Server Web basati su mSQL e perl mini HOWTO

Oliver Corff, corff@zedat.fu-berlin.de v0.1, 17 settembre 1997

Questo Mini HOWTO, fortemente ispirato dall'articolo di Michael Schilli *Gebunkert: Datenbankbedienung mit Perl und CGI*, pubblicato nel computer magazine tedesco iX nell'agosto 1997, descrive come costruire un database client/server SQL utilizzando WWW e HTML per l'interfaccia utente. Documentazione tradotta da Gianmario Parisi, gianmario.parisi@libero.it.

Contents

1 Scopi di questo documento

1.1 Destinatari

Chiunque voglia installare un server database per il web ma non sa quale software è necessario e come installarlo dovrebbe trarre beneficio dalla lettura di questo testo. Questo testo fornisce tutte le informazioni necessarie per ottenere un database SQL per web server funzionante; esso *non* riguarda alcun dettaglio della programmazione CGI, né spiega il linguaggio SQL. Sono disponibili dei libri eccellenti su entrambi gli argomenti, ed è intenzione di questo testo fornire una piattaforma funzionante sulla quale un utente può in seguito studiare la programmazione CGI e SQL.

Per ottenere una piccolo sistema SQL funzionante (non l'esempio del sistema di prenotazione di una grande compagnia aerea, o il database per la gestione di una missione spaziale) sarà sufficiente disporre del software descritto in questo testo e la documentazione che lo accompagna. Il manuale utente di msql (un database introdotto nel testo) fornisce sufficienti informazioni su SQL per costruire il proprio database.

Il lettore di questo testo dovrebbe avere una buona conoscenza su come ottenere file via `ftp` se non ha accesso ai CD-ROM, ed una comprensione di base di come ottenere file binari eseguibili partendo dai sorgenti. In ogni modo, tutti i passi spiegati nel testo sono stati provati su sistemi reali e dovrebbero funzionare sul sistema del lettore.

1.2 Convenzioni usate nel testo

Un comando utente:

```
# make install
```

Uscita a video d un programma:

```
    Programma installato. Leggere README per particolari su come iniziare.
```

Codice d'esempio:

```
# Mio commento
char letter;
```

2 Introduzione

Si può assumere con sicurezza che database con grandi volumi di dati o una complessa struttura relazionale (come, probabilmente, un database lessicale per linguaggio naturale) deve essere accessibile a molti utenti ed operatori simultaneamente. Idealmente, dovrebbe essere possibile usare differenti piattaforme hardware e software esistenti che possono essere combinate nel sistema. Allo scopo di ridurre i costi di implementazione, solo un sistema, il database server, deve essere potente. Le stazioni utente devono tipicamente mostrare i dati e accettare i comandi, ma l'elaborazione è eseguita su una sola macchina, cosa che porta al nome di database client-server. Inoltre, l'interfaccia utente dovrebbe essere facile da mantenere e dovrebbe richiedere il meno possibile sul lato client.

Un sistema che soddisfa questi criteri può essere costruito attorno agli elementi seguenti tra protocolli, concetti e software:

Linux

fornisce il sistema operativo. È una implementazione stabile di Unix che fornisce autentici servizi multi-utente e multi-tasking con pieno supporto di rete (TCP/IP e. a.). Eccetto i costi effettivi dei supporti e di spedizione, è disponibile gratuitamente in forma di cosiddette distribuzioni che di solito includono tutto il necessario dal SO base all'elaborazione testi, scripting, sviluppo software, generatori di interfacce, ecc.

HTML

è l'Hypertext Markup Language usato per costruire interfacce verso sistemi di rete quale le Intranet ed il World Wide Web. HTML è molto semplice è può essere prodotto con un editor di testo ASCII.

Browser

sono applicazioni con interfaccia testuale (per es. Lynx) o grafica (per es. Mosaic, Netscape, Arena ecc.) che accettano, valutano e mostrano documenti HTML. Sono le uniche parti di software utilizzate direttamente dall'utente del database. Utilizzando i browser, è possibile mostrare vari tipi di dati (testo, immagini) e comunicare con server http (vedi nel seguito) su ogni modello di computer per cui sia disponibile un browser.

server http

forniscono l'accesso all'area di un computer host dove sono immagazzinati i dati destinati ad uso pubblico su una rete. Essi "comprendono" il protocollo http e procurano le informazioni richieste dall'utente.

SQL

lo Structured Query Language (linguaggio strutturato per interrogazioni) è un linguaggio per manipolare dati in database relazionali. Ha una grammatica molto semplice e costituisce uno standard ampiamente supportato. Database basati su SQL sono diventati il nucleo del classico concetto di database client/server. Ci sono molti famosi sistemi SQL disponibili, come Oracle, Informix, ecc., e poi c'è anche msql disponibile a costo molto basso o nullo se usato in ambienti accademici e di istruzione.

CGI

la Common Gateway Interface è l'interfaccia di programmazione tra il sistema che mantiene i dati (nel nostro caso un sistema basato su SQL) ed il protocollo di rete (HTML, naturalmente). I CGI possono essere costruiti attraverso molti linguaggi di programmazione, ma un linguaggio particolarmente popolare è il perl.

perl

è un linguaggio di scripting estremamente potente che combina tutti i meriti del C, di diversi linguaggi shell e dei linguaggi per manipolazione di file come awk e sed. Perl ha moltissime interfacce modularizzate e può essere usato, ad esempio, per controllare database SQL.

