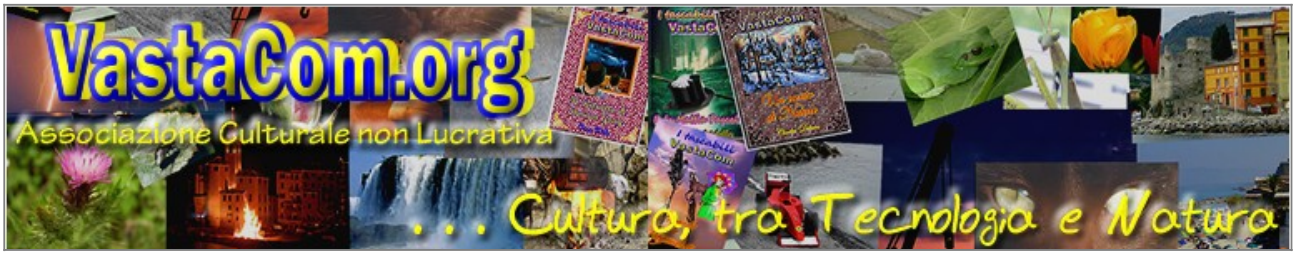

Corso Database Relazionali

Versione 1.2 del 01.10.2011

Autore: Stefano Lampredi

Prima edizione: 2011
Edizione elettronica VastaCom: 2011

VastaCom è un'associazione culturale non lucrativa con l'obiettivo di diffondere informazione e servizi di interesse sociale e culturale tramite canali telematici e in particolare il web **(www.vastacom.org).**

Indice generale

Lezione 1.1 - Introduzione - Corso Database Relazionali

Di questi tempi fare un corso sui database relazionali è un po come parlare del sesso degli angeli. Se ne è scritto e parlato in tutte le salse e probabilmente non si dirà di più (ma forse qualcosa di meno) di tanti libri, articoli, siti internet. Le motivazioni sono quindi prevalentemente personali. La modalità di espressione e l'approccio all'argomento è sicuramente all'acqua di rose. Quindi non me ne vogliate e non sparate troppo sulla croce rossa... che in questo caso sarei io.

Va bene ho messo le mani avanti possiamo cominciare.

Volendo iniziare il corso con una definizione un po' teorica possiamo considerare il Database come quel software che permette l'inserimento, la manipolazione e l'estrazione dei dati su un supporto informatico.

Il database oggi è un software che troviamo in una miriade di applicazioni informatiche e multimediali.
Possiamo dire che siamo circondati da database.

Se pensiamo all'oggetto che l'uomo di oggi porta sempre con se, il cellulare, la prima cosa che mi viene in mente è che la rubrica del cellulare è un database. Piccolo ma pur sempre un database.

La rubrica infatti ci permette di inserire nomi, cognomi, numeri di telefono (alcune volte anche altre informazioni).
La rubrica ci permette di manipolare queste informazioni cioè di modificarle.
La rubrica ci permette di estrarre queste informazioni e di utilizzarle per indirizzare una telefonata.

Quindi: 1) inserimento 2) manipolazione 3) estrazione.

Lezione 1.2 - Cos'è un database - 1^ parte

Ragionando sull'esempio della rubrica telefonica presentatovi nell'introduzione ecco

però sorgere qualche interrogativo.

Ma cosa è la rubrica? L'applicazione "Rubrica del cellulare" è un database?
Direi di no! O meglio non precisamente.

Infatti l'applicazione (o software) della Rubrica può essere vista su strati differenti.

Il **1° strato** rappresenta la tabella della rubrica composta da tante colonne quanti sono i dettagli che vogliamo memorizzare in rubrica (nome, cognome, cell, tel. fisso, cell lavoro....) e tante righe quante sono le persone registrate.

Il **2° strato** è l'interfaccia e la logica di gestione dell'inserimento, della visualizzazione e della modifica dei dati della rubrica. Quindi il software con il quale si inseriscono si vedono e si modificano i nomi, i numeri in rubrica.

Per dirla semplice da un lato abbiamo il contenitore delle informazioni e dall'altro abbiamo il motore di gestione del contenitore stesso.

