首先修改数据库的 my.ini 配置文件，允许系统在客户端到服务器端传递大数据时分配更多扩展内存以进行处理。

```
[mysqldump]
quick
max_allowed_packet = 500M
```

创建数据表时，修改数据引擎，紧跟在建表语句后面：

```
ENGINE=MyISAM DEFAULT    实现更改数据库引擎的作用
```

写数据库 url 时，增加一项参数设置：

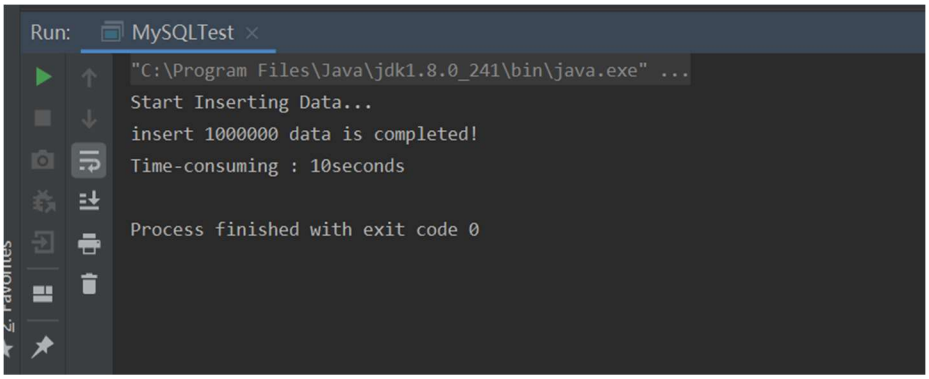
```
/url中的 rewriteBatchedStatements=true 参数，能够提高插入效率
```

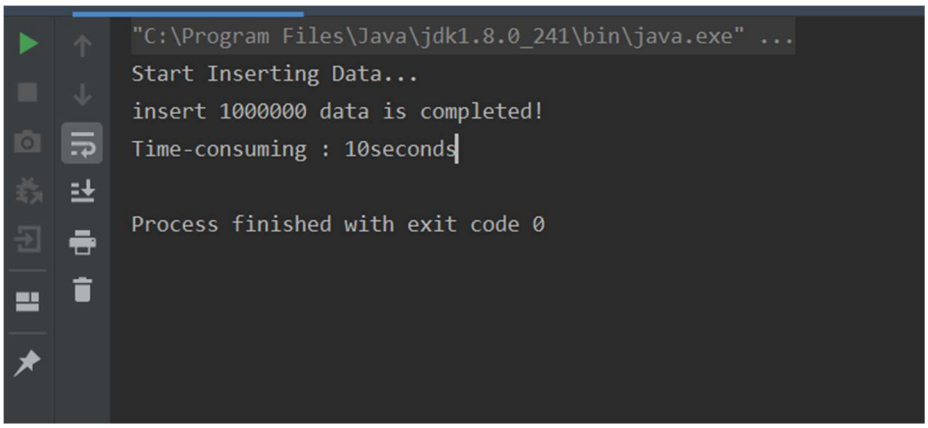
在运行 sql 语句进行数据插入时，采用运行动态 sql 语句的方法：

```
try {
    //获取数据库连接
    Connection conn = conn = DriverManager.getConnection(url, username, password);
    //实例化Statement
    //将sql语句中要插入的数据，用 ？ 来代替
    String sql = "INSERT INTO test1 (id, user_name, password, randomchar1, randomchar2, randomchar3, randomchar4, randomchar5) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

    conn.setAutoCommit(false);
    //执行动态sql语句
    PreparedStatement pstmt = conn.prepareStatement(sql);
    //对动态sql语句的 ？进行逐个赋值     有8个 ？所以，要对8个数据进行set（setInt 或 setString）
    for(int i = 0; i < insertNum; i++) {
        pstmt.setInt( parameterIndex: 1, i);
        pstmt.setString( parameterIndex: 2,randomStr( size: 8));
        pstmt.setString( parameterIndex: 3, randomStr( size: 8));
        pstmt.setString( parameterIndex: 4, randomStr( size: 8));
        pstmt.setString( parameterIndex: 5, randomStr( size: 8));
        pstmt.setString( parameterIndex: 6, randomStr( size: 8));
        pstmt.setString( parameterIndex: 7, randomStr( size: 8));
        pstmt.setString( parameterIndex: 8, randomStr( size: 8));
        pstmt.addBatch();
    }
    pstmt.addBatch();
    pstmt.executeBatch();
    //提交事务
    conn.commit();
```