3 Procedura di Installazione

3.1 Requisiti Hardware

Non possono essere fatte affermazioni generali riguardo i requisiti hardware di un server database. Troppi fattori dipendono dal numero di utenti, dal tipo di applicazione, dal carico di rete ecc. In piccoli ambienti con pochi utenti e scarso traffico di rete una macchina i486-equivalente con 16 Mb di RAM può essere completamente sufficiente. Linux, il sistema operativo, è molto efficiente in termini di risorse, e può fornire abbastanza potenza per eseguire un'ampia varietà di applicazioni allo stesso tempo. Naturalmente, processori più veloci e più RAM significano più velocità, ma molto più importante del processore è l'ammontare di RAM. Più RAM ha il sistema, meno esso è costretto ad eseguire swap di processi su disco nel caso la memoria diventi un collo di bottiglia.

Disponendo di 32 MB di RAM ed un bus PCI, operazioni di ricerca e ordinamento possono essere eseguite senza ricorso a file di swap ecc., ottenendo prestazioni velocissime.

Il modello di installazione descritto in questo articolo è stato realizzato su un IBM 686 (133 MHz) con 32 MB di RAM ed un hard disk IDE da 1.2 GB. Assumendo che il processo di installazione parta da zero, viene fornita una lista dei passi necessari.

3.2 Requisiti Software

Il software descritto in questo articolo è disponibile su Internet o su CD-ROM. Sono stati usati i seguenti prodotti:

- Red Hat Linux PowerTools: Red Hat 4.2 - 6 CD Complete Easy-to-Use estate '97; in alternativa `http://www.redhat.com`;
- msql SQL database server: è ora disponibile in due versioni. Le versioni differiscono nel numero di transazioni che possono gestire, nell'interfaccia di amministrazione, ecc. La versione più vecchia, 1.0.16, è disponibile dai mirror di Sunsite. Gli eseguibili ELF possono essere trovati a `sunsite:apps/database/sql/msql-1.0.16` o su CD-ROM (disco 4 di InfoMagic Linux Developer's Resource, 6-CD, Dicembre 1996) o alternativamente dalla seguente URL: `http://www.infomagic.com`.

  La versione più nuova, 2.0.1, può essere direttamente ottenuta dal homepage della Hughes in Australia (`http://www.hughes.com.au`) o da numerosi mirror sparsi per il mondo;
- perl da CPAN: Il Comprehensive Perl Archive Network. Walnut Creek CDROM, ISBN 1-57176-077-6, Maggio 1997;
- programmi d'esempio CGI di Michael Schilli's dalla rivista tedesca iX 8/1997, pagine 150–152, disponibile via ftp da `ftp.uni-paderborn.de:/doc/magazin/iX`;

3.3 Installazione del sistema operativo

Linux è installato nella forma della distribuzione Red Hat 4.2. Allo scopo di installare con successo, la macchina deve avere un drive CD-ROM accessibile da DOS, un drive CD-ROM avviabile (bootable), altrimenti dev'essere preparato un disco di boot seguendo le istruzioni sul CD di Linux.

Durante l'installazione l'utente può selezionare e configurare numerosi pacchetti software. È conveniente selezionare i seguenti elementi:

- supporto di rete TCP/IP,
- il server http Apache,
- il linguaggio di scripting perl
- il sistema X Window
- i browser Arena (grafico) e Lynx (testuale).

Tutti questi pacchetti sono forniti con la distribuzione Linux. Se questi pacchetti non vengono installati ora, si avrà la possibilità di farlo in seguito con l'assistenza di glint, il gestore per l'installazione software con interfaccia grafica ed intuitiva. Assicurarsi di operare come utente root durante l'installazione dei pacchetti software.

Va oltre gli scopi di questo articolo descrivere la procedura di installazione ed inizializzazione della rete. Consultare la documentazione in linea (manpage, HTML, texinfo) e stampata (Linux Bible, ecc. ecc.).

La procedura di installazione di Red Hat è matura e richiede poche attenzioni da parte dell'utente oltre alle scelte usuali (come fornire il nome host ecc.). Una volta terminata con successo l'installazione, il sistema è sostanzialmente pronto per l'esecuzione.

Installare il sistema X Window non è obbligatorio per un server puro, ma ciò rende l'accesso al sistema ed il test molto più semplice. La procedura di installazione di X è eseguita con uno qualsiasi tra diversi programmi; XF86Setup offre la più estesa procedura guidata e necessita del minor numero di operazioni manuali per la gestione di fastidiosi dettagli (come programmazione del clock video, ecc.). L'unico requisito è che il software possa rilevare l'adattatore video. Un adattatore video accelerato economico (come schede basate su Trio S64 precedenti a S64UV+) di solito funziona "al volo".

A questo punto assumiamo che il nostro sistema sia attivo ed in esecuzione e che Apache, Perl e X Window siano stati installati con successo. Si assumerà inoltre che tutte le strutture standard come come file e directory siano mantenute come definito nell'installazione. Ultimo, ma non meno importante, lasceremo l'hostname cosí com'è, e accettiamo in questo momento il nome `localhost`. Useremo questo nome per testare l'installazione. Una volta che l'intero sistema funziona può essere aggiunto il vero nome. Notare che il setup della rete richiede anche di editare il file `/etc/hosts`, tra gli altri. Idealmente questo dovrebbe essere fatto con gli strumenti di amministrazione forniti all'utente root.

