LEZIONE 27

La normalizzazione

Se lo schema della base di dati non è costruito correttamente, può accadere che si abbiano delle **anomalie** nel database, come, ad esempio, quelle derivanti dalla ripetizione dei dati, che portano a spreco di tempo (per l'inserimento e l'aggiornamento dei dati) e di spazio (memoria). Ciò può compromettere la congruenza dei dati contenuti nella base di dati durante le operazioni di inserimento, aggiornamento e modifica.
Facciamo un esempio. Consideriamo la seguente relazione *Atipica*.

CodCliente	Città	CodArticolo	Descrizione	NumPezzi
Rossi	Roma	Art1	Matita colorata	10
Rossi	Roma	Art2	Penna biro	20
Rossi	Roma	Art7	Penna stilografica	54
Neri	Lecce	Art5	Gomma	62
Verdi	Milano	Art1	Matita colorata	1
Verdi	Milano	Art2	Penna biro	25

Valutiamo, ora, le anomalie.

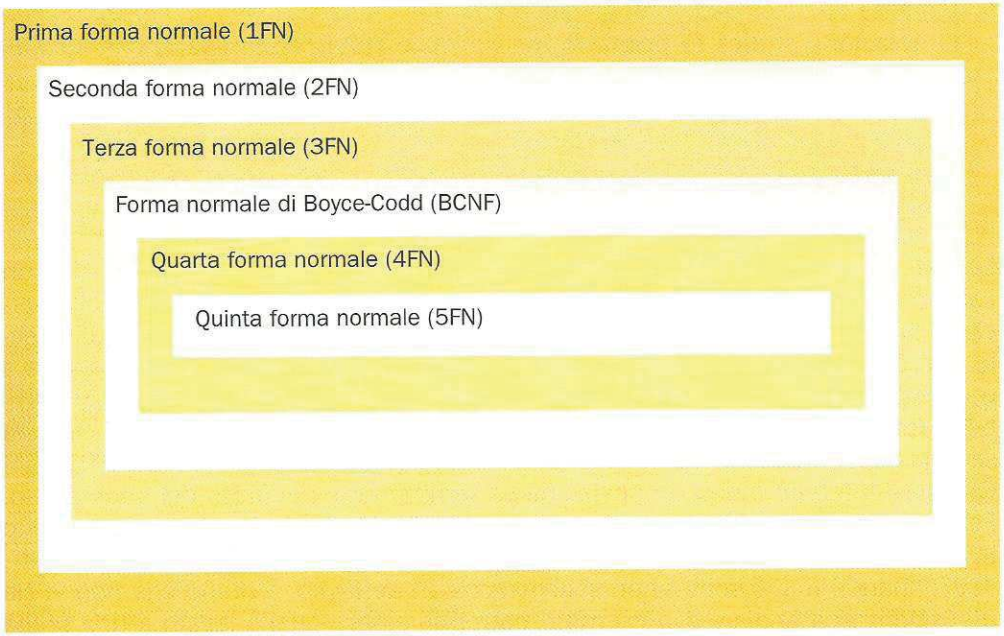
- **Anomalia in inserimento**: per inserire un nuovo cliente è necessario inserire contestualmente un articolo ordinato. Allo stesso modo, non è possibile inserire un nuovo articolo senza specificare un acquirente (ciò perché la chiave primaria della relazione è *CodCliente*, *CodArticolo* e non può mai essere nulla);
- **Anomalia in cancellazione**: se si cancella una t-upla relativa a un acquisto, si corre il rischio di cancellare anche dati relativi al cliente. Ad esempio, cancellando la t-upla (*Rossi, Art1*) non si cancella l'indirizzo del cliente che continua a rimanere grazie alle t-uple (*Rossi, Art2*) e (*Rossi, Art7*). Se, invece, si cancella la t-upla (*Neri, Art5*) si perdono irrimediabilmente anche i dati del cliente. Analoghe considerazioni possono essere fatte per l'articolo.
- **Anomalia in aggiornamento**: se occorre variare l'indirizzo di un cliente, occorrerà aggiornare ogni t-upla in cui compare quel cliente.

