Le regole di normalizzazione

Le regole di normalizzazione sono l'esplicitazione teorica di alcuni *problemi* che possono emergere durante l'utilizzo, l'interrogazione e la gestione dei dati in un DataBase e che possono impedire o rendere complicato l'uso delle informazioni archiviate in esso:

- *Ridondanza informativa*
- *Anomalie con rischio di incoerenza*

Non è sempre possibile applicare le regole di normalizzazione: in taluni casi potrebbe comportare la perdita di informazioni. In tali casi si dovranno risolvere i problemi provocati dalla mancata normalizzazione che possono presentarsi, oppure è possibile che il DBMS che si utilizza sia in grado di fornirci strumenti che gestiscono le situazioni reali senza bisogno di applicare le regole di normalizzazione.

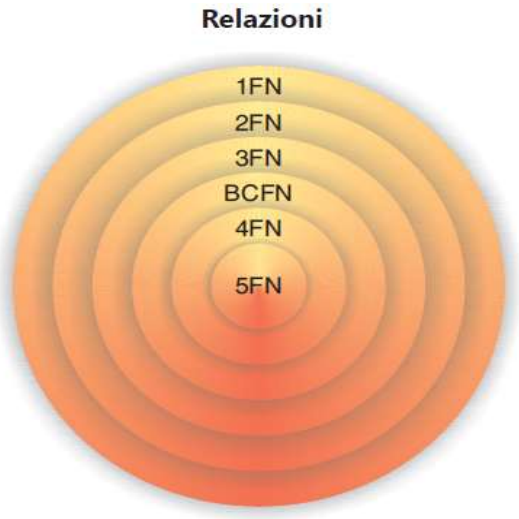
Le regole o forme normali sono numerose anche se in genere si ritiene che l'applicazione delle prime tre forme normali permetta di raggiungere un adeguato grado di *efficienza* del DataBase evitando che una tabella presenti anomalie.

Le regole prendono i seguenti nomi:

- **prima forma normale** (violazione: presenza di dati aggregati);
- **seconda forma normale** (violazione: dipendenza parziale dalla chiave primaria);
- **terza forma normale** (violazione: dipendenza indiretta dalla chiave primaria).

Oltre a queste prime tre forme normali, ne sono state formalizzate altre come: forma normale Boyce/Codd; quarta e quinta forma normale; forma normale Strong Join-Protection; forma normale Over-Strong Join-Protection; forma normale Domain –Key.

Le forme normali sono di solito definite sul modello relazionale, ma hanno senso anche in altri contesti, ad esempio nel modello E/R



Le regole di normalizzazione aiutano a comprendere come la soluzione di una relazione *molti a molti* nasca dall'esigenza di evitare le **difficoltà** che proprio tali regole mettono in evidenza:

- nel caso della **prima forma normale** una molteplicità di attributi contenenti il medesimo tipo di informazioni con grandi difficoltà nell'interrogazione e nella adattabilità futura del database;

- nel caso della **seconda e terza forma normale** una molteplicità di ripetizioni di informazioni con problemi di errori di digitazione, occupazione dello spazio di memoria e di interrogazione dei dati.

FORME NORMALI

- Una forma normale è una *proprietà di uno schema* relazionale che ne garantisce la "**qualità**", cioè l'**assenza di determinati difetti**
- Una **relazione non normalizzata**:
  - **presenta ridondanze**,
  - si presta a **comportamenti poco desiderabili durante gli aggiornamenti**
- L'attività che permette di trasformare schemi non normalizzati in schemi che soddisfano una forma normale è detta **normalizzazione**.
- La normalizzazione va utilizzata come **tecnica di verifica dei risultati della progettazione di una base di dati**
- Non costituisce quindi una metodologia di progettazione

Secondo la definizione (nel contesto del modello E/R) della **Prima forma normale**:

*Un'entità deve contenere attributi che contengono una sola informazione e che siano **"atomici"***

L'informazione già contenuta in un attributo non deve essere contenuta all'interno dell'entità in altra forma o in un attributo con nome diverso. **Applicare la prima forma normale** significa eliminare tutti gli attributi sovrabbondanti --> tutti i campi sono **semplici** cioè **non** ci sono **campi multipli**, ripetitivi**.**

Per risolvere i problemi dovuti alla ridondanza dei dati
viene in aiuto la **Seconda forma normale** la cui definizione è:

Un'entità risponde alla seconda forma normale quando
tutti i suoi attributi dipendono dalla chiave primaria dell'entità

Per poter applicare la seconda forma normale si deve aver già applicato la prima forma normale

Si deve **evitare** una **dipendenza parziale dalla chiave primaria**. **Applicare la seconda forma normale** significa eliminare i campi che non dipendono da tutta la chiave.