3.4 Il server http

Il server http fornito con Linux è conosciuto come Apache dagli umani e come httpd dal sistema. La manpage (man httpd) spiega come installare ed avviare il demone http (quindi http*d*) ma, come detto, se l'installazione è avvenuta senza problemi, il server dovrebbe essere in esecuzione. Si può verificare l'albero delle directory: deve esserci una directory `/home/httpd/` con tre sottodirectory: `../cgi-bin/`, `../html/` e `../icons/`. In `../html/` deve esserci un file `index.html`. In seguito modificheremo o sostituiremo questo file con l'`index.html` effettivo definito da noi. Tutte le informazioni di configurazione sono registrate in `/etc/httpd/conf/`. Il sistema è ben preconfigurato e non richiede un ulteriore setup se l'installazione non ha subito errori.

3.5 I Browser

Ci sono essenzialmente tre tipi di browser disponibili per Linux: sistemi puramente testuali come Lynx, semplici e sperimentali come Arena (gratis!) e commerciali come Netscape (Shareware!) con supporto Java. Mentre Lynx e Arena sono forniti con Linux, Netscape deve essere recuperato da altre fonti. Netscape è disponibile in forma binaria precompilata per Linux su architetture ix86 e potrà essere eseguito "al volo" appena estratto dall'archivio.

3.5.1 Configurazione di Lynx

Una volta avviato, Lynx cercherà una URL predefinita solitamente non molto significativa se il sistema non ha un accesso permanente ad Internet. Per cambiare la URL predefinita (e molti altri dettagli di configurazione) l'amministratore di sistema dovrebbe editare `/usr/lib/lynx.cfg`. Il file è grande, circa 57000 byte e contiene informazioni a volte contraddittorie. Esso dichiara la propria home come `/usr/local/lib/`. Non lontano dall'inizio del file c'è una linea che comincia con `STARTFILE`. Rimpiazzare questa linea con la voce seguente: `STARTFILE:http://localhost` assicurandosi che non siano inseriti spazi ecc. :

```
# STARTFILE:http://www.nyu.edu/pages/wsn/subir/lynx.html
STARTFILE:http://localhost
```

Dopo aver salvato il file, Lynx dovrebbe mostrare il nostro documento `index.html` se avviato senza argomenti.

3.5.2 Configurazione del browser Arena

Arena dapprima cerca la propria URL predefinita quando lanciato senza argomenti. Questa è codificata nell'eseguibile ma può essere sovrascritta tramite la variabile d'ambiente `WWW_HOME`. L'amministratore di sistema può inserire la linea `WWW_HOME="http://localhost"` in `/etc/profile`. La variabile deve essere esportata, o tramite una istruzione separate (`export WWW_HOME`) o appendendo `WWW_HOME` all'istruzione export esistente:

```
WWW_HOME="http://localhost"
export WWW_HOME
```

Al successivo login, la nuova URL predefinita sarà nota ad Arena a livello di sistema.

3.5.3 Installazione e configurazione di Netscape

Netscape è un prodotto commerciale e quindi non incluso nelle distribuzioni Linux. È scaricabile da Internet o disponibile in collezioni software su CD-ROM. Netscape giunge in forma binaria precompilata per ogni importante piattaforma hardware. Per scopi di installazione, è utile creare una directory `/usr/local/Netscape/` dove scompattare l'archivio. I file possono essere lasciati sul posto (eccetto per le librerie Java: seguire le istruzioni nel file `README` accluso ai binari Netscape), ed è sufficiente creare un link in `/usr/local/bin/` eseguendo il comando

```
# ln -s /usr/local/Netscape/netscape .
```

dalla directory `/usr/local/bin/`.

Netscape è ora pronto per l'uso e può essere configurato attraverso il menù 'Options". In "General Preferences" c'è una scheda "Appearance" con il campo "Home Page Location". Immettere qui `http://localhost` e non dimenticare di salvare le opzioni (attraverso “Options” — “Save Options”) prima di uscire da Netscape. All’avvio successivo, Netscape mostrerà l’homepage di Apache.

3.6 Cooperazione tra Apache e i Browser

Si può ora condurre il primo test reale del browser e del server http: avviare uno dei browser disponibili e la pagina `Apache: Red Hat Linux Web Server` apparirà. Questa pagina mostra la locazione dei file e altre informazioni basilari sull’installazione del server http. Se questa pagina non viene mostrata controllare se i file menzionati sopra sono nel posto giusto e se la configurazione del browser è corretta. Chiudere i file di configurazione editati prima di lanciare il browser di nuovo. Se tutti i file sono a posto e la configurazione del browser sembra corretta, esaminare il setup di rete della propria macchina. L’hostname potrebbe essere differente da quello immesso nella configurazione, o il setup di rete potrebbe essere in sé non corretto. È molto importante che `/etc/hosts` contenga almeno una linea come

```
127.0.0.1                localhost localhost.localdomain
```

che implica la possibilità di connettersi localmente alla propria macchina. Questo è verificabile richiamando uno dei programmi di rete che richiedono un hostname come argomento, come `telnet localhost` (ammesso che `telnet` sia installato). Se l’esecuzione fallisce occorre verificare la configurazione di rete prima di continuare.

