Wrapper VB6 - MySQL senza Driver ODBC

CONSIDERAZIONI INTRODUTTIVE

(Le informazioni sull'uso della demo sono in fondo a questo documento)

Com'è noto non esistono driver specifici che possano essere installati in VB6 allo scopo di supportare pienamente MySql.

L'unico driver utilizzabile è l'ODBC che presenta problemi di estrema lentezza in quanto *"...L'accesso vero e proprio al database è eseguito tramite Connector/ODBC, in cui, tra ADO e Connector /ODBC, sono presenti OLE-DB ed il driver ODBC per OLE-DB. La connessione avviene secondo questo schema: VB/VBA -> ADO ->OLE-DB -> driver per OLEDB -> Connector/ODBC -> MySQL. Se MySQL è attivo su un computer come programma[n.d.r. – o servizio] Visual Basic, le prime cinque fasi di questa catena di comunicazione avvengono sul sistema client. E' Connector/ODBC che comunica via rete con MySQL server*" (cfr. MySQL 5 - Guida completa - Michael Kofler - Apogeo 2006 - pag. 575).

Ciò che viene presentato in questo documento, è la soluzione ai problemi di cui sopra, in quanto viene illustrata una classe che accede direttamente a MySQL, da VB6, saltando tutte le fasi precedentemente indicate, non facendo uso di ODBC. ***Il risultato è un accesso velocissimo a MySql*** (vedere il test di velocità in fondo al presente documento).

La classe in questione, che in fase di impiego operativo, sarà bene che venga convertita in una opportuna DLL, si basa su un meraviglioso lavoro di **Jim Banasiak**, costruita oltre diciassette anni or sono(!), che per quanto si legge sulla rete(e per oscuri motivi), non è ben conosciuta dalla comunità degli sviluppatori in VB, i quali continuano a lamentarsi della mancanza di ausili validi al collegamento VB6/MySQL.

Il motivo del rinnovato interesse degli sviluppatori per VB ed i suoi annessi, risiede, a parere di chi scrive - un pò come sta avvenendo nel mondo della musica con il risorgere del vinile a scapito del digitale - per i seguenti motivi:

1. I linguaggi .NET sono estremamente complicati, lenti e con migliaia di dipendenze.
2. I rilasci di nuove versioni avvengono ormai con cadenza quasi settimanale(!), disorientando gli sviluppatori, e rendendo l'obsolescenza in tema di competenze degli stessi a dir poco drammatica.
3. Costringe gli utenti a dismettere tutto il patrimonio informatico costruito nel tempo, in quanto il mondo COM e il mondo .NET hanno ben poche linee di sovrapposizione.
4. I linguaggi .NET non producono eseguibili binari, per cui sono facilmente attaccabili (oltre che lenti, come detto) in tutte le loro fasi.
5. I committenti, in genere, non conoscono le tecnologie con cui sono sviluppati i prodotti che richiedono. A loro interessa che funzionino senza problemi.

Alle considerazioni indicate, se ne potrebbero aggiungere altre; ma quelle citate sono più che sufficienti per giustificare l'interesse per gli argomenti qui trattati.

Tornando alla classe che viene presentata, chi scrive ha ritenuto di apportare solo alcune integrazioni che ritiene utili, mentre la classe medesima è stata lasciata inalterata nella sua versione originale (salvo la non esposizione di alcuni metodi non particolarmente utili).

Detta classe, si basa su un insieme piuttosto corposo di API MySQL, racchiuse in una libreria che ha nome **"libmysql.dll"** (diversa dalla omonima libreria che viene installata da MySQL), che ha fondamentalmente il compito di stabilire una connessione con MySQL, restituendo un recordset pienamente compatibile con **ADO.COM**, per cui, nella pratica operativa, le operazioni sui database saranno del tutto trasparenti allo sviluppatore.

Inoltre, i recordset restituiti dai metodi possono essere tranquillamente assegnati ai controlli grafici per la gestione dei dati presenti in ADO.COM, come le griglie e gli altri controlli, senza alcun intervento o aggiustamento.

