanza.
- $\{$Employee, Project$\}$ $\to$ Role: $\{$Employee, Project$\}$ è (probabilmente) la chiave primaria. Questa FD **non causa anomalie**.

Le anomalie sono quindi causate dalla presenza di informazioni eterogenee.

3 Forma Normale di Boyce-Codd (BCNF)

La BCNF è una delle forme normali più stringenti e desiderabili.

3.1 Definizione

Una relazione r è in **BCNF** se, per ogni dipendenza funzionale non triviale $X \to Y$ definita su r:

- X è una **superchiave** di r.

Nella nostra tabella di esempio iniziale, non è in BCNF a causa di Employee $\to$ Wage e Project $\to$ Budget.

3.2 Cosa fare se una relazione non è in BCNF?

Si **decompone** la relazione in più relazioni più piccole, ognuna delle quali sia in BCNF.

3.3 Esempio di Decomposizione (per la tabella iniziale)

1. **ImpiegatiStipendi**(Employee, Wage)
2. **ProgettiBudget**(Project, Budget)
3. **ImpiegatiRuoliProgetto**(Employee, Project, Role)

Questa decomposizione elimina le anomalie.

3.4 Qualità della Decomposizione

Quando decomponiamo una tabella, dobbiamo assicurarci due proprietà fondamentali:

3.4.1 Lossless Join Property (Proprietà di Join Senza Perdita)

Dobbiamo essere in grado di ricreare la tabella originale facendo il JOIN delle tabelle decomposte. **Condizione:** Una decomposizione di $r(X)$ in $r_1(X_1)$ e $r_2(X_2)$ è senza perdita se l'intersezione degli attributi $X_0 = X_1 \cap X_2$ forma una chiave per almeno una delle relazioni decomposte ($X_0 \to X_1$ oppure $X_0 \to X_2$).

Esempio di Decomposizione CON PERDITA: Supponiamo R(Employee, Project, Office) con FD: Employee $\to$ Office e Project $\to$ Office. Decomposizione in:

- R_1(Employee, Office)
- R_2(Project, Office)

Attributo comune: `Office`. `Office` non è chiave né per R_1 né per R_2. Il join può generare tuple spurie.

Esempio di Decomposizione SENZA PERDITA: Tabella R(Employee, Project, Office) con FD: Employee $\to$ Office e chiave primaria {Employee, Project}. Decomposizione in:

- R_1(Employee, Office)
- R_2(Employee, Project)

Attributo comune: `Employee`. `Employee` è chiave per R_1. Lossless.

3.4.2 Dependency Preservation (Conservazione delle Dipendenze)

Tutte le dipendenze funzionali originali devono poter essere verificate esaminando una singola tabella nello schema decomposto. Nell'esempio di decomposizione R(E, P, O) con E $\to$ O e P $\to$ O (slide 28), se decomponiamo in R_1(E,O) e R_2(E,P), la FD P $\to$ O non è preservata.

4 Terza Forma Normale (3NF)

La 3NF è una forma normale leggermente meno stringente della BCNF che permette sempre una decomposizione lossless e che preserva le dipendenze.

4.1 Definizione

Una relazione r è in **3NF** se, per ogni dipendenza funzionale non triviale $X \to Y$ definita su r, almeno una delle seguenti condizioni è vera:

1. X è una **superchiave** di r (condizione BCNF). **OPPURE**
2. Ogni attributo in Y è parte di **almeno una chiave candidata** di r (cioè, ogni attributo in Y è un "attributo primo").

4.2 BCNF vs 3NF

- BCNF è più restrittiva. Ogni relazione in BCNF è anche in 3NF.
- Una relazione in 3NF potrebbe non essere in BCNF.
- Si può sempre decomporre una relazione in 3NF in modo lossless e preservando le dipendenze.
- Se una relazione ha una sola chiave candidata, allora 3NF e BCNF sono equivalenti.

4.3 Esempio 3NF (ma non BCNF)

Tabella R(Chief, Project, Office) FDs:

1. $\{$Project, Office$\} \to$ Chief (chiave candidata: $\{$Project, Office$\}$)
2. Chief $\to$ Office

Analizziamo Chief $\to$ Office:

- **BCNF:** Chief non è superchiave. **NON è in BCNF.**
- **3NF:** Chief non è superchiave. MA Office è parte della chiave candidata $\{$Project, Office$\}$. **È in 3NF.**

4.4 Algoritmo di Decomposizione in 3NF (Idea Generale)

1. Trova un insieme minimo di dipendenze funzionali (copertura canonica).
2. Per ogni FD $X \to Y$ in questa copertura, crea una tabella con attributi $X \cup Y$.
3. Se nessuna delle tabelle create contiene una chiave candidata della relazione originale, aggiungi un'ulteriore tabella contenente solo gli attributi di una chiave candidata originale.