A questo punto possiamo dire che la rubrica sia un software che permette la gestione delle informazioni in un database.

Questo database, nel nostro esempio, è formato da una tabella che chiameremo "tab_rubrica".

Cerchiamo quindi di esprimere un po meglio la prima definizione e spacchiamola in due:

Data Base "Rubrica": contenitore di informazioni.

Applicazione "Rubrica": software che permette l'inserimento, la manipolazione e l'estrazione dei dati contenuti nel Data Base "Rubrica".

Lezione 1.2 - Cos'è un database - 2^ parte

Come abbiamo visto nella 1^ parte abbiamo suddiviso l'Applicazione Rubrica in due strati.

Nel primo abbiamo solo il contenitore delle informazioni ed è quindi rappresentato da oggetti come tabelle (nel nostro caso la tabella tab_rubrica).
Nel secondo abbiamo raggruppato le parti applicative e funzionali del software Rubrica.

Io però voglio spaccare il pelo in tre.
Il mio obiettivo è suddividere in parti ben definite il software Rubrica.
Mi fermerò comunque alle tre parti principali e non vi tedierò ulteriormente.

Infatti il secondo strato può essere suddiviso ulteriormente in due parti ben definite:

1) Il **motore del database**. Questo rappresenta tutti quei metodi pubblici che possono essere utilizzati per inserire, modificare e interrogare il dato immaganizzato nel contenitore.

2) il **software che usa questo motore**. Questa è la parte che usa questi metodi per permettere ad un utente di inserire, modificare e visualizzare il dato immagazzinato nel contenitore.

Per fare un esempio pratico un utente medio del cellulare quando inserisce un nome in rubbrica usa una interfaccia grafica di inserimento (secondo strato).
L'interfaccia grafica sotto di se traduce quello che è inserito nella casella della rubrica in un linguaggio particolare (SQL che poi vedremo), lo trasferisce al motore (primo strato).
Il motore interpreta questo linguaggio e inserisce il dato nella tabella.

Quindi riassumendo gli strati partendo dal livello più vicino all'utente:

1) **Maschera Utente**: che ha in se una interfaccia grafica e la logica di trasformazione delle azioni impartite dall'utente in un linguaggio comprensibile al motore del database (SQL).
2) **Motore del database** (DBMS). Inserisce, modifica, interroga il database in base ai comandi che gli vengono impartiti dalla Maschera Utente.
3) **Data Base** o contenitore dei dati che accoglie le informazione passategli dai livelli superiori e le organizza in tabelle (DB).

Vediamo ora gli acronimi:

SQL = Structured Query Lamguage ossia linguaggio strutturato per fare interrogazioni e modifiche al database.
DB = Data Base ossia Base di Dati.
DBMS = Data Base Management System ossia Sistema per gestire un Data Base.

Lezione 1.2 - Cos'è un database - 3^ parte

Con il ragionamento fatto in questo paragrafo ho voluto semplicemente farvi entrare dentro la definizione del Database.

Ho cercato di portare allo scoperto le sue caratteristiche che possono ai nostri occhi apparire nascoste vedendo un software molto semplice come la nostra rubrica dell'esempio.

Con questo voglio anche dire che le definizioni del database che si possono trovare in giro o che si possono ascoltare nel linguaggio "tecnichese" spesso tendono ad incorporare nel termine "Database" un insieme di concetti differenti che poi sono gli strati / livello che vi ho mostrato.

Dal mio, modesto, punto di vista di apprendista-artigiano del software, a me preme fissare il concetto della differenziazione del motore che opera su una base dati rispetto all'organizzazione delle informazioni contenute nella base dati stessa.

Continuando con l'esempio della rubrica e paragonandolo ad una rubrica di carta dei

tempi andati possiamo dire che:

1) la maschera utente corrisponde all'involucro, la rilegatura alle paginette, alla loro suddivisione ed ergonomia;

2) il motore DBMS corrisponde a noi stessi che cerchiamo, modifichiamo e inseriamo con la penna sulla carta le informazioni di rubrica;

3) il Data Base corrisponde a:

- dettaglio e informazioni predefinite che la rubrica cartacea ci permette di inserire: Nome, cognome, indirizzo, telefono, cellulare, etc.. etc.. (salute!!!)
- il dato vero e proprio che corrisponde alle informazioni inserite: ste, lampi, via le mani di qua 38, 555-*****, 335-879*****, etc.. etc.. (ri-salute!!!).

