MySQL 安装与初始设置

1、 FreeBSD 安装 MySQL

略。

2、 Windows 安装 MySQL

安装后执行

X:\mysql\bin> mysqld-nt –install

如果出现 Service successfully installed 的提示，这表示你已成功的将 MySQL 安装成一项 Windows 的服务。登录 MySQL 数据库

启动可以在服务窗口操作，或者 net start MySQL

3、 登录

MySQL 安装后默认用户为 root，默认密码为空

格式：mysql -h host -u user –p

例如：mysql –u root –p

4、 修改密码

格式：mysqladmin －u 用户名 -p 旧密码 password 新密码

例如：给 root 加个密码 ab12。键入命令：mysqladmin –u root password ab12

5、建立数据库

格式：create database 库名;

例如：建立新数据库 shopex

mysql> create database shopex;

6、建立新用户

格式：grant all privileges on 数据库.* to 用户名@登录主机 identified by "密码";

例如： 增加一个用户 test 密码为 1234,让他只可以在 localhost 上登录，并可以对数据库 Shopex 进行所有的操作（localhost 指本地主机，即 MYSQL 数据库所在的那台主机），

mysql> grant all privileges on shopex.* to test@localhost identified by "1234";

通过以上操作，你建立一个新的数据库 shopex，并增加了一个名为 test 对 shopex 数据库有所有操作权限。

7、显示数据库

格式：show databases;

8、修改列名

格式：ALTER table_name CHANGE old_col_name new_col_name [type];

例如：将 agr 改为 age，类型为 int

Mysql> alter table children change agr age int;

MySQL C API 使用简要

1、 mysql_real_connect()

原型：MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag)

描述：mysql_real_connect()尝试与运行在主机上的 MySQL 数据库引擎建立连接。在能够执行需要有效 MySQL 连接句柄结构的任何其他 API 函数之前，mysql_real_connect()必须成功完成。

返回值：如果连接成功，返回 MYSQL*连接句柄。如果连接失败，返回 NULL。对于成功的连接，返回值与第 1 个参数的值相同。

注意：一般来说在使用 mysql_real_connect()之前需要先使用 mysql_init()初始化 MYSQL 对象。

例如：

```
MYSQL my_connection;
mysql_init(&my_connection);
if (mysql_real_connect(&my_connection, "localhost", "root", "", "fm", 0, NULL,
CLIENT_FOUND_ROWS)) {
        printf("Connection Success\n");
        ……
}
```

2、mysql_close()

原型：void mysql_close(MYSQL *mysql)

描述：关闭前面打开的连接。如果句柄是由 mysql_init()或 mysql_connect()自动分配的，mysql_close()还将解除分配由 mysql 指向的连接句柄。

返回值：无。

例如：

```
mysql_close(&my_connection);
```

3、mysql_errno()

原型：unsigned int mysql_errno(MYSQL *mysql)

描述：对于由 mysql 指定的连接，mysql_errno()返回最近调用的 API 函数的错误代码，该函数调用可能成功也可能失败。“0”返回值表示未出现错误。在 MySQL errmsg.h 头文件中，列出了客户端错误消息编号。

*注意：*如果成功，某些函数，如 mysql_fetch_row()等，不会设置 mysql_errno()。经验规则是，如果成功，*所有向服务器请求信息的函数*均会复位 mysql_errno()。

返回值：如果失败，返回上次 mysql_xxx()调用的错误代码。“0”表示未出现错误。

例如：

```
Fprintf(stderr, "ERROR Num %d:", mysql_errno(&my_connection));
```

4 、mysql_error()

原型：const char *mysql_error(MYSQL *mysql)

描述：对于由 mysql 指定的连接，对于失败的最近调用的 API 函数，mysql_error()返回包含错误消

息的、由 Null 终结的字符串。如果该函数未失败，mysql_error()的返回值可能是以前的错误，或指明无错误的空字符串。

经验规则是，如果成功，所有向服务器请求信息的函数均会复位 mysql_error()。

返回值：返回描述错误的、由 Null 终结的字符串。如果未出现错误，返回空字符串。

例如：

```
Fprintf(stderr, "ERROR: %s\n", mysql_error(&my_connection));
```

5、 mysql_query()

原型：int mysql_query(MYSQL *mysql, const char *query)

描述：执行由“Null 终结的字符串”查询指向的 SQL 查询。正常情况下，字符串必须包含 1 条 SQL 语句，而且不应为语句添加终结分号（ ‘;’ ）或“\g”。如果允许多语句执行，字符串可包含多条由分号隔开的语句。

mysql_query()不能用于包含二进制数据的查询，应使用 mysql_real_query()取而代之（二进制数据可能包含字符‘\0’，mysql_query()会将该字符解释为查询字符串结束）。

