

Normalizzazione dei database

Uno dei concetti fondamentali della modellazione dei dati relazionali è la normalizzazione. Nei modelli di database relazionale un buon design di database si contraddistingue per una bassa ridondanza e la ragione è che i dati ridondanti portano ad anomalie semantiche. A propria volta queste anomalie appesantiscono l'elaborazione automatica dei dati e la gestione del database. La **normalizzazione è una strategia per eliminare le ridondanze nei database relazionali**. In questo articolo vi mostriamo come procedere.

Che cos'è la normalizzazione?

Con normalizzazione si intende un approccio al design di database che serve a evitare le ridondanze nei **database relazionali**.

La seguente tabella mostra i dati di fatturazione archiviati di un fornitore fittizio di una fabbrica: Geronimo Scatandro ha ordinato 10 monitor, 12 tappetini per il mouse e 1 sedia da ufficio per la propria attività, mentre l'ordine di Gioia Rinace consiste in 2 computer portatili e 2 cuffie.

Dati di fatturazione									
Nr. fatt.	Data	Cliente	Nr. cliente	Indirizzo	Nr. pos.	Articolo	Nr. art	Quantità	Prezzo
123	29.01.2018	Geronimo Scatandro	11	Piazza Roma 1, 00118 Roma	1	Monitor	2-0023-D	10	200 euro
123	29.01.2018	Geronimo Scatandro	11	Piazza Roma 1, 00118 Roma	2	Tappetino per il mouse	4-0023-D	12	50 centesimi
123	29.01.2018	Geronimo Scatandro	11	Piazza Roma 1, 00118 Roma	3	Sedia da ufficio	5-0023-D	1	120 euro
124	30.01.2018	Gioia Rinace	12	Via Roma 2, 00118 Roma	1	Computer portatile	1-0023-D	2	1200 euro
124	30.01.2018	Gioia Rinace	12	Via Roma 2, 00118 Roma	2	Cuffie	3-0023-D	2	75 euro



Nel database del negozio online vengono memorizzati i dati di fatturazione negli attributi *numero di fattura (nr. fatt.)*, *data*, *cliente*, *numero cliente (nr. cliente)*, *indirizzo*, *numero posizione fattura (nr. pos.)*, *articolo*, *numero articolo (nr. art.)*, *quantità* e *prezzo*.

La sezione di database mostrata sopra è il perfetto **esempio di un design di database non riuscito**: le ridondanze della tabella si notano già al primo sguardo. Inoltre le colonne *cliente* e *indirizzo* contengono dati che hanno più valori. In questi casi si parla di un database non normalizzato.

Svantaggi dei database non normalizzati:

- 1) Ridondanza e quindi anomalie possibili nelle operazioni di cancellazione, inserimento, modifica
- 2) La ridondanza può portare facilmente all'incongruenza
- 3) la necessità di un maggiore spazio di archiviazione, reso appunto necessario dai valori ridondanti.

- 4) Eventuali attributi multivalori (array) potrebbe non essere possibile attraverso query estrapolare i dati da valori multipli

Esempio: entrambi i clienti nella sezione di database mostrata hanno sede a Roma. Tuttavia, poiché queste informazioni non vengono considerate separatamente, le query avrebbero una notevole difficoltà nel filtrare ad esempio i clienti provenienti da uno stesso luogo.

Per evitare intervalli di valori doppi o multivalore, nell'ambito dei modelli di database relazionali sono state sviluppate tre forme normali che si susseguono progressivamente aumentando i requisiti.

La forma normale indica lo stato che si vuole raggiungere e ogni **forma normale** ha dei particolari requisiti che devono essere soddisfatti per raggiungere appunto questo stato. Esistono tre tipi di forme normali (prima, seconda e terza) e per ciascuna sono necessari requisiti diversi.

Fatto

Con normalizzazione si indica la trasposizione di una tabella di database in una forma normale di grado più alto. Se invece si scende di livello a una forma normale di grado più basso, si parla di denormalizzazione.

Normalizzazione: come si normalizza un database

Per illustrare il processo che porta un database relazionale nella prima, seconda e terza forma normale, diamo un'occhiata alle **singoli fasi della normalizzazione usando un esempio**. Il punto di partenza è la porzione di database presentata nell'esempio sopra.