Lezione 1.3 - RDBMS: i database relazionali - 1^ parte

Finora abbiamo usato sigle come DB, DBMS che rappresentano in maniera più o meno estesa delle tipologie di database generici.
Dico questo poichè la struttura con cui può essere disegnata una base di dati non è univoca.
Dipende dalle nostre capacità di "architetti" ma anche dalle modalità con cui il DBMS ci permette di costruire le tabelle e le relazioni fra loro.

Tra le architetture che troviamo sul mercato possiamo annoverare i database di tipo gerarchico, relazionale, multidimensionale, object-oriented e, se ne parla da poco tempo, column-based.

Come avrete intuito dal titolo della pubblicazione proveremo ad addentrarci nei database relazionali.

Il pioniere del modello relazionale fu un certo Edgar F. Codd che elenco ben 12 regole

per definire un vero e puro database relazionale.
Non ve le elencherò tutte ma solo quelle che per me sembrano importanti e adatte all'approfondimento di questo libro:

1) Il dato deve essere memorizzato in tabelle in maniera univoca (quindi una sola volta) sulle righe specificando nelle colonne i suoi diversi attributi.

2) Dotato di funzionalità relazionali di "parentela" fra le tabelle.

3) Vincoli di integrità tra le entità rappresentate in tabelle e relazioni.

Per quel che riguarda il primo punto abbiamo già parlato di tabelle nell'esempio della RUBRICA. Quel che rimane da dire che in un database relazionale dobbiamo limitare la ripetizione delle informazioni e cercare di unicizzare i dati in tabella. Questo significa che la riga in cui registriamo il signor Mario Rossi in rubrica sarà una sola in modo tale che io non abbia dubbi quando estraggo il dato che l'informazione che mi viene fornita sia unica.
Il sistema RDBMS mi viene incontro evitando di farmi inserire dati duplicati fornendo all'architetto del DB dei controlli che imposti sulle colonne riescono a evitare tale problema, i vincoli di unicità, che vedremo successivamente.

Lezione 1.3 - RDBMS: i database relazionali - 2^ parte

Se io prendo un documento tipo la fattura di una azienda emessa da un fornitore a fronte di una vendita di un prodotto abbiamo degli elementi, delle entità in relazione fra loro.
Queste entità sono il cliente, il fornitore, i prodotti, la fattura e mi fermo a questo livello di dettaglio.

Se io voglio registrare queste informazioni in un database posso costruire delle tabelle che tengono memorizzati i dati in questione.
Quindi costruirò una tabella clienti, una tabella fornitori, una tabella prodotti e una

tabella fatture.
Queste tabelle come le entità rappresentate hanno delle relazioni fra di loro che permettono di tenere legate le informazioni cliente-fattura-prodotto-fornitore garantendo una buona organizzazione del database stesso.

Infatti il fornitore sarà legato al prodotto, la fattura sarà legata sia ai clienti, sia ai prodotti e sia ai fornitori, nel senso che per avere un senso informativo ha bisogno delle informazioni presenti nelle altre tabelle elencate.
Questa è la funzionalità relazionale che si costruisce fra le tabelle. Ossia il secondo punto della lista.

Possiamo immaginare i legami in questo modo:

```
( Lista dei Fornitori )
    |               |
( Fatture ) – ( Lista dei Prodotti )
    |
( Lista dei Clienti )
```

Il terzo punto si collega molto ai precedenti poiché ne è il completamento.
Questo perché un insieme di tabelle non possono dirsi appartenenti ad un modello relazionale se non utilizzano i vincoli di integrità fra di loro.
I vincoli di integrità sono dei controlli che garantiscono che la fattura numero “9” sia relativa alla merce “z” del fornitore “gamma”.

Questo significa che legano i dati di tabelle differenti fra di loro in modo coerente senza mischiare la fattura “9” con il fornitore “beta” che magari fornisce la merce “xy”.

