# 为什么存入mysql数据库中的timestamp，晚了13个小时

## 查看数据库时区

```
show variables like '%time_zone%';
select @@global.system_time_zone;
select @@global.time_zone;
```

可以得到默认数据库时区：
system_time_zone | CST |
time_zone | SYSTEM|

## CST时区：4个含义

> CST可以为如下4个不同的时区的缩写：
> 1，美国中部时间：Central Standard Time (USA) UT-6:00 ，又美国从“3月11日”至“11月7日”实行夏令时，美国中部时间改为 UTC-05:00
> 2，澳大利亚中部时间：Central Standard Time (Australia) UT+9:30
> 3，中国标准时间：China Standard Time UT+8:00
> 4，古巴标准时间：Cuba Standard Time UT-4:00

> PS:即中国标准时间UT+8,和美国UT-5，中间相差13个小时

## 查看java程序运行的本地时区

```
TimeZone.getDefault();//得到"Asia/Shanghai"
```

## debug与源码分析：

> 1，测试发现，客户端到java程序端的时间戳是正确的，即通过mybatis写入数据库之前时间戳是正确的
> 2，**从mybatis一路跟踪：mybatis SqlTimestampTypeHandler.setNonNullParameter()->mybatis PreparedStatement.setTimestamp-》mysql-connector 》**

> **preparedStatement.setTimestamp()-》preparedStatement.setTimestampInternal()-》TimeUtil.changTimestamp()，通过计算本地时区和数据库时区差值，得到数据的时间戳，再转成SimpleDateFormat.format yyyy-MM-dd HH:mm:ss格式的时间戳日期字符串，写入数据库**
> 3，问题：java运行的本地时区是"Asia/Shanghai"，那mysql-connector得到的数据库时区是什么样的？连接数据库的时候，mysql-connector会获取数据库的时区信息，如上数据库时区查询，得到SYSTEM,CST

# mysql-connector获取数据库时区

> 1，CST 的时区是一个很混乱的时区，在与 MySQL 协商会话时区时，Java 会误以为是 CST -0500，而非 CST +0800

```
private void configureTimezone() throws SQLException {
                String configuredTimeZoneOnServer = (String) this.serverVariables
                                .get("timezone");

                if (configuredTimeZoneOnServer == null) {
                        configuredTimeZoneOnServer = (String) this.serverVariables
                                        .get("time_zone");

                        if ("SYSTEM".equalsIgnoreCase(configuredTimeZoneOnServer)) {
                                configuredTimeZoneOnServer = (String) this.serverVariables
                                                .get("system_time_zone");//得到CST，mysql-connector以为的CST是美国
                        }
                }
        ...
}
```

> 2，TimeZone.getTimeZone(canonicalTimezone)得到CST，mysql-connector以为的CST是美国的CST-5:00，{"CST", "America/Chicago"}
> 3，mysql-connector ZoneInfoFile class时区简写和时区对应关系

```
{{"ACT", "Australia/Darwin"},
 {"AET", "Australia/Sydney"},
 {"AGT", "America/Argentina/Buenos_Aires"},
 {"ART", "Africa/Cairo"},
 {"AST", "America/Anchorage"},
 {"BET", "America/Sao_Paulo"},
 {"BST", "Asia/Dhaka"},
 {"CAT", "Africa/Harare"},
 {"CNT", "America/St_Johns"},
 {"CST", "America/Chicago"},
 {"CTT", "Asia/Shanghai"},
 {"EAT", "Africa/Addis_Ababa"},
 {"ECT", "Europe/Paris"},
 {"IET", "America/Indiana/Indianapolis"},
 {"IST", "Asia/Kolkata"},
 {"JST", "Asia/Tokyo"},
 {"MIT", "Pacific/Apia"},
 {"NET", "Asia/Yerevan"},
 {"NST", "Pacific/Auckland"},
 {"PLT", "Asia/Karachi"},
 {"PNT", "America/Phoenix"},
 {"PRT", "America/Puerto_Rico"},
 {"PST", "America/Los_Angeles"},
 {"SST", "Pacific/Guadalcanal"},
 {"VST", "Asia/Ho_Chi_Minh"}};
```