Il limite di questa libreria, essendo stata scritta agli albori di MySQL (circa nell'anno 2000), è rappresentato dal fatto che supporta solamente la codifica password a 16 caratteri e non quella attuale a 40(41) caratteri, per cui è pensabile che in futuro essa diventi incompatibile con gli sviluppi di MySQL, atteso che Oracle ha annunciato che le prossime versioni di MySql potrebbero non supportare più la codifica a16 caratteri (come sta avvenendo). Il problema, però, potrebbe non interessare i cloni di MySql, che stanno venendo alla luce in questi ultimi anni.

A meno di un aggiornamento della libreria (auspicabile, ma difficile), si potrebbe continuare ad utilizzare la classe, collegandosi soltanto con il nome utente, lasciando vuota la password. Ma anche questo escamotage potrebbe rivelarsi infruttuoso.

Al momento, però, non esiste questo problema, a patto di seguire le istruzioni sotto indicate.

INSTALLAZIONE DI MYSQL

Sebbene l'istallazione dell'ambiente sia cosa nota, è necessario sviluppare le considerazioni che seguono per adeguare MySQL alle condizioni operative imposte dalla libreria "libmysql.dll".

La classe è stata testata con successo sulla versione 5.1 di MySQL, che risponde positivamente senza interventi di alcun genere, e sul clone MariaDb 10.2, (ultima versione disponibile agosto 2017) utilizzando gli accorgimenti che verranno indicati. Inoltre è stata testata su MySQL 5.5.58. Gli adeguamenti riguardano tutti la codifica a 16 caratteri della password, di cui si è detto.

Di seguito verranno esposte sommariamente le modalità di collegamento di VB6 ai server testati.

MYSQL ver. 5.1

1. Installare MySQL **senza password** per l'utente 'root'.
2. Una volta installato l'ambiente, non è richiesto null'altro.
3. Entrare in MySql ed assegnare una password sicura all'utente "root", con il comando: **SET PASSWORD FOR 'root'@'localhost'=OLD_PASSWORD('nuova_password')**

In tal modo si avrà la possibilità di amministrare il server, in modo sicuro, anche da VB6, entrando con user = **root** e password=**nuova_password**

MARIADB ver. 10.2

1. Installare MySQL **senza password** per l'utente 'root'
2. Raggiungere **my.ini** (in ..\MariaDB 10.2\Data) ed aggiungere *in fondo alla sezione* **[mysqld]**, quanto segue:

secure_auth=0
old_passwords=1

Dalla linea di comando, o utility di gestione, spegnere il server (Shutdown), e successivamente riavviare il servizio (net start MySql), oppure riavviare il computer.

Ripetere quanto fatto per MySql 5.1 al punto 3. per assegnare una password sicura, a 16 caratteri all'amministratore, per gestire MySql da VB6.

MySQL ver. 5.5.58

MySQL 5.5.20 ha un comportamento ancora diverso.

Pur installando l'utente "root" senza password, non accetta l'autenticazione da VB6 con il solo nome utente.

Per ovviare al problema, comportarsi come segue:

1. Raggiungere il file **my.ini** in [..\MySQl\MySql Server 5.5] ed aggiungere *in fondo alla sezione server* **[mysqld]** quanto segue:
   secure_auth=0
   old_passwords=1

2. Salvare e riavviare il computer.
3. Entrare con "root" , dalla riga di comando o utility amministrativa, e creare un **nuovo amministratore** con password a 16 caratteri:
   GRANT ALL PRIVILEGES ON *.* to 'new_root'@'localhost' IDENTIFIED BY 'new_pwd' WITH GRANT OPTION

In tal modo, è possibile accedere come amministratore da VB6.

MySQL versioni successive alla 5.5.58

Non sono state testate versioni successive alla 5.5.58. Tuttavia Oracle ha deciso di disabilitare l'accesso alla variabile di sistema "old_passwords" (sicuramente dalla versione 5.7.xx), per cui non sono più permesse password con hashing a 16.