Prima forma normale (1FN)

Una tabella di un database relazionale rientra nella prima forma normale (1FN) quando sono soddisfatti i seguenti requisiti:

1 tutti i dati hanno **valori atomici**;

2 tutte le colonne della tabella contengono **valori dello stesso tipo**.

Inoltre, il punto 1 impone anche che ogni attributo contenga un singolo valore (no valori multipli)

Di seguito trovate la nostra tabella sui dati di fatturazione nella quale abbiamo evidenziato in rosso i campi che o non sono atomici o che non contengono dati equivalenti.

Dati di fatturazione								
Data	Cliente	Nr. cliente	Indirizzo	Nr. pos.	Articolo	Nr. articolo	Quantità	Prezzo
29.01.2018	Geronimo Scatandro	11	Piazza Roma 1, 00188 Roma	1	Monitor	2-0023-D	10	200 euro
29.01.2018	Geronimo Scatandro	11	Piazza Roma 1, 00188 Roma	2	Tappetino per il mouse	4-0023-D	12	50 centesimi
29.01.2018	Geronimo Scatandro	11	Piazza Roma 1, 00188 Roma	3	Sedia da ufficio	5-0023-D	1	120 euro
30.01.2018	Gioia Rinace	12	Via Roma 2, 00188 Roma	1	Computer portatile	1-0023-D	2	1200 euro
30.01.2018	Gioia Rinace	12	Via Roma 2, 00188 Roma	2	Cuffie	3-0023-D	2	75 euro

Valori non atomici appesantiscono le query al database.

Le numerose ridondanze mostrano che la nostra tabella di partenza **viola entrambe le condizioni**, non rientrando nella prima forma normale.

Per normalizzare il database al livello della prima forma normale, è richiesta la seguente procedura:

1. separate i dati con più valori in colonne separate;
2. controllate che i valori di ogni colonna abbiano la stessa unità di misura o che siano dello stesso tipo.

Per portare i record di dati nella forma atomica, gli attributi *cliente* e *indirizzo* devono essere separati negli attributi specifici *cognome*, *nome*, *via*, *numero civico*, *codice di avviamento postale (CAP)* e *città*.

N.B.

Per stabilire da quando un valore può essere visto come atomico, occorre tenere conto del contesto di utilizzo. Se ad esempio non è necessaria una separazione tra nome e cognome, anche il nome completo di una persona può contare come valore atomico. Tuttavia nella pratica si consiglia di suddividere i valori composti da più elementi nelle unità più piccole possibili.

Nella colonna “prezzo” si trovano i valori espressi in euro e in centesimi. Occorre perciò decidere a quale tipo di unità di misura attenersi per avere intervalli di valore dello stesso tipo.

Portando alla prima forma normale tutti i dati con più valori vengono scomposti in dati atomici. I valori espressi in diverse unità di misura nella stessa colonna vengono trasformati in valori con le stesse unità di misura. Le celle dove sono stati apportati cambiamenti risultano evidenziate nell'immagine.

Nr. fatt.	Data	Nome	Cognome	Nr. cliente	Via	Nr. civico	CAP	Città	Nr. pos.	Articolo	Nr. Articolo	Quantità	Prezzo in €
123	29.01.2018	Geronimo	Scatandro	11	Piazza Roma	1	00188	Roma	1	Monitor	2-0023-D	10	200
123	29.01.2018	Geronimo	Scatandro	11	Piazza Roma	1	00188	Roma	2	Tappetino per il mouse	4-0023-D	12	0,50
123	29.01.2018	Geronimo	Scatandro	11	Piazza Roma	1	00188	Roma	3	Sedia da ufficio	5-0023-D	1	120
124	30.01.2018	Gioia	Rinace	12	Via Roma	2	00188	Roma	1	Computer portatile	1-0023-D	2	1200
124	30.01.2018	Gioia	Rinace	12	Via Roma	2	00188	Roma	2	Cuffie	3-0023-D	2	75

Il risultato è una tabella che corrisponde alla **prima forma normale**.