如果希望了解查询是否应返回结果集，可使用 mysql_field_count()进行检查。

返回值：如果查询成功，返回 0。如果出现错误，返回非 0 值。

例如：

```
int res;
res = mysql_query(&my_connection, "insert into children values(7, 'Annd', 5)");
if (!res)
    printf("Inserted %lu rows\n", (unsigned long)mysql_affected_rows(&my_connection));
else
    fprintf(stderr, "Insert ERROR %d: %s\n", mysql_errno, (&my_connection), mysql_error(&my_connection));
```

6、mysql_store_result()

原型：MYSQL_RES *mysql_store_result(MYSQL *mysql)

描述：对于成功检索了数据的每个查询（SELECT、SHOW、DESCRIBE、EXPLAIN、CHECK TABLE 等），必须调用 mysql_store_result()或 mysql_use_result() 。对于其他查询，不需要调用 mysql_store_result()或 mysql_use_result()，但是如果在任何情况下均调用了 mysql_store_result()，它也不会导致任何伤害或性能降低。通过检查 mysql_store_result()是否返回 0，可检测查询是否没有结果集（以后会更多）。

如果希望了解查询是否应返回结果集，可使用 mysql_field_count()进行检查。

mysql_store_result()将查询的全部结果读取到客户端，分配 1 个 MYSQL_RES 结构，并将结果置于该结构中。

如果查询未返回结果集，mysql_store_result()将返回 Null 指针（例如，如果查询是 INSERT 语句）。

如果读取结果集失败，mysql_store_result()还会返回 Null 指针。通过检查 mysql_error()是否返回非空字符串，mysql_errno()是否返回非 0 值，或 mysql_field_count()是否返回 0，可以检查是否出现了错误。

如果未返回行，将返回空的结果集。（空结果集设置不同于作为返回值的空指针）。

一旦调用了 mysql_store_result()并获得了不是 Null 指针的结果，可调用 mysql_num_rows()来找出结果集中的行数。

可以调用 mysql_fetch_row()来获取结果集中的行，或调用 mysql_row_seek()和 mysql_row_tell()来获取或设置结果集中的当前行位置。一旦完成了对结果集的操作，必须调用 mysql_free_result()。

返回值：具有多个结果的 MYSQL_RES 结果集合。如果出现错误，返回 NULL。

7、mysql_num_rows()

原型：my_ulonglong mysql_num_rows(MYSQL_RES *result)

描述：返回结果集中的行数。

mysql_num_rows()的使用取决于是否采用了 mysql_store_result()或 mysql_use_result()来返回结果集。如果使用了 mysql_store_result()，可以立刻调用 mysql_num_rows()。如果使用了 mysql_use_result()，mysql_num_rows()不返回正确的值，直至检索了结果集中的所有行为止。

返回值：结果集中的行数。

8、mysql_fetch_row()

原型：MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)

描述：返回查询结果中的行。

检索结果集的下一行。在 mysql_store_result()之后使用时，如果没有要检索的行，mysql_fetch_row()返回 NULL。在 mysql_use_result()之后使用时，如果没有要检索的行或出现了错误，mysql_fetch_row()返回 NULL。

行内值的数目由 mysql_num_fields(result)给出。如果行中保存了调用 mysql_fetch_row()返回的值，将按照 row[0]到 row[mysql_num_fields(result)-1]，访问这些值的指针。行中的 NULL 值由 NULL 指针指明。

可以通过调用 mysql_fetch_lengths()来获得行中字段值的长度。对于空字段以及包含 NULL 的字段，长度为 0。通过检查字段值的指针，能够区分它们。如果指针为 NULL，字段为 NULL，否则字段为空。

返回值：下一行的 MYSQL_ROW 结构。如果没有更多要检索的行或出现了错误，返回 NULL。

9、mysql_fetch_field()

原型：MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)

描述：返回列名。

采用 MYSQL_FIELD 结构的结果集的列。重复调用该函数，以检索关于结果集中所有列的信息。未剩余字段时，mysql_fetch_field()返回 NULL。

每次执行新的 SELECT 查询时，将复位 mysql_fetch_field()，以返回关于第 1 个字段的信息。调用 mysql_field_seek()也会影响 mysql_fetch_field()返回的字段。

如果调用了 mysql_query()以在表上执行 SELECT，但未调用 mysql_store_result()，如果调用了 mysql_fetch_field()以请求 BLOB 字段的长度，MySQL 将返回默认的 Blob 长度（8KB）。之所以选择 8KB 是因为 MySQL 不知道 BLOB 的最大长度。应在日后使其成为可配置的。一旦检索了结果集，field->max_length 将包含特定查询中该列的最大值的长度。