CREAZIONE DEGLI UTENTI ORDINARI

Per la creazione degli utenti ordinari (non amministratori) è possibile utilizzare la clausola "IDENTIFIED BY", oppure ricorrere alla creazione per fasi, come segue, come già indicato precedentemente:

Metodo "breve" (si crea l'utente con password, e si assegnano i privilegi sul Db):

GRANT ALL PRIVILEGES ON Db.* TO 'user'@'localhost' IDENTIFIED BY 'password'

Metodo "lungo", per fasi:

1. **CREATE USER 'nome'@'server'**

2. **SET PASSWORD FOR 'nome'@'server' =OLD_PASSWORD('password')**

3. GRANT ALL ON Db.* TO 'nome'@'server'

Con i comandi sopra indicati, gli utenti avranno le loro password codificate a 16.

E' sempre opportuno chiudere le catene di comandi con FLUSH PRIVILEGES, allo scopo di rendere effettive le modifiche senza attendere il riavvio del server.

In definitiva, giova rimarcare ancora una volta, che, **indipendentemente dalle potenzialità del server in uso** ,con la "**libmysql.dll**", qui usata, potranno essere gestiti solo accessi codificati **a16 caratteri**

LA CLASSE

A seguire, verranno illustrati i metodi e le proprietà esposte dalla classe, con una breve indicazione circa il loro utilizzo.

Tutti i metodi (tranne due) e tutte le proprietà (tranne una) sono accreditabili a Jim Banasiak. Gli altri, a chi scrive.

AffectRows

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna il numero delle righe interessate dalla query

Uso: ***long = <class>.AffectRows***

ClientFlags

Tipo: Proprietà

Autore: Jim Banasiak

Descrizione: Non documentata

Uso: ***long = <class>.ClientFlags (Get/Let)***

Connect

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Apre la connessione a MySQL

Uso: **Call <class>.Connect(Optional ByVal Host as String, Optional ByVal User as String, Optional ByValPassword as String])**

CreateDb

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Crea il database Dbname

Uso: : ***Call <class>.CreateDb(Dbname as String)***

DB

Tipo: Proprietà

Autore: Jim Banasiak

Descrizione: Ritorna/Assegna il nome del database corrente

Uso: ***String = <class>.DB (Get/Let)***

DropDb

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Elimina il database indicato da Dbname

Uso: ***Call <class>.DropDb(Dbname as String)***

ErrDescription

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna al chiamante la descrizione dell'errore di MySQL

Uso: ***String = <class>.ErrDescription***

ErrNumber

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna al chiamante il numero dell'errore di MySQL

Uso: ***Long = <class>.ErrNumber***

EscapeString

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Consente di inserire caratteri di escape (es. \) in una stringa da utilizzare in un'istruzione SQL tenendo conto del set di caratteri corrente della connessione. Funzione

deprecata e non usata

Uso: ***Call <calss>.EscapeString(str1 as String, str2 as String, <lunghezza_str2> as Long)***

GetClientInfo

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna un numero di versione che indentifica il client

Uso: ***String = <class>.GetClientInfo***

GetHostInfo

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna il nome del server ed il protocollo usato (es. "localhost via TCP/IP")

Uso: ***String = <class>.GetHostInfo***

GetServerInfo

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna la versione del server

Uso: ***String = <class>.GetServerInfo***

Host

Tipo: *Proprietà*

Autore: Jim Banasiak

Descrizione: Ritorna il nome del server corrente

Uso: ***string =<class>.Host (Get/Set)***

InsertID

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna l'ultimo indice (Colonna ID) con tipo ***Auto_Increment*** generato

Uso: ***Long = <class>.InsertID***

Kill

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Elimina un determinato thread

Uso: ***Call <class>.Kill(Long as <indice thread>)***

ListDbs

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Elenca i databases presenti in MySQL in base ad una espressione regolare

Uso: ***recordset = <class>.ListDbs(Optional str as String ="%")***

ListFields

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna il nome dei campi di una tabella

Uso: ***recordset = <class>.ListFields(Optional table as String, Optional str as String ="%")***

ListProcesses

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna una lista dei processi (threads) in corso