# 如何解决

## 一，修改数据库时区

```
set global time_zone = '+8:00';//设置全局时区为东八区
set time_zone = '+8:00'; //
flush privileges;//刷新权限使设置立即生效
```

## 二，添加jdbc参数：serverTimezone=GMT%2B8

```
db?useUnicode=true&characterEncoding=UTF-8&useAffectedRows=true&useTimezone=true&serverTimezone=GMT%2B8
```

# 会有什么问题

> 1，因为老数据是基于CST-5:00，得到的时间戳日期字符串(yyyy-MM-dd HH:mm:ss.SSS)，写入数据库中，改了数据库时区或修改了JDBC的时区配置，会导致旧数据比以前慢13个小时

# 那旧数据怎么办

> 1，创建一个mybatis TimstampTypehandler专门处理timestamp类型，将某个时间以前的时间戳加上13个小时的时间戳间隔，即可

```
@MappedJdbcTypes(JdbcType.TIMESTAMP)
@MappedTypes(Timestamp.class)
public class TimestampHandler extends SqlTimestampTypeHandler {
    @Override
    public void setNonNullParameter(PreparedStatement ps, int i, Timestamp parameter, JdbcType jdbcType)
                    throws SQLException {
        ps.setTimestamp(i, parameter);
    }

    @Override
    public Timestamp getNullableResult(ResultSet rs, String columnName)
                    throws SQLException {
        //TimeZone tz=TimeZone.getDefault();
        //TimeZone.setDefault(TimeZone.getTimeZone("Asia/Shanghai"));
        Timestamp timestampTemp=rs.getTimestamp(columnName);
        long lt=timestampTemp.getTime();
        long timestampSplit=1590249600000L;//2020-05-24 00:00:00的毫秒时间戳
        if(timestampSplit>lt){
            Timestamp timestamp=new Timestamp(lt+13*60*60*1000);
            return timestamp;
        }else{
            return timestampTemp;
        }

    }

    @Override
    public Timestamp getNullableResult(ResultSet rs, int columnIndex)
                    throws SQLException {
        TimeZone.setDefault(TimeZone.getTimeZone("Asia/Shanghai"));
        Timestamp timestampTemp=rs.getTimestamp(columnIndex);
        long lt=timestampTemp.getTime();
        long timestampSplit=1590249600000L;//2020-05-24 00:00:00的毫秒时间戳
        if(timestampSplit>lt){
            Timestamp timestamp=new Timestamp(lt+13*60*60*1000);
            return timestamp;
        }else{
            return timestampTemp;
        }
    }

    @Override
    public Timestamp getNullableResult(CallableStatement cs, int columnIndex)
                    throws SQLException {
        TimeZone.setDefault(TimeZone.getTimeZone("Asia/Shanghai"));
        Timestamp timestampTemp=cs.getTimestamp(columnIndex);
        long lt=timestampTemp.getTime();
        long timestampSplit=1590249600000L;//2020-05-24 00:00:00的毫秒时间戳
        if(timestampSplit>lt){
            Timestamp timestamp=new Timestamp(lt+13*60*60*1000);
            return timestamp;
```

```
        }else{
            return timestampTemp;
        }
    }
}
```

# 多人开发，timestamp时间戳使用规约

> 1，接口参数涉及时间，都用时间戳，精确到秒或毫秒，全项目统一
> 2，时间戳参数直接入库，不要在代码层再做一次SimpleDateFormat.format yyyy-MM-dd HH:mm:ss.SSS转换，这样会附加本地时区，导致时间戳失效，mysql connector在入库前对timestamp类型做了本地时区和数据库时区差值计算的