Dopo aver applicato la prima e la seconda si può passare all'applicazione della **Terza forma normale** la quale stabilisce che: **non esistono dipendenze funzionali** cioè

la dipendenza tra gli attributi deve essere basata sulla chiave primaria e non su altri attributi.

Applicare la terza forma normale significa **evitare** una **dipendenza indiretta alla chiave primaria** cioè eliminare i campi che dipendono da quelli non identificatori (*dipendenza transitiva*).

Glossario

Dipendenza funzionale (FD): un particolare vincolo di integrità che esprime legami funzionali tra gli attributi; una proprietà semantica (cioè che dipende dai fatti) per formalizzare la nozione di schema senza anomalie.

Le FD non si "ricavano" dall'analisi dei dati, ma ragionando sugli attributi dello schema: diciamo che in un'istanza (r) di uno schema vale una *dipendenza funzionale* se un sottoinsieme di attributi Y non nullo (il valore di un attributo) **implica** (o *determina funzionalmente*) un altro sottoinsieme di attributi Z non nullo (Y →Z).

Esempio considerando uno schema logico:

DotazioniLibri (IDLibro, NomeNegozio, IndNegozio, Titolo, Quantità)

FD:

IDLibro → Titolo
NomeNegozio → IndNegozio
IDLibro, NomeNegozio → IndNegozio, Titolo, Quantità

Mediateca e ***normalizzazione***

Schema di tabella *grassa* (in inglese: *fat table*)

Media (***TbMedia***) potrebbe essere strutturato così:

Campo	Descrizione
Id	Chiave primaria – Intero ad incremento automatico
Desc	Descrizione
TipoSupp	Tipo del supporto di memorizzazione (CD, DVD, Rivista, File, etc.)
Argom	Argomento o classificazione
Ubicazione	Ubicazione (Scaffale, Scatola, Num Rivista, Percorso su Hd etc.)
Prezzo	Prezzo (di acquisto, di vendita, di prestito...), forse non necessario, ma sicuramente didattico
Note	Annotazioni libere
Prestato	Si / No

Siccome è bene, prima di progettare qualunque Database, preparare uno schema *reale* di quello che la Tabella dovrà contenere, ecco come potrebbe essere una parte del nostro archivio:

Desc	TipoSupp	Argom
Pearl Jam - Binaural	CD Audio	Rock
Fromm – Avere o Essere ?	Libro	Filosofia
Harry Potter e la camera dei segreti	DVD Film	Fantasy
Peter Gabriel – Growing Up Live	DVD Musicale	Rock
Condividere risorse con Samba 3	Rivista	Linux / Samba
Rossini – 10 Ouvertures – Chailly	CD Audio	Classica

TbMedia (Id, Desc, TipoSupp, Argom, Ubicazione, Prezzo, Note, Prestato)

- È in **1NF** poiché tutti gli attributi sono semplici o **atomici** ed **esiste** una ***chiave primaria*** (ossia esiste un insieme di attributi, che identifica in modo univoco ogni tupla della relazione)
- È in **2NF** : la dipendenza non è parziale essendo stata scelta una chiave unica (*codice univoco Id*) ma tale scelta inserisce *transitività* (TipoSupp o Argom non dipendono direttamente da un codice che individua luogo e prezzo ….) pertanto lo schema **non è in 3NF** e, come esemplificato, non si evitano ripetizioni con problemi di errori di digitazione e occupazione dello spazio di memoria.

Normalizzazione: spezzare la ***tabella grassa*** in più tabelle *magre*.

In questo caso occorrerebbero più tabelle:

TbMedia (MId, MDesc, MUbic, MPrezzo, ….) con chiavi esterne *SuppId* e *ArgId*

TbSupporti (SuppId, SuppDesc)

TbArgomenti (ArgId, ArgDesc)

Progettando poi una archiviazione più ricca delle informazioni relative ai prestiti

Si distinguono tre tipi di anomalia, uno per ogni tipo di transazione.

1. *Anomalia di inserimento*. Se nell'inserire un nuovo record in una tabella si è costretti a inserire informazioni già presenti nel DB.

2. *Anomalia di cancellazione*. Se nel cancellare un record si è costretti a cancellare informazioni che possono essere ancora utili nel DB.

3. *Anomalia di aggiornamento*. Se dovendo aggiornare un record si è costretti ad aggiornarne molti altri.

Un errore di progettazione molto comune che dà quasi sempre luogo ad anomalie è quello di voler realizzare il Data-Base con un'unica grande tabella che contenga tutte le informazioni possibili (in gergo detta ***tabella grassa*** - in inglese: *fat table*).