Tornando allo schema di prima:

```
( Fornitore  “Gamma” )
```

```
      |                |
( Fattura n° 9 ) – ( Prodotto “Z” )
       |
( Cliente “Pippo” )
```

Lezione 2 - RDBMS: Prima, seconda e terza forma normale

Introduciamo il concetto principe dei database relazionali: la normalizzazione dei dati. Con questo concetto si intente fondamentalmente l'organizzazione efficiente ed efficace dei dati.

Efficiente per consentire un risparmio di spazio e di tempi nella memorizzazione dei dati e nel loro recupero.

Efficace è inteso qui nel senso di mantenere la coerenza del dato nella sua rappresentazione della realtà, ossia senza la distorsione del risultato ottenuto nella sua memorizzazione.

Vengono quindi presentate le cosiddette “forme normali” del dato. Esistono almeno cinque forme normali a cui una Base Dati può essere ricondotta. Noi per semplicità ci fermeremo alla terza forma normale.

L'obiettivo che ci si pone è quello di scomporre la nostra base dati in maniera tale da migliorarne la qualità. Se prendiamo il nostro esempio potremo scomporre la tabella Fornitore in diverse tabelle senza perdere coerenza di significato e, anzi, guadagnando in termini di spazio occupato e mantenendo livelli di performance accettabili.

Ma andiamo al sodo!

La 1^ forma normale è rispettata se:

- non esistono righe che contengono gruppi di dati uguali nella stessa tabella, eventuali righe che abbiano gruppi di dati ripetuti devono essere esternalizzati su un'altra tabella legata alla precedente da una relazione di parentela. L'esempio più semplice è quella per cui si ripete il nome e cognome di un voce della rubbrica per inserire più righe per memorizzare più numeri di telefono:

questi numeri di telefono devono essere esternalizzati in altra tabella figlia della tabella da cui siamo partiti.

- ogni tabella ha una sua chiave o colonna univoca (primary key) che rappresenta la riga in questione senza ambiguità.

La 2^ forma normale riguarda quelle tabelle che hanno una chiave primaria composta ossia rappresentata da più colonnd. Con questa considerazione **la 2^ forma normale è rispettata se:**

- la 1^ forma normale è rispettata;

- ogni colonna che non sia chiave primaria dipende funzionalmente dall'intera primary key. Ad esempio nel nostro caso della tabella rubrica, l'indirizzo della persona memorizzata è riferito alla chiave primaria del cognome + nome e non solo alla chiave parziale del campo cognome. Questo perchè potrebbero esserci dei nomi propri di persona differenti per lo stesso cognome e se l'indirizzo dipendesse solo dal cognome non sarebbe più un informazione riconducibile al “cognome + nome” esatto.

La 3^ forma normale è rispettata se:

- la 2^ forma normale è rispettata;

- non esistono attributi non chiave che dipendono da altri attributi non-chiave, ad esempio: il codice dell'area geografica di appartenenza del fornitore la descrizione di tale area. In questo caso bisogna esternalizzare la descrizione in un'altra tabella in cui portiamo il dominio di valori e le chiavi primarie degli stessi. Sulla tabella originaria rimarrà il campo chiave corrispondente alla chiave primaria presente sulla tabella creata.

Esistono poi ulteriori livelli di normalizzazione come il 4° e il 5° che regolano i legami di tipo “molti a molti” fra due tabelle. Spesso però la normalizzazione fino a 5° livello può generare problemi di performance nella lettura/scrittura dei dati e quindi la tralasciamo poiché va oltre gli obiettivi di questo manuale.

Bisogna dire che l'obiettivo che un architetto di Base Dati si propone è quello di rispettate almeno le tre forme normali che ho indicato, ma è anche vero che spesso nei progetti bisogna mediare fra esigenze di performance ed esigenze di leggibilità del dato e in alcuni casi particolari rispettare pedissequamente le tre forme normali può risultare un utopia inutile da raggiungere. Nella mia esperienza di vita lavorativa non ho mai trovato e non ho mai creato una Base Dati “normalizzata” in maniera perfetta.