Seconda forma normale (2FN) → si applica solo quando ci sono chiavi composte

Una tabella conforme alla seconda forma normale deve soddisfare tutti i requisiti della prima forma normale e in aggiunta il seguente:

ogni attributo non chiave deve dipendere funzionalmente dall'intera chiave primaria e non solo da una parte della chiave.

Esempio di tabella che non rispetta la seconda forma normale (Chiave primaria composta da Numero fattura e numero posizione)

Nr. fatt.	Nr. pos.	Data	Nome	Cognome	Nr. cliente	Via	Nr. civico	CAP	Città	Articolo	Nr. articolo	Quantità	Prezzo in €
123	1	29.01.2018	Ger oni mo	Scatandro	11	Piazza Roma	1	00188	Roma	Monitor	2-0023-D	10	200
123	2	29.01.2018	Ger oni mo	Scatandro	11	Piazza Roma	1	00188	Roma	Tappetino per il mouse	4-0023-D	12	0,50
123	3	29.01.2018	Ger oni mo	Scatandro	11	Piazza Roma	1	00188	Roma	Sedia da ufficio	5-0023-D	1	120
124	1	30.01.2018	Gioi a	Rinace	12	Via Roma	2	00188	Roma	Computer portatile	1-0023-D	2	1200
124	2	30.01.2018	Gioi a	Rinace	12	Via Roma	2	00188	Roma	Cuffie	3-0023-D	2	75

La nostra tabella di esempio utilizza una chiave primaria composta, che consiste nel *numero di fattura* e nel *numero posizione*.

Per **portare una tabella di database nella seconda forma normale** occorre:

1. Accertatevi che tutti gli attributi non chiave siano pienamente dipendenti dalla chiave primaria.
2. Archivate tutti gli attributi non chiave che non sono completamente dipendenti dalla chiave primaria intera in tabelle separate.

Se osservate con attenzione la tabella di esempio, vi renderete conto che i **requisiti per la seconda forma normale non sono raggiunti**, per i seguenti motivi: la colonna *data* è dipendente soltanto da quella *numero di fattura (nr. fatt.)*, ma non da quella *numero posizione (nr. pos.)*. Lo stesso vale per i dati cliente *nome*, *cognome*, *via*, *numero civico*, *CAP* e *città*.

Per portare una tabella alla seconda forma normale archiviamo perciò in una tabella separata, che chiameremo “fattura”, tutti gli attributi che sono dipendenti soltanto dal numero di fattura.

Fattura								
<u>Nr. fatt.</u>	Data	Nome	Cognome	Nr. cliente	Via	Nr. civico	CAP	Città
123	29.01.2018	Geronimo	Scatandro	11	Piazza Roma	1	00188	Roma
124	30.01.2018	Gioia	Rinace	12	Via Roma	2	00188	Roma

Chiamiamo poi la tabella con i dati rimanenti “posizione fattura”.

Posizione fattura					
<u>Nr. fatt.</u>	<u>Nr. pos.</u>	Articolo	Nr. articolo	Quantità	Prezzo in €
123	1	Monitor	2-0023-D	10	200
123	2	Tappetino per il mouse	4-0023-D	12	0,50
123	3	Sedia da ufficio	5-0023-D	1	120
124	1	Computer portatile	1-0023-D	2	1200
124	2	Cuffie	3-0023-D	2	75

Dopo la normalizzazione il [numero di fattura](#) si trova in entrambe le tabelle e le collega l’una all’altra. Mentre l’attributo nella tabella “fattura” funge da chiave primaria, nella tabella “posizione fattura” svolge la funzione di **chiave esterna** e allo stesso tempo è parte della chiave primaria composta della tabella.

I dati di esempio soddisfano ora la seconda forma normale, ma non si può dire che tutte le ridondanze siano state eliminate: questo è proprio il fine della terza forma normale.

Terza forma normale (3FN)

Per portare una tabella alla terza forma normale occorre naturalmente che soddisfi innanzitutto i prerequisiti della prima e della seconda forma normale e in aggiunta il seguente:

Gli attributi non chiave non devono dipendere transitivamente dalla chiave.