Tabella **TbMedia -> Archivio Principale**

Campo	Tipo di Campo	Commenti	Idx
MId	Integer auto_increment	Identificatore - Chiave primaria	Pk
MDes	Varchar(100)	Descrizione max 100 caratteri	*
SuppId	Integer	Identificatore del Supporto	*
ArgId	Integer	Identificatore dell'Argomento	*
MUbic	Varchar(50)	Ubicazione max 50 caratteri	
MPrezzo	Decimal(14,2)	Prezzo in Euro (max due cifre dec)	

Tabella **TbSupporti -> Tipologie dei Supporti**

Campo	Tipo di Campo	Commenti	Idx
SuppId	Integer auto_increment	Identificatore - Chiave primaria	Pk
SuppDes	Varchar(30)	Descrizione max 30 caratteri	*

Tabella **TbArgomenti -> Tipologie degli Argomenti**

Campo	Tipo di Campo	Commenti	Idx
ArgId	Integer auto_increment	Identificatore - Chiave primaria	Pk
ArgDes	Varchar(30)	Descrizione max 30 caratteri	*

Tabella **TbUtenti -> Archivio degli Utenti dei Prestiti**

Campo	Tipo di Campo	Commenti	Idx
UtId	Integer auto_increment	Identificatore - Chiave primaria	Pk
UtDen	Varchar(50)	Denominazione max 50 caratteri	*
UtVia	Varchar(50)	Indirizzo max 50 caratteri	
UtCit	Varchar(100)	Città max 100 Caratteri	
UtTel	Varchar(20)	Telefono max 20 Caratteri	
UtDNas	Data	Data di Nascita	
UtImg	Immagine	Foto dell'Utente	
UtCodFis	Varchar(16)	Codice Fiscale	
UtSesso	Char(1)	Sesso (M o F)	
UtTessera	Integer(1)	Tesserato (valore Logico, 0 o 1)	

Tabella **TbPrestiti -> Archivio dei prestiti e delle restituzioni**

Campo	Tipo di Campo	Commenti	Idx
PreId	Integer auto_increment	Identificatore - Chiave primaria	Pk
PreData	Date	Data del prestito	*
UtId	Integer	Identificativo Utente (TbUtenti)	*
MId	Integer	Identificativo Media (TbMedia)	*
PreDataR	Date	Data di restituzione	

Nelle figure **Pk** individua la **chiave primaria** ed una ***** evidenzia gli altri campi ***indicizzati***. Tra i campi ***indicizzati*** le *chiavi esterne* e quelli previsti di frequente consultazione.

Normalizzare o no?

- La normalizzazione non va intesa come un obbligo, in quanto in alcune situazioni le anomalie che si riscontrano in schemi non normalizzati sono un male minore rispetto alla situazione che si viene a creare normalizzando
- In particolare, le cose da considerare sono:
  - Normalizzare elimina le anomalie, ma può appesantire l'esecuzione di certe operazioni (join tra gli schemi normalizzati)
  - La frequenza con cui i dati vengono modificati incide su qual è la scelta più opportuna (relazioni "quasi statiche" danno meno problemi se non normalizzate)
  - La ridondanza presente in relazioni non normalizzate va quantificata, per capire quanto incida sull'occupazione di memoria, e sui costi da pagare quando le repliche di una stessa informazione devono essere aggiornate

In generale, se un numero significativo di query richiede il join di più di cinque o sei tabelle, è consigliabile considerare la denormalizzazione.

Riassumiamo:

- La definizione delle forme normali (3NF e successive) si basa sul rispetto del vincolo di dipendenza funzionale (FD)
- Normalizzare uno schema significa decomporlo in sottoschemi
- Ogni decomposizione deve essere senza perdita, ovvero deve permettere di ricostruire esattamente la relazione originaria non decomposta
- È anche opportuno che la decomposizione preservi le FD, al fine di evitare (o ridurre la complessità di) query di verifica che garantiscano che i vincoli siano rispettati

Il processo di normalizzazione si fonda su un semplice criterio: se una relazione presenta più concetti tra loro indipendenti, la si decompone in relazioni più piccole, una per ogni concetto.

Forma normale	Verifica	Rimedio (Normalizzazione)
Prima (1NF)	Le relazioni non devono contenere attributi multivalore o relazioni annidate	Creare nuove relazioni per ciascun attributo multivalore o per ciascuna relazione annidata
Seconda (2NF)	Nel caso di relazioni in cui la chiave primaria contiene più attributi, nessun attributo non-primo deve essere funzionalmente dipendente da una parte della chiave primaria	Decomporre creando una nuova relazione per ciascuna chiave parziale con i relativi attributi dipendenti. Deve rimanere anche una relazione che contiene la chiave primaria originaria e tutti gli attributi che sono dipendenti in modo completo dalla chiave
Terza (3NF)	La relazione non deve contenere attributi non-primi che siano funzionalmente dipendenti da altri attributi non-primi.	Decomporre creando una relazione che contenga gli attributi non-primi che determinano funzionalmente altri attributi non primi