Uso: ***recordset = <class>.ListProcesses***

ListTables

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna la lista delle tabelle del database corrente

Uso: ***recordset = <class>.ListTables(Optional str as String="%")***

NumFields

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna il numero delle colonne interessate dall'ultima query

Uso: Long = <class>.NumFields(Optional str as String="%")

Nota - Ad evitare l'errore di VB6 "necessario oggetto", passare sempre il carattere wild "%"

NumRows

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna il numero di righe che hanno interessato l'ultima query

Uso: ***Long = <class>.NumRows***

Password

Tipo: Proprietà

Autore: Jim Banasiak

Descrizione: Valore password

Uso: ***String = <class>.Password="mia_password"*** *(**Get/Let**)*

Ping

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Indica se il server è in attività o meno

Uso: ***Non ritorna nulla. Se all'interno il valore di <ret> dell'API di MySQL è diverso da 0, solleva un'eccezione. In effetti dovrebbe controllare che la connessione sia attiva, ed eventualmente ripristinarla. Ma non sembra che il metodo sia implementato***

Port

Tipo: Proprietà

Autore: Jim Banasiak

Descrizione: Assegna/legge la porta di comunicazione di MySQL. Usualmente: 3306

Uso: ***Long = <class>.Port (Get/Let)***

Query

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Esegue una query determinata e terminata con un carattere NULL

La query può essere di qualunque tipo (SELECT,INSERT,UPDATE E DELETE), ma è responsabilità dello sviluppatore la costruzione (corpo e dati inclusi) della medesima.

Es. "Select * from tabella where Campo1='Gennaio'" oppure

"Insert into tabella(Campo1) values('Febbraio') " ecc.ecc.

Inoltre il metodo può essere utilizzato, oltre che per l'invio di query al database, per inviare comandi di qualsivoglia natura a MySql.

Può essere quindi usato per creare utenti, database, tabelle, gestire transazioni ed inviare comandi di altro genere a MySQL.

Uso: ***recordset = <class>.Query(<testo_query_letterale as String)***

QueryParam

Tipo: Metodo

Autore: Lino Aiello

Descrizione: Si tratta del medesimo metodo [Query] che figura nella classe originale, terminata con un carattere NULL, ma che accetta, nella sua firma, un secondo parametro opzionale costituito da un vettore variant che contiene i dati su cui opera la query.

Si tratta della costruzione di query **DINAMICHE**.

Nel caso di assenza del parametro opzionale, la struttura ed il comportamento del metodo è del tutto identico a quello della funzione [Query]; le cose cambiano radicalmente qualora sia presente il vettore variant.

Il vettore variant opera sostituendo, con modalità **UNO ad UNO**, i segnaposto contenuti nel testo dell'espressione SQL della query da eseguire, con i propri valori.

Nel corpo dell'espressione SQL i segnaposto sono rappresentati dal carattere **"?".**

Gli esempi a seguire, chiariranno meglio i termini del problema.

Esempio 1)

Si voglia eseguire la query seguente (letterale) così come la richiederebbe il metodo Query:

"Insert into tablella (Nome,Data) values('Cavour','1830-07-15')

con il vettore variant diventa:

.....

Dim Dati(1) as variant

```
Dim strSQL as string
    Dati(0)="Cavour"
    Dati(1)=CDate("15/07/1830")
    strSQL="Insert into tabella(Nome,Data) values(?,?)
    Call <class>.QueryParam(strSQL,Dati())
```

....

Esempio 2)

Si voglia eseguire la query seguente (letterale) così come la richiederebbe il metodo Query:

"Update Tabella set Cognome='Mazzini',Data='1855-02-03',Valore=1234.56 where Citta='Milano' and Nome='Giuseppe'

Diventa:

....

```
Dim Dati(4) as variant
Dim strSQL as  string
    Dati(0)="Mazzini"
    Dati(1)=CDate("03/02/1855")
    Dati(2)=1234.56
    Dati(3)="Milano"
    Dati(4)="Giuseppe"
    strSQL="Update tabella set Cognome=?,Data=?,Valore=? where Citta=? and Nome=?"
    Call <class>.QueryParam(strSQL,Dati())
```

....