3.7 Il Motore Database e la sua installazione

L’installazione del database richiede poca preparazione in più rispetto ai passi precedenti. Ci sono pochi motori database SQL disponibili, con differenti prerequisiti per l’amministrazione e l’esecuzione, ed uno dei migliori è msql, o “Mini-SQL” di David Hughes. msql è shareware. A seconda della versione utilizzata, è previsto un addebito di 250 dollari USA o più per siti commerciali, 65 dollari o più per utenti privati, mentre enti educativi, formativi e organizzazioni no-profit registrate possono usare il software gratuitamente. I costi esatti sono forniti nelle note di licenza della documentazione del database. Le cifre indicate sono solo un indicatore approssimativo.

Alcune parole per giustificare la scelta di msql da parte dell’autore. Prima di tutto, c’è l’esperienza personale. Durante la ricerca di motori database, msql si è dimostrato il più semplice da installare e mantenere, e fornisce una sufficiente copertura del linguaggio SQL tale da soddisfare le esigenze generali. Solo durante la stesura di questo articolo l’autore ha scoperto le seguenti parole nell’Alligator Descartes’ DBI FAQ (perl database interface FAQ):

> Dal punto di vista dell’autore, se l’insieme di dati è relativamente piccolo, con tabelle inferiori al milione di righe, e meno di mille tabelle in un dato database, allora mSQL è una soluzione perfettamente accettabile per il vostro problema. Questo database è estremamente economico, meravigliosamente robusto ed ha un supporto eccellente. [...]

Msql è disponibile attualmente in due versioni, msql-1.0.16 e msql-2.0.1, che differiscono in prestazioni (non osservabili in progetti di piccola scala) e software allegato (la versione più recente presenta più strumenti, un proprio linguaggio di scripting ecc.). Verranno descritte entrambe le versioni di msql siccome la loro installazioni differisce in alcuni punti.

3.7.1 Installazione di msql-1.0.16

msql è disponibile in formato sorgente ed in forma binaria compilata con supporto ELF. L'uso dei binari ELF rende l'installazione semplice in quanto l'archivio `msql-1.0.16.ELF.tgz` contiene un albero di directory assoluto e completo, cosicché tutte le directory sono create correttamente quando estratte da `/`.

Se si decide di compilare msql-1.0.16 e si intende usare il pacchetto MsqlPerl piuttosto che l'interfaccia DBI (vedere una trattazione dettagliata delle differenze più oltre) tenere presente che MsqlPerl potrebbe segnalare degli errori in fase di test che richiedono l'installazione di una patch per correggere una errata gestione dell'SQL. La patch è descritta nella documentazione MsqlPerl (file `patch.lost.tables`). In particolare le richieste di MsqlPerl includeranno tre linee in `msqldb.c` dopo la linea 1400 che riporta `entry->def = NULL;`:

```
*(entry->DB) = 0;
*(entry->table) = 0;
entry->age = 0;
```

La porzione di codice dovrebbe apparire come

```
freeTableDef(entry->def);
safeFree(entry->rowBuf);
safeFree(entry->keyBuf);
entry->def = NULL;
*(entry->DB) = 0;
*(entry->table) = 0;
entry->age = 0;
```

La compilazione di msql richiede diversi passi. Dopo la scompattazione dell'archivio con i sorgenti è necessario costruire una directory destinazione. Questo si ottiene con

```
# make target
```

Se tutto va bene, il sistema risponde con

```
Build of target directory for Linux-2.0.30-i486 complete
```

Bisogna ora spostarsi nella directory appena creata ed immettere dapprima il comando

```
# ./setup
```

La sequenza `./` è necessaria per assicurarsi di eseguire il comando `setup` in questa directory e non un altro con lo stesso nome. Verrà richiesto di scegliere la locazione della directory con i sorgenti e se si desidera una installazione come utente root. Una volta che l'utente ha effettuato le scelte il sistema esegue una serie di test per verificare la disponibilità di software (compilatore, utilità ecc.) per terminare col messaggio

```
Ready to build mSQL.
You may wish to check "common/site.h" although the defaults should be
fine.  When you're ready, type  "make all" to build the software
```

Si potrà immettere

```
# make all
```

Se tutto va come previsto, si leggerà:

```
make[2]: Leaving directory '/usr/local/Minerva/src/msql'
<-- [msql] done

Make of mSQL complete.
You should now mSQL using make install

NOTE : mSQL cannot be used free of charge at commercial sites.
       Please read the doc/License file to see what you have to do.

make[1]: Leaving directory '/usr/local/Minerva/src'
```

Tutti i binari devono essere resi visibili dal percorso di ricerca creando dei link software in `/usr/local/bin/`. Spostarsi nella directory ed immettere il comando

```
# ln -s /usr/local/Minerva/bin/* .
```

dopo il quale i link saranno impostati correttamente.

3.7.2 Test di msql-1

Dopo l'installazione è possibile testare se il database funziona. Prima di ogni altra cosa il server (demone) deve essere avviato. L'amministratore di sistema con i privilegi di root immette il comando

```
# msqld &
```

(non dimenticare di aggiungere il simbolo `&`, altrimenti msql non verrà eseguito in background) dopodiché apparirà il seguente messaggio a schermo:

```
mSQL Server 1.0.16 starting ...

Warning : Couldn't open ACL file: No such file or directory
Without an ACL file global access is Read/Write
```

Questo messaggio ci informa che ogni cosa funziona ma che non esistono restrizioni di accesso. Per il momento è sufficiente avviare il demone msql da shell, ma in seguito si potrebbe desiderare che il sistema esegua automaticamente il comando per noi. Il comando deve essere inserito in uno script `rc.d` appropriato. Solo ora l'amministratore può immettere il primo comando effettivo di database:

```
# msqladmin create inventur
```

msql risponde con `Database "inventur" created.`. Come ulteriore prova, si verifichi che la directory `/usr/local/Minerva/msqldb/` contiene ora la sottodirectory vuota `../inventur/`. Potremmo manipolare il nuovo database con gli strumenti di amministrazione; queste procedure sono tutte coperte in dettaglio dalla documentazione msql.