Queste anomalie, in particolare quelle in cancellazione e in aggiornamento, sono una diretta conseguenza della **ridondanza**, ossia della presenza di dati ripetuti inutilmente, cioè senza l'apporto di nuova informazione.
Lo scopo della **teoria della normalizzazione** è quello di fornire un metodo per progettare basi di dati senza anomalie, ossia per creare tabelle corrette.

La **normalizzazione** è un procedimento di tipo graduale, che realizza un'ottimizzazione progressiva a partire da relazioni non normalizzate fino a raggiungere un certo livello di normalizzazione. In particolare, consente di verificare se la definizione dello schema corrisponde ai "canoni standard" di correttezza della base di dati e, in caso, avvalendosi di un preciso insieme di regole, di riportare le tabelle in quelle che sono definite le **forme normali** delle tabelle relazionali.
Una **forma normale** è una proprietà di uno schema relazionale che ne garantisce la "qualità", cioè l'assenza di determinati difetti.

Nella teoria delle basi di dati relazionali esistono diverse forme normali (prima, seconda, terza, forma normale di Boyce-Codd, quarta e quinta). Nella nostra trattazione giungeremo sino alla forma normale di Boyce-Codd, lasciando la trattazione della quarta e della quinta a studi superiori.

La normalizzazione va utilizzata come tecnica di verifica dei risultati della progettazione di una base di dati. Non costituisce, quindi, una metodologia di progettazione.



È possibile riassumere la teoria della normalizzazione nelle seguenti regole:

- ogni tabella deve avere una chiave primaria;
- ogni campo deve contenere un solo valore;
- i campi di una tabella non devono dipendere da altri campi che non siano la chiave primaria;
- bisogna evitare le ripetizioni e la ridondanza dei dati.

In fase di progettazione occorre sempre verificare che ogni tabella abbia una chiave primaria (composta da un campo o da un insieme di campi). Facciamo un esempio. Consideriamo la seguente tabella *OrdiniAcquisto*:

Data	Cliente	Quantità	Articolo	Prezzo	PrezzoTotale	TipoPagamento
15/12/04	Rossi	25	A023	€ 50,25	€ 1.256,25	Contanti
15/12/04	Neri	120	A789	€ 25,80	€ 3.096,00	Contanti
15/12/04	Rossi	100	A976	€ 22,14	€ 2.214,00	Bonifico
15/12/04	Rossi	45	G324	€ 50,25	€ 2.261,25	Assegno
29/12/04	Verdi	120	A023	€ 25,80	€ 3.096,00	Contanti

La tabella contiene chiavi candidate molto lunghe. Decidiamo pertanto di aggiungere un nuovo campo, ad esempio il codice dell'ordine, che sia una chiave minimale.

CodOrdine	Data	Cliente	Quantità	Articolo	Prezzo	PrezzoTotale	TipoPagamento
1	15/12/04	Rossi	25	A023	€ 50,25	€ 1.256,25	Contanti
2	15/12/04	Neri	120	A789	€ 25,80	€ 3.096,00	Contanti
3	15/12/04	Rossi	100	A976	€ 22,14	€ 2.214,00	Bonifico
4	15/12/04	Rossi	45	G324	€ 50,25	€ 2.261,25	Assegno
5	29/12/04	Verdi	120	A023	€ 25,80	€ 3.096,00	Contanti

LEZIONE 28

La prima forma normale (1FN)

Partiamo dalla definizione:

Una relazione si dice in prima forma normale (1FN), chiamata anche forma atomica, se:

- esiste una chiave primaria (un insieme di attributi che identifica in modo univoco ogni t-upla della relazione);
- ogni attributo è definito su un dominio di valori atomici (deve essere cioè un campo semplice, quindi non composto e non multiplo).

