

# Database relazionali

Archiviare in Linux:

costa poco, funziona bene.

di Marco Iannacone <ianna@iol.it>

## Prima parte/ i vari sistemi di archiviazione dati

L'incredibile diffusione che Internet ha avuto negli ultimi anni ha reso consapevole il grande pubblico dell'esistenza di tecnologie di comunicazione efficienti e a basso costo alternative a quelle tradizionali. Uomini di marketing, dirigenti, ma anche casalinghe e studenti hanno subito apprezzato l'efficacia di questo mezzo. Internet tuttavia sta compiendo, più lentamente, un'altra rivoluzione: sta spostando l'attenzione delle aziende dal pc desktop alle tecnologie client-server.

Si tratta di una rivoluzione altrettanto importante che spinge tutto il settore dell'Information Technology ad adattarsi a questa nuova esigenza.

Ripensando per un attimo alle tecnologie in uso alle origini dell'era informatica, osserviamo che la potenza elaborativa era centralizzata (per motivi di costi e prestazioni) nei grossi mainframe che eseguivano tutte le operazioni di calcolo. Si trattava

di applicazioni monolitiche utilizzabili attraverso l'uso di terminali stupidi. Le procedure che

accedevano a sorgenti di dati comuni creavano duplicati di codice.

Le applicazioni risiedevano in un solo luogo ed erano in grado di supportare centinaia di utenti contemporaneamente.

Da allora, l'introduzione degli home e personal computer e la loro evoluzione hanno messo a disposizione sulla scrivania di chiunque applicazioni (per singolo utente) in grado di coprire le esigenze SOHO (=Small Office Home Office).

Ovviamente tra le due filosofie (mainframe e pc standalone) non esiste una intermedia che presenta i vantaggi di entrambi ma non i difetti: la filosofia client-server.

E' proprio questa la filosofia della Rete che grazie all'adozione di architetture Unix riesce a rendere disponibile tutta la potenza dei server indipendentemente dalle caratteristiche del computer usato per il collegamento.

Come molti sapranno, grazie al lavoro dell'Università di Berkeley e di Linus Torvalds esistono alcune versioni di Unix di pubblico dominio (rispettivamente FreeBSD e Linux) che possono essere installate sui normali pc che avete in casa (anzi, la maggior parte degli Internet provider hanno basato la loro architettura proprio su questi sistemi).

Un sistema operativo come Linux (sia esso Slackware, Debian, Caldera o RedHat), infatti, viene distribuito con tutti i componenti necessari per poter diventare un potente server ed è quindi l'ideale per soddisfare le esigenze di qualsiasi azienda (siano esse rivolte ad Internet o Intranet).

In qualsiasi tipo di organizzazione, molte attività e risorse sono dedicate alla raccolta, all'archiviazione, all'elaborazione e allo scambio delle informazioni secondo procedure studiate per raggiungere determinati obiettivi.

Risulta quindi molto interessante la possibilità di realizzare un database accessibile attraverso un'interfaccia web che svincola da qualsiasi piattaforma hardware i pc client, abbatte notevol-



Marco Iannacone, si occupa di UNIX, networking e Internet come sistemista e consulente. Lavora come System Administrator e responsabile dei servizi Internet presso la Dun & Bradstreet Kosmos e collabora frequentemente con la rivista *Inter.net*

## INDIPENDENZA DEI DATI

L'indipendenza dei dati può essere definita come l'immunità delle applicazioni ai cambiamenti nella struttura dei dati. In particolare possiamo distinguere tra due livelli: indipendenza logica e indipendenza fisica.

**Indipendenza fisica:** significa che la strategia di memorizzazione fisica dei dati (sui dischi) può cambiare senza influenzare i programmi applicativi che pertanto non devono essere né riscritti né modificati.

**Indipendenza logica:** significa che la memorizzazione logica dei dati (il tracciato degli archivi) o la loro organizzazione può essere cambiata dal DBA (Database Administrator) senza dover modificare in alcuna parte le applicazioni in uso. Naturalmente il livello di corrispondenza tra natura logica e

quella fisica dei dati nelle applicazioni più frequentemente utilizzate, influisce pesantemente sulle prestazioni ottenibili dal sistema.

Un disegno efficiente contribuisce a minimizzare l'uso delle risorse elaborative e a massimizzare le prestazioni.

Ovviamente è sempre possibile intervenire a posteriori sulle strutture dati per ottenere migliori prestazioni (operazione di *tuning*), senza per questo dover modificare i programmi, ma essa sarà tanto più pesante quanto più nei programmi venga richiesta una strategia di navigazione tra i dati.

Nei DBMS relazionali, dove la strategia viene decisa dal DBMS solo al momento dell'esecuzione, l'attività di tuning risulta essere anche un fatto di logica applicativa oltre al discorso del disegno fisico.

mente i costi di training e consente, volendo, di utilizzare dei Network computer.

Risulta quindi una scelta logica, chi ha a disposizione un budget ridotto, creare questo sistema attorno ad un'architettura Linux. E' proprio di questa possibilità che parleremo in questo articolo, affrontando ora un po' di teoria relativa ai database e dedicandoci in un prossimo numero a descrivere la realizzazione di un'applicazione vera e propria basata su un database relazionale con buone prestazioni e costi davvero ridotti.

Verso la fine degli anni settanta si sono sviluppate differenti tecnologie per la gestione dei dati mediante le quali sono stati implementati sistemi informativi basati su un insieme di dati persistenti ed un insieme di procedure usate per accedere ed aggiornare gli stessi.

I primi sistemi erano basati sull'uso di archivi separati (per es. ISAM e VSAM).