返回值：当前列的 MYSQL_FIELD 结构。如果未剩余任何列，返回 NULL。

例如：

```
MYSQL_FIELD *field;
while((field = mysql_fetch_field(result))) {
```

```
    printf("field name %s\n", field->name);
}
```

MySQL C API 简单示例

首先需要确认 mysql.h 文件所在目录，如/usr/local/include/mysql/mysql.h。

创建数据库：

```
mysql>create database fm;
mysql>use fm;
mysql>create table childrencreate table children(childno int not null unique,fname varchar(20),age int);
mysql>insert into children values(5,"花儿",10）;
……
```

编译方法：

例如：

```
cc –o insert insert.c `mysql_config –cflags –libs`
```

例 1 插入数据

使用 MySQL API：

mysql_init();	获取或初始化 MYSQL 结构。
mysql_real_connect();	连接到 MySQL 服务器。
mysql_query();	执行指定为“以 Null 终结的字符串”的 SQL 查询。
mysql_affected_rows();	返回上次 UPDATE、DELETE 或 INSERT 查询更改 / 删除 / 插入的行数。
mysql_errno();	返回上次调用的 MySQL 函数的错误编号。
mysql_error();	返回上次调用的 MySQL 函数的错误信息。
mysql_close();	关闭服务器连接。

```
/* insert.c */
#include <stdio.h>
#include <stdlib.h>
#include "/usr/local/include/mysql/mysql.h"

int
main(int argc, char *argv[])
{
       MYSQL my_connection;
       int res;
       mysql_init(&my_connection);

       if (mysql_real_connect(&my_connection, "localhost", "root", "", "fm", 0, NULL, CLIENT_FOUND_ROWS)) {
              printf("Connection Success\n");
              res = mysql_query(&my_connection, "insert into children values(7,
```

```
'Annd', 5)");

              if (!res)
                     printf("Inserted %lu rows\n", (unsigned
long)mysql_affected_rows(&my_connection));
              else
                     fprintf(stderr, "Insert ERROR %d: %s\n", mysql_errno,
(&my_connection), mysql_error(&my_connection));

              mysql_close(&my_connection);
       } else {
              fprintf(stderr, "Connection Failed\n");
              if (mysql_errno(&my_connection)) {
                     fprintf(stderr, "Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_errno(&my_connection));
              }
       }

       return EXIT_SUCCESS;
}
```

例 2 修改数据库中数据

将 例 1

```
res = mysql_query(&my_connection, "insert into children values(7, 'Annd', 5)");
```

改为

```
res = mysql_query(&my_connection, "update children set age=21 where childno<2");
```

例 3 查询数据，输出查询到的行数

使用 MySQL API ：

mysql_init();	获取或初始化 MYSQL 结构。
mysql_real_connect();	连接到 MySQL 服务器。
mysql_query();	执行指定为“以 Null 终结的字符串”的 SQL 查询。
mysql_affected_rows();	返回上次 UPDATE、DELETE 或 INSERT 查询更改 / 删除 / 插入的行数。
mysql_store_result();	检索完整的结果集至客户端。
mysql_fetch_row();	从结果集中获取下一行。
mysql_free_result();	释放结果集使用的内存。
mysql_errno();	返回上次调用的 MySQL 函数的错误编号。
mysql_error();	返回上次调用的 MySQL 函数的错误信息。
mysql_close();	关闭服务器连接。

```
/* select.c */
#include <stdio.h>
#include <stdlib.h>
#include "/usr/local/include/mysql/mysql.h"
```

```
int
main(int argc, char *argv[])
{
	MYSQL my_connection;
	MYSQL_RES *res_ptr;
	MYSQL_ROW sqlrow;

	int res;

	mysql_init(&my_connection);

	if (mysql_real_connect(&my_connection, "localhost", "root", "", "fm", 0,
NULL, CLIENT_FOUND_ROWS)) {
		printf("Connection Success\n");
		res = mysql_query(&my_connection, "SELECT * from children");

		if (res) {
			printf("SELECT error: %s\n", mysql_error(&my_connection));
		} else {
			res_ptr = mysql_store_result(&my_connection);

			if (res_ptr) {
				printf("Retrieved %lu Rows\n", (unsigned
long)mysql_num_rows(res_ptr));
				while((sqlrow = mysql_fetch_row(res_ptr))) {
					printf("Fetched data...\n");
				}

				if (mysql_errno(&my_connection)) {
					fprintf(stderr, "Retrive error: %s\n",
mysql_error(&my_connection));
				}
				mysql_free_result(res_ptr);
			}
			mysql_close(&my_connection);
		} else {
			fprintf(stderr, "Connection failed\n");
			if (mysql_errno(&my_connection))
				fprintf(stderr, "Connection error %d: %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
		}

	}
```

```
        return EXIT_SUCCESS;
}
```