3.7.3 Installazione di msql-2.0.1

C'è una versione più recente e più potente del server mSQL di Huges, l'installazione del quale differisce in pochi punti. L'installazione ex-novo di msql-2 implica i passi seguenti. Copiare l'archivio nel punto di estrazione previsto, per esempio `/usr/local/msql-2/`, poi estrarre i file:

```
# tar xfvz msql-2.0.1.tar.gz
```

Spostarsi nella directory radice dell'albero di installazione e immettere

```
# tar xfvz msql-2.0.1.tar.gz
```

spostarsi in `targets` e cercare il proprio tipo di piattaforma. Dovrebbe esserci una nuova sottodirectory `Linux-`*(vostra versione)-(vostra cpu)/*. Spostarsi in essa ed avviare il programma di setup:

```
# ./setup
```

C'è anche un file `site.mm` che può essere editato. Vogliamo conservare il percorso di installazione in `/usr/local/Minerva/` (predefinito per msql 1.0.16)? In questo caso modificare la linea `INST_DIR=...` di conseguenza. Altrimenti, lasciare tutto com'è.

Ora possiamo lanciare la compilazione del database

```
# make
# make install
```

Se tutto va per il vero giusto, vedremo un messaggio come:

```
[...]

Installation of mSQL-2 complete.

*********
**   This is the commercial, production release of mSQL-2.0
**   Please see the README file in the top directory of the
**   distribution for license information.
*********
```

Dopo che tutto è installato correttamente dobbiamo curarci dei dettagli amministrativi. Qui comincia la vera differenza da msql-1. Anzitutto, creare un utente responsabile della amministrazione del database.

```
# adduser msql
```

Poi si imposti `msql` come proprietario di tutti i file nella directory mSQL col comando:

```
# cd /usr/local/Minerva
# chown -R msql:msql *
```

Poi possiamo creare link per tutti gli eseguibili binari del database in `/usr/local/bin/` col comando:

```
# ln -s /usr/local/Minerva/bin/* .
```

3.7.4 Test di msql-2

Possiamo avviare il server database con il comando `msql2d &`; dovremmo ottenere una risposta simile a questa:

```
Mini SQL Version 2.0.1
Copyright (c) 1993-4 David J. Hughes
Copyright (c) 1995-7 Hughes Technologies Pty. Ltd.
All rights reserved.

        Loading configuration from '/usr/local/Minerva/msql.conf'.
        Server process reconfigured to accept 214 connections.
        Server running as user 'msql'.
        Server mode is Read/Write.

Warning : No ACL file.  Using global read/write access.
```

Ciò sembra perfetto. Il database è compilato ed installato, e possiamo ora continuare con i moduli perl siccome questi si basano in parte sulla presenza di un database funzionante per il test.

Incidentalmente, questo è anche un buon momento per stampare il manuale completo fornito con msql-2.0.1:

```
# gzip -d manual.ps.gz
# lpr manual.ps
```

Possiamo ora procedere con la compilazione delle interfacce, ma è una buona idea mantenere il server SQL attivo ed in esecuzione perché ciò rende il test delle librerie di interfaccia un po' più semplice.

3.8 Scelta delle interfacce: DBI/mSQL, MsqlPerl, Lite

Una frase spesso citata del Camel Book (la principale documentazione perl) afferma che c'è più di un modo di ottenere un risultato quando si usa perl. Questo, ahimè, resta vero anche per il nostro modello di applicazione. Ci sono fondamentalmente tre modi di accedere ad un database msql via CGI. Prima di tutto la domanda è se utilizzare o meno perl. Se usiamo perl (su ciò si focalizza questo articolo) potremo ancora scegliere tra due modelli completamente diversi di interfaccia. A fianco del perl, possiamo anche utilizzare il linguaggio scripting di msql, chiamato Lite, che è ragionevolmente semplice e simile al C.

3.8.1 DBI e DBD-mSQL

Al momento della redazione di questo articolo, l'uso della interfaccia perl DBI per l'accesso a database viene preferito. DBI ha alcuni vantaggi: fornisce un controllo unificato di accesso ad un numero di database commerciali per mezzo di un unico insieme di comandi. Il database effettivamente in uso su un dato sistema è poi contattato attraverso un driver che nasconde efficacemente le peculiarità del database al programmatore. Cosí, l'uso di DBI permette una agevole transizione tra differenti database di differenti produttori. In un singolo script è possibile contattare diversi database. Fare riferimento alle DBI-FAQ per i dettagli. C'è, tuttavia, uno svantaggio: l'interfaccia DBI è ancora in fase di sviluppo e i numeri di versione aumentano rapidamente (talvolta gli aggiornamenti si hanno in meno di un mese). Analogamente, anche i singoli driver di database sono aggiornati frequentemente e possono riferirsi a specifiche versioni dell'interfaccia di database. Utenti che effettuano nuove installazioni dovrebbero attenersi strettamente ai numeri di versione dati in questo articolo perché altre versioni possono causare problemi di compilazione e di test la risoluzione dei quali non è cosa per gente debole di cuore.