Esempio 3)

Si voglia eseguire la query seguente (letterale) così come la richiederebbe il metodo Query:

"Delete from tabella where Cognome='Garibaldi'

Diventa:

...

```
Dim strSQL as string
Dim Dati(0) as variant
        Dati(0)="Garibaldi"
        strSQL="Delete from tabella where Cognome=? "
        Call <class>.QueryParam(strSQL,Dati())
```

....

Esempio 4)

Si voglia eseguire la query seguente (letterale) così come la richiederebbe il metodo Query:

> "Select Cognome,Nome,Valore from tabella where Citta='Milano' and Data=’2018-01-10’”

Diventa:

...

```
Dim strSQL as string
Dim Dati(1) as variant
    Dati(0)="Milano"
    Dati(1)=CDate(“10/01/2018”)
     strSQL="Select Cognome,Nome,Data from tabella where Citta=? and Valore=?"
    set recordset = <class>.QueryParam(strSQL,Dati())
```

....

In questo caso, in presenza di una clausola SELECT, il valore di ritorno è significativo, essendo un recordset scorrevole che può essere *assegnato, per esempio, ad un controllo di VB6.*

In conclusione, il vettore dati ha indice superiore pari a [n-1] parametri, e può ricevere qualunque dato che interessi la query (variabili di programma, textbox o altri controlli, valori di ritorno di funzioni, ecc.ecc.) e può avere, naturalmente, qualunque nome.

In aggiunta**, non è necessario rispettare il formato dei dati di MySql** che vengono passati a [QueryParam], in quanto il metodo si occupa dell’adeguamento delle variabili in modo trasparente per lo sviluppatore.

Appare quindi evidente, che le query necessarie al software in sviluppo, possono essere create a priori in maniera molto semplice senza ricorrere a fastidiose costruzioni e concatenazioni di stringhe (con annesse formattazioni!) , in quanto i valori che devono interpretare, in genere di volta in volta diversi, sono passati in maniera dinamica in fase di runtime.

Inoltre il presente metodo dispone di una **espressione regolare** diretta a raddoppiare gli apici (apostrofi) che dovessero essere presenti nel corpo della stringa SQL e/o nelle variabili assegnate dal vettore, senza che di tale problema, ancora una volta, debba occuparsene lo sviluppatore.

Allo scopo, per il buon funzionamento dell'espressione regolare, è necessario che l'apice da raddoppiare sia preceduto e seguito da un carattere stampabile.

(es. "D'Annunzio" e non "D 'Annunzio o "D' Annunzio").

Uso: ***set recordset=<class>.QueryParam(strSQL as String, Optional ByVal Dati as Variant)***

RealConnect

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Come Connect, ma con un numero maggiore di parametri. In particolare, permette di scegliere il Database e la porta di comunicazione in sede di connessione

Uso: ***<class>.RealConnect(Optional host As String = vbNullString, Optional user As String = vbNullString, Optional Passwd As String = vbNullString, Optional DB As String = vbNullString, Optional Port As Long = MYSQL_PORT, Optional Unix_Socket As String = vbNullString, Optional clientflag As Long= 0)***

RealQuery

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Identica a [Query], ma non terminata con il carattere NULL; richiede la lunghezza della query

Uso: ***recordset = <class>.RealQuery(strSQL as String, <lunghezza> as Long)***

RealQueryParam

Tipo: Metodo

Autore: Lino Aiello

Descrizione: Come QueryParam nel funzionamento. Richiede lunghezza della stringa. Mantenuta per corrispondenza nella classe

Uso: ***recordset = <class>.RealQueryParam(strSQL As String, <lunghezza> As Long, Optional ByVal Dati As Variant)***

Refresh

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Metodo non documentato

Uso: ***Call <class>.Refresh(Enum refresh option)***

SelectDb

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Permette di selezionare il database su cui operare

Uso: ***Call <class>.SelectDb(ByVal Dbname as String)***

ShowQuery

Tipo: Proprietà

Autore: Lino Aiello

Descrizione: Restituisce il corpo della stringa SQL espansa ed elaborata da [QueryParam]. In scrittura all'interno della classe; in lettura per il chiamante

Uso: ***String = <class>.ShowQuery() (Get/Let)***

ShutDown

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Spegne il server

Uso: ***Call <class>.ShutDown()***

Stat

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna lo stato del server in una stringa

Uso: ***String = <class>.Stat()***

ThreadID

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Ritorna l'indice del thread corrente

Uso: ***Long = <class>.ThreadID***

ThreadSafe

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Restituisce 1 se i clients sono compilati come thread-safe.