运行结果测试：
测试数据表有八个字段，插入 1000000 条数据，多测测试，完成时间均在十秒内

```
Run:  MySQLTest
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Start Inserting Data...
insert 1000000 data is completed!
Time-consuming : 9seconds

Process finished with exit code 0
```





完成代码（包括数据表创建语句）：

```
package test;

import java.sql.*;

public class MySQLTest {

    //连接参数
    //url 中的 rewriteBatchedStatements=true 参数，能够提高插入效率
    public static String url =
"jdbc:mysql://localhost:3306/test1?useUnicode=true&rewriteBatchedStatements=true&char
acterEncoding=UTF8&useSSL=false&serverTimezone=UTC";
    public static String username = "root";
    public static String password = "iesapp";

    //注册驱动
    static{
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
```

```
                throw new ExceptionInInitializerError(e);
            }
        }

        //随机生成字符串
        public static String randomStr(int size) {
            //Define an empty string
            String result = "";
            for (int i = 0; i < size; ++i) {
                //Generate an int type integer between 97 ~ 122
                int intVal = (int) (Math.random() * 26 + 97);
                //Force conversion (char) intVal Convert the corresponding value to the corresponding character, and splicing the characters
                result = result + (char) intVal;
            }
            //Output string
            return result;
        }
        //创建数据表
        public static void createTables() throws SQLException {

            try {
                //获取数据库连接
                Connection conn = conn = DriverManager.getConnection(url, username, password);
                //实例化 Statement
                //最后的 ENGINE=MyISAM DEFAULT    实现更改数据库引擎的作用
                String sql = "CREATE TABLE `test1` (`id` int(11) DEFAULT NULL,  `user_name` varchar(100) DEFAULT NULL,  `password` varchar(100) DEFAULT NULL, `randomchar1` varchar(100) DEFAULT NULL,  `randomchar2` varchar(100) DEFAULT NULL,  `randomchar3` varchar(100) DEFAULT NULL,  `randomchar4` varchar(100) DEFAULT NULL,  `randomchar5` varchar(100) DEFAULT NULL) ENGINE=MyISAM DEFAULT CHARSET=utf8;";

                //运行静态 sql 语句
                Statement pstmt = conn.createStatement() ;
                int rest = pstmt.executeUpdate(sql) ;
                //关闭连接
                pstmt.close();
                conn.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

```
    public static void insert(int insertNum){

        //开始时间
        Long begin = System.currentTimeMillis();
        System.out.println("Start Inserting Data...");

        try {
            //获取数据库连接
            Connection conn = conn = DriverManager.getConnection(url, username,
password);
            //实例化 Statement
            //将 sql 语句中要插入的数据，用 ？来代替
            String sql = "INSERT INTO test1 (id, user_name, password, randomchar1,
randomchar2, randomchar3, randomchar4, randomchar5) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";

            conn.setAutoCommit(false);
            //执行动态 sql 语句
            PreparedStatement pstmt = conn.prepareStatement(sql);
            //对动态 sql 语句的 ？进行逐个赋值      有 8 个 ？所以，要对 8 个数据进行
set（setInt 或 setString）
            for(int i = 0; i < insertNum; i++) {
                pstmt.setInt(1, i);
                pstmt.setString(2,randomStr(8));
                pstmt.setString(3, randomStr(8));
                pstmt.setString(4, randomStr(8));
                pstmt.setString(5, randomStr(8));
                pstmt.setString(6, randomStr(8));
                pstmt.setString(7, randomStr(8));
                pstmt.setString(8, randomStr(8));
                pstmt.addBatch();
            }
            pstmt.addBatch();
            pstmt.executeBatch();
            //提交事务
            conn.commit();

            //关闭连接
            pstmt.close();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }

        //结束时间
```

```
        Long end = System.currentTimeMillis();
        System.out.println("insert "+insertNum+" data is completed!");
        System.out.println("Time-consuming : " + (end - begin) / 1000 + "seconds");
    }

    public static void main(String[] args) throws SQLException {

        createTables();
        insert(1000000);
    }

}
```