3.8.2 MsqlPerl

MsqlPerl è una libreria per l'accesso a msql direttamente da script perl. Essa scavalca l'interfaccia DBI ed è piuttosto compatta. Sebbene essa lavori bene con entrambe le versioni di msql, il suo uso non è più consigliato a vantaggio dell'interfaccia DBI generalizzata. Nondimeno, in un contesto specifico, MsqlPerl può risultare la scelta giusta grazie alle dimensioni contenute ed alla facilità di installazione. Da notare, essa ha meno dipendenze dalla versione di quelle rivelate dall'interazione di DBI con diversi driver di database.

3.8.3 Il linguaggio scripting di msql: Lite

Ultimo ma non meno importante, msql-2 possiede un suo linguaggio di script: Lite. Il linguaggio è un parente stretto del C, snellito dalle sue stranezze e arricchito con caratteristiche di tipo shell (in sintesi, qualcosa di simile ad una versione di perl molto specializzata). Lite è un linguaggio semplice ed è ben documentato nel manuale msql-2. Il pacchetto msql-2 fornisce anche una applicazione di esempio basata su Lite.

Non descriveremo Lite in questa sede perché esso è specifico di msql-2, e perché si assume che i lettori di questo articolo abbiano un interesse ed una comprensione di base del perl. Nondimeno un approfondimento di Lite è altamente raccomandato: Lite può fornire la soluzione vincente in un ambiente basato esclusivamente su msql-2 (senza ricorso ad altri database), grazie alla sua concezione semplice e diretta.

3.9 La via generale: DBI e DBD-msql

Assumiamo che perl sia stato installato durante il setup di sistema o attraverso il gestore pacchetti summenzionato. Non daremo altri dettagli qui. Per verificare che la versione di perl sia aggiornata eseguiamo:

```
# perl -v
```

perl dovrebbe rispondere col seguente messaggio:

```
 This is perl, version 5.003 with EMBED
        Locally applied patches:
        SUIDBUF - Buffer overflow fixes for suidperl security

        built under linux at Apr 22 1997 10:04:46
        + two suidperl security patches

Copyright 1987-1996, Larry Wall
```

Probabilmente, tutto è a posto. Il passo successivo include l'installazione delle librerie perl per database in generale (DBI), il driver msql (DBD-mSQL) e CGI. Il driver CGI è necessario in ogni caso. Sono necessari i seguenti archivi:

1. DBI-0.81.tar.gz
2. DBD-mSQL-0.65.tar.gz
3. CGI.pm-2.31.tar.gz (or successivo)

Qui occorre una puntualizzazione per i neofiti: l'installazione di test descritta qui funziona bene utilizzando software con *esattamente* questi numeri di versione, mentre combinazioni di altre versioni falliscono per un motivo o per l'altro. Il debug di combinazioni di versioni difettose è sconsigliabile a chi non abbia grande familiarità con i dettagli delle convenzioni di chiamata delle interfacce ecc. In alcuni casi un metodo può essere semplicemente rinominato pur effettuando lo stesso compito, ma a volte la struttura interna cambia significativamente. Quindi, ancora una volta, è necessario mantenere i numeri di versione qui indicati se si vuole operare in sicurezza, anche se nel frattempo fossero comparse versioni successive. Aggiornamenti frequenti di queste interfacce sono una regola piuttosto che un'eccezione, quindi l'installazione di versioni differenti da quelle indicate può essere fonte di problemi.

È molto importante che il driver di database per mSQL (DBD-mSQL) sia installato *dopo* l'interfaccia generica DBI.

Si comincerà creando la directory `/usr/local/PerlModules/` siccome è molto importante mantenere l'albero originale delle directory perl intatto. Potremmo anche scegliere un nome di directory diverso siccome il nome non è assolutamente critico, e sfortunatamente ciò non è specificato nei file README dei vari moduli perl. Dopo aver copiato gli archivi suddetti in `/usr/local/PerlModules/` li scompattiamo con

```
# tar xzvf [file-archivio]
```

per ognuno dei tre archivi. Non dimenticare di fornire il nome di file corretto a `tar`. Il processo di installazione per i tre moduli è essenzialmente standardizzato; solo i messaggi a schermo che mostrano passi significativi per i singoli pacchetti sono riportati nel seguito.

3.9.1 Installazione dell'interfaccia database di perl - DBI

L'interfaccia verso il database deve sempre essere installata prima del driver di database specifico. La scompattazione dell'archivio DBI crea la directory `/usr/local/PerlModules/DBI-0.81/`. Spostandosi nella directory, si trovano un file `README` (che andrebbe letto) ed un perl-Makefile (estensione .PL). Diamo il comando

```
# perl Makefile.PL
```

Il sistema dovrebbe rispondere con un lungo messaggio la cui parte importante è mostrata qui:

```
[...]
MakeMaker (v5.34)
Checking if your kit is complete...
Looks good
        NAME => q[DBI]
        PREREQ_PM => {  }
        VERSION_FROM => q[DBI.pm]
        clean => { FILES=>q[$(DISTVNAME)/] }
        dist => { DIST_DEFAULT=>q[clean distcheck disttest [...]
Using PERL=/usr/bin/perl

WARNING! By default new modules are installed into your 'site_lib'
directories. Since site_lib directories come after the normal library
directories you MUST delete old DBI files and directories from
your 'privlib' and 'archlib' directories and their auto subdirectories.

Writing Makefile for DBI
```