Uso: ***Long = <class>.ThreadSafe***

UnixSocket

Tipo: Metodo

Autore: Jim Banasiak

Descrizione: Utilizzato in RealConnect (Non usare in Windows)

Uso: ***Vedere la firma del metodo RealConnect***

User

Tipo: Proprietà

Autore: Jim Banasiak

Descrizione: Ritorna/assegna il nome dell'User

Uso: ***String = <class>.User (Get/Let)***

Pacchetto Allegato

Il pacchetto allegato, oltre al progetto dimostrativo, contiene i sorgenti della classe ed il codice sorgente della classe originaria di Jim Banasiak.

Nella esecuzione della demo e nelle proprie applicazioni, ricordarsi di aprire i seguenti riferimenti, se non già presenti:

- Microsoft ActiveX Data Object 2.7 Library o superiore.
- Microsoft VBScript Regular Expressions 5.5.
- La libreria "libmysql.dll" deve essere collocata nella stessa directory degli eseguibili (non va registrata).

Se si entra come amministratore, premendo il pulsante "*Connect*", senza indicare il database, bisognerà scegliere il database (es. mysql) dalla linguetta 1 del controllo tab. Altrimenti ciò non è necessario.

Una volta entrati in MySql, nella linguetta 2 del controllo tab, è possibile inviare tutti i tipi di comandi che si desidera (creazione utenti, database, tabelle, query letterali ecc.)

La linguetta 3 è invece riservata alle query parametriche.

Dopo aver approntato le query con gli schemi proposti (creare prima un database con una tabella!), aggiungere tramite il bottone "*Carica lista*" i valori da assegnare alla query(in ordine uno a uno), tenendo presente che sono possibili, per brevità espositiva, solo valori di tipo varchar,datetime,double ed int. Al termine, premere il bottone corrispondente alla query voluta.

I dati, sempre per comodità, devono essere immessi separando il valore dal tipo mediante un **";"**

es. **Rossi;varchar**

Mario;varchar

20/05/1960;datetime

1543,77;double

100;int

Test di velocità effettuati con la classe VB6:

Macchina: Win7/32 - Processore I5-3470 3,20 GHz - Rete WiFi

Tabella di 12 campi così composta :

N. 3 campi varchar

N. 3 campi datetime

N .3 campi double

N. 3 campi int

MYSQL 5.1.win32

Scrittura > 165 records/sec. (localhost/rete)

Lettura di 2.000 records = < 1 sec. (localhost/rete)

MARIADB 10.2.win32

Scrittura > 165 records/sec. (localhost/rete)

Lettura di 2.000 records = < 1 sec. (localhost/rete)

MYSQL 5.5.58.win32

Scrittura > 180 records/sec. (locahost/rete)

Lettura di 2.000 records = < 1 sec. (localhost/rete)

Macchina: Win7/64 - Processore I5-3470 3,20 GHz – 8Gb - Rete WiFi

MYSQL 5.1.win32

Scrittura > 650(!!) records/sec. (localhost/rete)

Lettura di 2.000 records = 0 sec. (localhost/rete)

MARIADB 10.2.win32

Scrittura > 660(!!) records/sec. (localhost/rete)

Lettura di 2.000 records = 0 sec. (localhost/rete)

MYSQL 5.5.58.win32

Scrittura > 660(!!) records/sec. (locahost/rete)

Lettura di 2.000 records = 0 sec. (localhost/rete)

Lino Aiello (linuscontest@gmail.com)

Pescara, gennaio 2018