A partire da questa tecnologia, si è passati ad un approccio in cui i dati sono integrati in un unico insieme o base dei dati. Si tratta di una raccolta di dati INTEGRATA (cioè costituita da insiemi comunicanti e correlati tra loro) e UTILIZZABILE SENZA RIDONDANZE (ripetizioni) da diversi utenti o programmi applicativi; inoltre è organizzata secondo le relazioni logiche esistenti tra i dati ed è pertanto indipendente dalla rappresentazione interna del calcolatore.

Come accennavamo, nei sistemi ad archivi tradizionali programmi diversi utilizzano archivi diversi con spreco di memoria (dovuto alle duplicazioni) e situazioni di

incoerenza dei dati per aggiornamenti, etc.

Una base di dati integrata, ma non organizzata secondo le relazioni logiche esistenti tra i dati, vincola i programmi alla conoscenza della rappresentazione fisica dei dati stessi ed impedisce la standardizzazione delle applicazioni e la trasportabilità dei dati.

La soluzione ottimale è quindi rappresentata da una base di dati integrata, organizzata e indipendente il cui controllo viene gestito da un DBMS (DataBase Management System). Si tratta di un insieme di procedure, centralizzate o distribuite, in grado di definire la base dati e selezionare le strutture dei dati necessarie a memorizzare e ricercare le informazioni (interattivamente o mediante un linguaggio di programmazione).

Esistono tre principali tipi di DBMS:

# Berkeley

University of California

[About UC Berkeley](#)

[News & Events](#)

[Departments](#)

[For Students](#)

[For Faculty & Staff](#)

[Alumni  
&  
Friends](#)



[Message  
from the  
Chancellor](#)



# Hughes Technologies

## - a struttura gerarchica (come IMS e System 2000)

nei quali le relazioni tra i dati sono espresse secondo un criterio di gerarchie: per accedere ad un dato occorre prima accedere a quello ad esso connesso di livello gerarchicamente superiore

## - a struttura reticolare (NETWORK STRUCTURE)

nei quali le relazioni tra i dati sono espresse mediante concatenazione tra dati tra loro correlati; a differenza delle strutture gerarchiche, in cui ogni *figlio* può avere un solo padre, in quelle reticolari ogni *membro* può avere un numero qualsiasi di proprietari.

## - a struttura relazionale

nei quali le relazioni tra i dati sono espresse in modo da apparire dati esse stesse

Proprio quest'ultimo tipo di struttura si è rivelata la soluzione più interessante per la sua semplicità e facilità d'uso, tanto da consentire l'utilizzo delle informazioni archiviate anche in modi non previsti nella fase di progettazione del database o mediante l'uso di linguaggi di alto livello.

La semplicità dei meccanismi di astrazione del modello relazionale, infatti, ha consentito lo sviluppo di semplici linguaggi di interrogazione tra cui il più noto (e più usato) è l'SQL (Structured Query Language).

Tali sistemi sono caratterizzati, oggi, da prestazioni eccellenti e da un insieme di funzionalità che assicurano la segretezza, la sicurezza e la qualità dei dati.

Le varie operazioni fornite da un DBMS sono espresse tramite uno o più linguaggi.

Normalmente un DBMS fornisce un DDL (Data Definition Language) che permette di specificare lo schema della base di dati, dove per schema si intende la specifica

di un insieme di relazioni.

Tale specifica include: il nome di ogni relazione,

il nome

e il dominio (la tipologia del dato) di tutti gli attributi che indicano una relazione, oltre agli eventuali vincoli di integrità semantica - ad esempio il vincolo che un attributo deve assumere valori diversi dal valore nullo.

Inoltre un DBMS fornisce un DML (Data Manipulation Language) che permette le operazioni di accesso e che è spesso indicato come linguaggio di interrogazione (query language). Si tratta quindi di un insieme di operatori per manipolare a livello di singolo record i dati contenuti nel database. In aggiunta a questi tipi di linguaggi, un DBMS ha un ulteriore linguaggio che permette l'esecuzione di operazioni di controllo e amministrazione della base dati.

Tale linguaggio, spesso indicato come DCL (Data Control Language), offre funzionalità quali la gestione delle autorizzazioni (dei permessi) e delle risorse fisiche (ad esempio allocazione di indici) in modo da assicurare un accesso agevole ed efficiente ai dati.

Forse ci siamo eccessivamente dilungati nella descrizione delle qualità di un database di tipo relazionale, ma i vantaggi da esso offerti rispetto ad applicazioni tradizionali sviluppate con i sistemi di sviluppo single user sono enormi. Dal punto di vista dello sviluppo, i database relazionali mettono a disposizione una serie di interfacce e di tool che non pongono limiti alla creatività dei programmatori.

E' possibile, infatti, sviluppare applicazioni direttamente in C utilizzando le API messe a disposizione dal database; oppure effettuare operazioni in maniera interattiva utilizzando i client che vengono forniti insieme al DBMS; oppure utilizzare i client passandogli degli script da eseguire in modalità batch; oppure accedervi utilizzando JDBC (l'interfaccia Java per accedere ai database) e sviluppando un applicativo in Java; o ancora, accedere attraverso pagine web che contengono istruzioni SQL embedded che vengono interpretate on-the-fly da particolari CGI... Ebbene, analizzeremo tutto ciò direttamente nel prossimo articolo in cui realizzeremo una piccola applicazione.

## Per ulteriori approfondimenti

### SISTEMI DI BASI DI DATI ORIENTATE AGLI OGGETTI

E. Bertino - LD. Martino - ADDISON-WESLEY MASSON

### IL LINGUAGGIO SQL

Cuppellini - Testori - Vaggi - FRANCO ANGELI

FreeBSD