Questo dovrebbe andare bene, come indicato dal programma (looks good), e possiamo procedere col passo successivo:

```
# make
```

Se non si hanno messaggi di errore (il protocollo dettagliato mostrato a schermo *non* è un messaggio di errore) si testi la nuova libreria installata con il comando

```
# make test
```

Osservare le seguenti linee di output (effettuare lo scroll all'indietro con `[Shift]-[PgUp]`):

```
[...]
t/basics...........ok
t/dbidrv...........ok
t/examp............ok
All tests successful.
[...]
DBI test application $Revision: 1.20 $
Switch: DBI-0.81 Switch by Tim Bunce, 0.81
Available Drivers: ExampleP, NullP, Sponge
ExampleP: testing 2 sets of 5 connections:
Connecting... 1 2 3 4 5
Disconnecting...
Connecting... 1 2 3 4 5
Disconnecting...
Made 10 connections in  0 secs ( 0.00 usr  0.00 sys =  0.00 cpu)

test.pl done
```

Il passo finale è quello di installare tutti i file nelle directory appropriate. Ciò si ottiene col comando

```
# make install
```

Non resta altro. Se per qualche ragione l'installazione fallisce e deve essere rieseguita non dimenticarsi di eseguire prima

```
# make realclean
```

Questo rimuove i resti indesiderati della precedente installazione. Si possono anche rimuovere i file installati copiando il contenuto dello schermo (mostrato abbreviato)

```
Installing /usr/lib/perl5/site_perl/i386-linux/./auto/DBI/DBIXS.h
Installing /usr/lib/perl5/site_perl/i386-linux/./auto/DBI/DBI.so
Installing /usr/lib/perl5/site_perl/i386-linux/./auto/DBI/DBI.bs
[...]
Writing /usr/lib/perl5/site_perl/i386-linux/auto/DBI/.packlist
Appending installation info to /usr/lib/perl5/i386-linux/5.003/perllocal.pod
```

in un file, rimpiazzando ogni occorrenza di `Installing` con `rm`. Se tale file viene chiamato `uninstall` si può poi eseguire

```
# . uninstall
```

che rimuoverà i file installati.

3.9.2 Il driver msql di perl DBD-mSQL

Il driver msql può essere installato soltanto *dopo* la felice installazione dell'interfaccia generica per database di perl.

I passi fondamentali sono gli stessi di sopra; così avremo dapprima

```
# perl Makefile.PL
```

Qui, il sistema dovrebbe rispondere con un invito alla lettura della documentazione a corredo. Esso rileverà poi dove risiede msql, e chiederà quale versione è in uso.

```
$MSQL_HOME not defined. Searching for mSQL...
Using mSQL in /usr/local/Hughes

 -> Which version of mSQL are you using [1/2]?
```

inserire il numero di versione corretto. Seguiranno poche linee di testo. Osservare le seguenti:

```
Splendid! Your mSQL daemon is running. We can auto-detect your configuration!

I've auto-detected your configuration to be running on port: 1114
```

Si può ora testare il driver con

```
# make test
```

Di nuovo, segue un output piuttosto lungo. Se esso termina con

```
Testing: $cursor->func( '_ListSelectedFields' ). This will fail.
        ok: not a SELECT in msqlListSelectedFields!
Re-testing: $dbh->do( 'DROP TABLE testaa' )
        ok
*** Testing of DBD::mSQL complete! You appear to be normal! ***
```

si è sulla buona strada ed è possibile installare il driver con

```
# make install
```

Questo conclude le operazioni di installazione; il prossimo paragrafo riguarda MsqlPerl e se si è scelto l'uso di DBI può essere saltato.

3.10 L'interfaccia MsqlPerl

Se si decide di usare esclusivamente l'interfaccia MsqlPerl non occorre il driver di database generico, ma solo `MsqlPerl-1.15.tar.gz`, siccome, come detto in precedenza, MsqlPerl fornisce una interfaccia diretta tra server database e perl senza l'uso dell'interfaccia DBI. Installazione e test sono immediati.

Dopo aver eseguito `perl Makefile.PL` l'utilità make può essere avviata. Dapprima occorrerà rispondere alla domanda su dove risiede msql. Se msql è in `/usr/local/Minerva/` si potrà confermare la risposta di default.

Poi eseguire `make test`. Prima di ciò bisogna assicurarsi di avere un database chiamato `test` e di possedere i diritti di lettura/scrittura su di esso. Tale database può essere creato con

```
# msqladmin create test
```

3.11 La libreria CGI di perl

L'installazione del componente CGI di perl è il più semplice dei tre passi. È sufficiente eseguire i comandi seguenti nell'ordine dato e tutto è fatto:

```
# perl Makefile.PL
# make
# make install
```

Diversamente dai driver precedenti questa interfaccia non ha una opzione di test (`# make test`) siccome gli altri moduli *devono* essere testati in ogni caso.

Viene anche creata una sottodirectory con script CGI di esempio. È possibile copiare i contenuti di questa directory in `/home/http/cgi-bin/` ed usare il browser per sperimentare gli script.