4.5 Approccio Pratico Consigliato

1. Decomponi la relazione per raggiungere la 3NF.
2. Verifica se le tabelle risultanti sono anche in BCNF.
3. Se una tabella è in 3NF ma non in BCNF, valuta il trade-off.

4.6 Teoria delle Dipendenze e Implicazioni

Dato un insieme di dipendenze funzionali F, possiamo derivare altre dipendenze funzionali. Diciamo che F implica f se ogni relazione che soddisfa F soddisfa anche f. L'insieme di tutte le dipendenze implicite da F è chiamato **chiusura di F (F^+)**.

4.6.1 Assiomi di Armstrong

1. **Riflessività:** Se $Y \subseteq X$, allora $X \rightarrow Y$.
2. **Aumento:** Se $X \rightarrow Y$, allora $XZ \rightarrow YZ$.
3. **Transitività:** Se $X \rightarrow Y$ e $Y \rightarrow Z$, allora $X \rightarrow Z$.
    - Esempio: MatricolaStudente $\rightarrow$ CodiceCorsoLaurea e CodiceCorsoLaurea $\rightarrow$ NomeCorsoLaurea. Allora, MatricolaStudente $\rightarrow$ NomeCorsoLaurea.

4.6.2 Chiusura di un insieme di attributi X^+

L'insieme di tutti gli attributi che sono funzionalmente determinati da X, dato un insieme di FDs F.

4.6.3 Copertura Minima (o Canonica)

Un insieme "minimale" di FDs equivalente a F, senza ridondanze.

5 Normalizzazione nel Design Concettuale (Modello E-R)

La teoria della normalizzazione può essere usata anche per verificare la qualità di un modello Entità-Relazione.

5.1 Esempio: Normalizzazione su Entità

Considera un'entità `Prodotto` con attributi: `Codice` (PK), `NomeProdotto`, `Prezzo`, `PartitaIVAFornitore`, `NomeFornitore`, `IndirizzoFornitore`.

Identifichiamo una FD: PartitaIVAFornitore $\rightarrow$ NomeFornitore, IndirizzoFornitore. Qui, `PartitaIVAFornitore` non è la chiave di `Prodotto`. Questo viola le forme normali.

Decomposizione dell'Entità:

- Entità `Prodotto`(`Codice`, `NomeProdotto`, `Prezzo`)
- Entità `Fornitore`(`PartitaIVAFornitore`, `NomeFornitore`, `IndirizzoFornitore`)
- Relazione `Fornisce` tra `Fornitore` e `Prodotto`.

Nello schema proposto dalla slide 54:

- `Product(Code, Name, Price)`
- `Supplier(VATNum, Name, Address)`
- `Supply` (relationship)

5.2 Esempio: Normalizzazione su Relazioni (Relationship)

Considera una relazione `Tesi` che collega `Studente`, `Professore`, `DipartimentoProf`, `CorsoLaureaStudente`. Assumiamo che (MatricolaStudente, IDProfessore) sia la chiave. FDs:

- MatricolaStudente $\rightarrow$ CorsoLaureaStudente
- IDProfessore $\rightarrow$ DipartimentoProf

Nella tabella `Tesi`(MatricolaStudente, IDProfessore, CorsoLaureaStudente, DipartimentoProf):

- IDProfessore $\rightarrow$ DipartimentoProf: `IDProfessore` è parte della chiave, non la chiave intera. `DipartimentoProf` non è un attributo primo. Viola 3NF/BCNF.

Decomposizione della Relazione (Conceptual Level):

1. Creare un'entità `Professore` con attributo `Dipartimento` (slide 50: `Professor -(1,1)- Work -(0,N)- Dept`).

2. Creare un'entità `Studente` con attributo `CorsoLaurea` (slide 52: `Student -(1,1)- Enroll -(0,N)- Degree`).
3. La relazione `Tesi` ora collegherebbe solo `Studente` e `Professore` (slide 52: `Professor -(0,N)- Thesis -(0,1)- Student`).

Questo processo porta a un modello concettuale più robusto.