Una **dipendenza transitiva** si ha quando un attributo non chiave è dipendente da un altro attributo non chiave e perciò indirettamente (appunto per la proprietà transitiva) dalle sue chiavi candidate.

Il nostro schema di database trasgredisce i requisiti della terza forma normale in più di un punto:

Fattura								
<u>Nr. fatt.</u>	Data	Nome	Cognome	Nr. cliente	Via	Nr. civico	CAP	Città
123	29.01.2018	Geronimo	Scatandro	11	Piazza Roma	1	00188	Roma
124	30.01.2018	Gioia	Rinace	12	Via Roma	2	00188	Roma

Tutte le colonne evidenziate in arancione nella tabella “Fattura” indicano dipendenze che sono incompatibili con le specifiche della terza forma normale.

Nella tabella “fattura” il nome e il cognome, come gli attributi *strada*, *numero civico*, *CAP* e *città*, sono dipendenti non soltanto dalla chiave primaria (il *numero fattura*), ma anche dal *numero cliente*.

Nella tabella “posizione fattura” gli attributi “articolo” e “prezzo” non dipendono dalla chiave primaria, data da *numero di fattura* e *numero posizione*, ma dal *numero dell’articolo*. Anche qui viene intaccato il requisito specifico della terza forma normale.

Posizione fattura					
<u>Nr. fatt.</u>	<u>Nr. pos.</u>	Articolo	Nr. articolo	Quantità	Prezzo in €
123	1	Monitor	2-0023-D	10	200
123	2	Tappetino per il mouse	4-0023-D	12	0,50
123	3	Sedia da ufficio	5-0023-D	1	120
124	1	Computer portatile	1-0023-D	2	1200
124	2	Cuffie	3-0023-D	2	75

Per eliminare tutte le **dipendenze transitive**, collochiamo i rispettivi attributi in tabelle separate, Ci sono così le quattro tabelle normalizzate “fattura”, “cliente”, “posizione fattura” e “articolo”.

Fattura		
<u>Nr. fatt.</u>	Data	Nr. cliente
123	29.01.2018	11
124	30.01.2018	12

Cliente						
<u>Nr. cliente</u>	Nome	Cognome	Via	Nr. civico	CAP	Città
11	Geronimo	Scatandro	Piazza Roma	1	00188	Roma
12	Gioia	Rinace	Via Roma	2	00188	Roma

Posizione fattura			
<u>Nr. fatt.</u>	<u>Nr. pos.</u>	Nr. articolo	Quantità
123	1	2-0023-D	10
123	2	4-0023-D	12
123	3	5-0023-D	1
124	1	1-0023-D	2
124	2	3-0023-D	2

Articoli		
<u>Nr. articolo</u>	Articolo	Prezzo in €
1-0023-D	Computer portatile	1200
2-0023-D	Monitor	200
3-0023-D	Cuffie	75
4-0023-D	Tappetino per il mouse	0,50
5-0023-D	Sedia da ufficio	120

La terza forma normale è ritenuta generalmente lo standard per i database relazionali, da cui ci si discosta solo in casi eccezionali. Per esempio può capitare che i database che si trovano nella terza forma normale vengano denormalizzati nella seconda forma normale. Questo accade perché utilizzare join su più tabelle può comportare un grande dispendio di tempo, mentre **attraverso la denormalizzazione si diminuisce il numero di tabelle e perciò si abbrevia il tempo della query.**

Altre forme normali

Di solito la normalizzazione termina con la terza forma normale. Le seguenti forme normali si riferiscono a database con condizioni speciali e sono pertanto utilizzate solo in casi eccezionali.

Vantaggi e svantaggi della normalizzazione

Il fine della normalizzazione è la riduzione dei valori doppi. Se portate il vostro database a una delle forme normali, avrete il vantaggio che lo schema raggiunto presenterà **meno ridondanze** rispetto a quello di partenza (e quindi meno incongruenze ed anomalie).

Lo svantaggio principale è il fatto che le query sono più complesse e richiedono più tempo di esecuzione in un database normalizzato, visto che la normalizzazione fa crescere il numero di tabelle.