3.12 Lista di controllo dell'installazione

Abbiamo compiuto i passi seguenti, nell'ordine dato:

1. Installazione di Linux con supporto di rete
2. Installazione di un server http (Apache)
3. Installazione di un browser (Arena, lynx o Netscape)
4. Installazione di un server SQL (msql)
5. Installazione di una interfaccia perl SQL
6. Installazione dei file CGI

Infine, è possibile fare un po' di pulizia. Tutti i rami di directory contenenti i sorgenti per le installazioni possono essere rimossi (tuttavia, non vanno cancellati gli archivi originali!) siccome tutti i file binari e la documentazione si trovano in directory differenti.

4 Esecuzione di un database di esempio

Dopo il completamento dell'installazione del sistema si può finalmente eseguire una modello di applicazione. A seconda della versione di msql installata e dell'interfaccia perl utilizzata si dovrà modificare il programma di esempio in qualche punto.

Innanzitutto, il file `index.html` posto in `/home/httpd/html/` deve essere modificato per permettere di richiamare l'applicazione database campione. È possibile porre il database (che verrà chiamato `database.cgi` o `inventur.cgi` nonostante il nome del file `perl.lst.ck`) nella directory `/home/httpd/html/test/`.

Per ottenere lo scopo, appendere una linea (naturalmente dipendente dalle scelte di installazione) simile alla seguente nel file `index.html`:

```
<LI>Test the <A HREF="test/database.cgi">Database, DBI:DBD-mSQL style!</A>
<LI>Test the <A HREF="test/inventur.cgi">Database, MsqlPerl style!</A>
```

Solitamente si dovrebbe mantenere una sola di queste due scelte, ma disponendo di entrambi i tipi di interfaccia database installata è possibile lasciare entrambe le linee così come sono. Sarà in seguito possibile comparare le prestazioni, ecc.

4.1 Adattamento dello script di esempio per MsqlPerl

Allo script campione deve essere notificato l'uso dell'interfaccia MsqlPerl. Le modifiche intervengono in diversi punti. Dapprima, vicino all'inizio del file, rimpiazzare la clausola `use`:

```
#
# use DBI;            # Interfaccia Database Generica
use Msql;
```

Poi, alla linea 27, la sintassi MsqlPerl non richiede la menzione di un driver specifico:

```
# $dbh = DBI->connect($host, $database, '', $driver) ||
$dbh = Msql->connect($host, $database) ||
```

Poi, dalla linea 33 per tutto l'intero script, bisogna modificare tutte le istanze di `do` con `query`:

```
# $dbh->do("SELECT * FROM hw") || db_init($dbh);
$dbh->query("SELECT * FROM hw") || db_init($dbh);
```

Infine, la linea 207 deve essere commentata:

```
# $sth->execute || msg("SQL Error:", $sth->errstr);
```

Inoltre, può diventare necessario scambiare tutte le chiamate `errstr` come quella nel precedente frammento di codice con `errmsg`. Anche questa scelta dipende dalla versione.

Dopo queste modifiche, lo script dovrebbe girare senza intoppi.

4.2 Adattamento dello script di esempio per for msql-2

La sintassi SQL è stata ridefinita durante lo sviluppo di msql-2. Lo script originale fallirà l'esecuzione delle istruzioni di inizializzazione tabella nelle linee 45 – 58. Il modificatore `primary key` non è più supportato da msql-2, e dovrebbe essere semplicemente evitato:

```
    $dbh->do(<<EOT) || die $dbh->errstr; # Neue Personen-Tabelle
        create table person (
# We do not need the 'primary key' modifier anymore in msql-2!
#           pn        int primary key,   # Personalnummer
            pn        int,               # Personalnummer
            name      char(80),          # Nachname, Vorname
            raum      int                # Raumnummer
        )
EOT
    $dbh->do(<<EOT) || die $dbh->errstr; # Neue Hardware-Tabelle
        create table hw (
# We do not need the 'primary key' modifier anymore in msql-2!
#           asset int primary key,       # Inventurnummer
            asset int,                   # Inventurnummer
            name   char(80),             # Bezeichnung
            person int                   # Besitzer
        )
EOT
```

Sfortunatamente, questo script accetterà nuovi elementi con identico numero di personale; il modificatore msql-1 `primary key` intende prevenire esattamente questo comportamento. La documentazione msql-2 mostra come usare la clausola `CREATE INDEX` per ottenere elementi univoci.

5 Conclusione e Riepilogo

Se si è installato msql-2 sul proprio sistema si può dare un'occhiata ai programmi d'esempio scritti in Lite, il linguaggio di scripting proprietario di msql-2.

Entrambe le versioni di msql sono fornite con un insieme base di strumenti di amministrazione che permettono all'utente di creare e cancellare tabelle (`msqladmin`) e di esaminare le strutture di database (`relshow`).

La seconda generazione di msql (cioè msql-2) presenta qualche strumento in più: `msqlimport` e `msqlexport`. Questi permettono l'esportazione e l'importazione di dati sotto forma di file di testo da e verso database SQL. Possono essere usati per il caricamento di quantità di dati esistenti *d'un colpo* in tabelle già create, o l'estrazione di dati da tabelle, in modo che l'utente non debba scrivere *alcuna* linea di perl o SQL o qualsiasi altro codice per questi compiti.

Se si desidera scrivere propri script perl per l'interazione con database si troverà sufficiente supporto nei file di esempio e nella estensiva documentazione in linea fornita col modulo DBI.

Ad ogni modo, si è ora pronti per cominciare e presentare i propri dati ad utenti della propria rete, o anche del Web.