例 4 查询数据，输出查询到的结构

mysql_init();	获取或初始化 MYSQL 结构。
mysql_real_connect();	连接到 MySQL 服务器。
mysql_query();	执行指定为“以 Null 终结的字符串”的 SQL 查询。
mysql_store_result();	检索完整的结果集至客户端。
mysql_fetch_field();	返回所有字段结构的数组。
mysql_fetch_row();	从结果集中获取下一行。
mysql_free_result();	释放结果集使用的内存。
mysql_close();	关闭服务器连接。

```
/* my1.c */
#include <stdio.h>
#include <stdlib.h>
#include "/usr/local/include/mysql/mysql.h"

int
main(void)
{
    MYSQL my_con;
    MYSQL_RES *my_res;
    MYSQL_FIELD *field;
    MYSQL_ROW sqlrow;
    int res;

    mysql_init(&my_con);
    if (mysql_real_connect(&my_con, "127.0.0.1", "root", "", "fm", 0, NULL,
CLIENT_FOUND_ROWS)) {
        if (!mysql_query(&my_con, "select * from children")) {
            my_res = mysql_store_result(&my_con);
            while ((field = mysql_fetch_field(my_res))) {
                printf("%s\t", field->name);
            }
            printf("\n");
            while((sqlrow = mysql_fetch_row(my_res))) {
                printf("%s\t%s\t%s\n", sqlrow[0], sqlrow[1], sqlrow[2]);
            }
            mysql_free_result(my_res);
            mysql_close(&my_con);
        } else {
            printf("SELECT error\n");
        }
```

```
    } else {
        fprintf(stderr, "Can not connect MySQL Server\n");
        exit(1);
    }
    return 0;

}
```

例 5 查询数据，输出查询到的结果

使用 MySQL API ：

mysql_init();	获取或初始化 MYSQL 结构。
mysql_real_connect();	连接到 MySQL 服务器。
mysql_query();	执行指定为“以 Null 终结的字符串”的 SQL 查询。
mysql_store_result();	检索完整的结果集至客户端。
mysql_num_rows();	返回结果集中的行数。
mysql_fetch_field();	返回所有字段结构的数组。
mysql_num_fields();	返回结果集中的列数。
mysql_fetch_row();	从结果集中获取下一行。
mysql_free_result();	释放结果集使用的内存。
mysql_errno();	返回上次调用的 MySQL 函数的错误编号。
mysql_error();	返回上次调用的 MySQL 函数的错误信息。
mysql_close();	关闭服务器连接。

```
/* select1.c */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "/usr/local/include/mysql/mysql.h"

int
main(int argc, char *argv[])
{
    MYSQL my_connection;
    MYSQL_RES *res_ptr;
    MYSQL_ROW sqlrow;
    MYSQL_FIELD  *fd;
    char aszflds[25][25];

    int res;
    int i, j, k;

    mysql_init(&my_connection);

    if (mysql_real_connect(&my_connection, "localhost", "root", "", "fm", 0,
NULL, CLIENT_FOUND_ROWS)) {
```

```
        printf("Connection Success\n");
        res = mysql_query(&my_connection, "SELECT * from children");

        if (res) {
            printf("SELECT error: %s\n", mysql_error(&my_connection));
        } else {
            res_ptr = mysql_store_result(&my_connection);

            if (res_ptr) {
                printf("Retrieved        %lu        Rows\n",        (unsigned
long)mysql_num_rows(res_ptr));
                for (i = 0; fd = mysql_fetch_field(res_ptr); i++)
                    strcpy(aszflds[i], fd->name);
                printf("these info:\n");
                j = mysql_num_fields(res_ptr);

                for (i = 0; i < j; i++)
                    printf("%s\t", aszflds[i]);
                printf("\n");

                while((sqlrow = mysql_fetch_row(res_ptr))) {
                    for (i = 0; i < j; i++)
                        printf("%s\t", sqlrow[i]);
                    printf("\n");
                }

                if (mysql_errno(&my_connection)) {
                    fprintf(stderr,        "Retrive       error:        %s\n",
mysql_error(&my_connection));
                }
            }
            mysql_free_result(res_ptr);
        }
        mysql_close(&my_connection);
    }else {
        fprintf(stderr, "Connection failed\n");
        if (mysql_errno(&my_connection))
            fprintf(stderr,      "Connection      error      %d:      %s\n",
mysql_errno(&my_connection), mysql_error(&my_connection));
    }
    return EXIT_SUCCESS;
}
```