La 1NF è di norma implicita: ogni informazione deve essere "atomica", cioè un campo deve contenere una e una sola informazione. Diciamo che è implicita perché i campi contengono sempre un tipo di dato "semplice" (stringa, intero, reale...) e mai aggregazioni (vettori, record...). Le date potrebbero contravvenire a questo principio, perché in una data ci sono il giorno, il mese e l'anno che potrebbero essere visti come entità separate e separabili, ma di solito si tralascia questa scomposizione considerando una "data" come un tutt'uno.
Qualsiasi intervento di scomposizione va sempre fatto nei limiti del buon senso. Per esempio, un indirizzo "Via Santa Maria 80 Bari", dovrebbe essere scomposto in parti come *Topotassia*, *Toponimia*, *Civico*, *Città* vale a dire ("Via", "Santa Maria", "80", "Bari"), ma di solito, non si arriva così in dettaglio, a meno che non vi sia un'esigenza particolare. Un esempio potrebbe essere quello legato a una realtà che si occupa delle spedizioni postali divise per città; in questo caso l'indirizzo andrebbe disaggregato in maniera profonda. In tutti gli altri casi è sufficiente avere *Indirizzo* e *Città*.
In generale, quindi, quando si incontrano campi che contengono più valori, questi campi devono essere suddivisi in modo che contengano un unico valore su ogni record. In alcuni casi la suddivisione è semplice e naturale, in altri è molto più complessa. Come al solito, un esempio vale molto più di tante parole e, a tal proposito, riprendiamo proprio il caso di un indirizzo. Analizziamo la seguente relazione *Persone*.

CodPersona	Cognome	Nome	Indirizzo
1	Neri	Antonio	Viale Unità d'Italia, 34 Lecce
2	Verde	Marco	Via Piave, 6/B Ancona
3	Bianco	Piero	Piazza Dante, 9 Milano

La tabella ha una chiave primaria, quindi la prima regola è confermata. Il campo *Indirizzo*, però, contiene più valori: la via con il numero civico e la città. Per normalizzare la relazione si suddivide il campo *Indirizzo* in modo che ogni informazione sia rappresentata con un apposito attributo.

CodPersona	Cognome	Nome	Indirizzo	Città
1	Neri	Antonio	Viale Unità d'Italia, 34	Lecce
2	Verde	Marco	Via Piave, 6/B	Ancona
3	Bianco	Piero	Piazza Dante, 9	Milano

Osserviamo che *Indirizzo*, ora, contiene più informazioni, la via e il numero civico, ma risulta comunque di tipo stringa, che è un tipo semplice. Se queste informazioni devono essere utilizzate solo per la memorizzazione, la scrittura e la let-

tura, possono essere lasciate in questa forma, ma se ad esempio si volessero effettuare delle ricerche in base alla via, l'algoritmo di ricerca potrebbe risultare più complicato e certamente più lento, perché occorrerebbe estrarre dal campo le parti di testo che corrispondono al nome della via. In questi casi può essere utile scomporre ulteriormente l'attributo.

CodPersona	Cognome	Nome	Indirizzo	NomeVia	Numero	Città
1	Neri	Antonio	Viale	Unità d'Italia	34	Lecce
2	Verde	Marco	Via	Piave	6/B	Ancona
3	Bianco	Piero	Piazza	Dante	9	Milano

Per sapere fino a dove suddividere il campo si deve consultare l'analisi dei requisiti. In generale, tutti i dati che servono per confronti o ricerche, nel normale utilizzo del database, devono essere isolati.

Un altro caso di relazione non normalizzata è quello in cui sono presenti attributi multipli. Ricordiamo che un attributo è multiplo se può essere una sequenza di valori tutti dello stesso tipo. Ad esempio, per una persona l'attributo *NumeriDiTelefono*, può essere multiplo poiché una persona può avere più recapiti telefonici.

CodPersona	Cognome	Nome	NumeriDiTelefono		
1	Neri	Antonio	02/324328525	02/89346874	348/4324324
2	Bianco	Marco	06/90596590	335/94054946	
3	Testo	Piero	06/90596590	380/39676654	

In un certo senso è come se a 1 persona corrispondessero N numeri di telefono. Per avere uno schema in 1FN si sostituisce la relazione non normalizzata con due relazioni: una simile a quella di origine, ma senza l'attributo multiplo, l'altra contenente la chiave primaria della prima relazione e un attributo semplice che contiene ogni singolo valore della sequenza originale. La chiave primaria di quest'ultima relazione è data dall'unione della chiave primaria della relazione iniziale e dell'attributo.
Nel nostro esempio avremmo:

CodPersona	Cognome	Nome
1	Neri	Antonio
2	Verde	Marco
3	Bianco	Piero

CodPersona	NumeroDiTelefono
1	02/32432825
1	02/89346874
1	348/4324324
2	06/90596590
2	335/94054946
3	06/90596590
3	380/39676654