# 开发笔记day2

## 分布式系统中唯一主键生成策略

Mybatis-plus在插入数据时，如果没有手动的设置逐渐，则mp会自动的生成主键，其生成策略有：

### 数据库自增序列

在Mysql中关键字为auto increment。

好的特性：理解简单，易于操作

可能存在的问题：

1. 容易被猜出业务逻辑
2. UUID生成的时间太长，且不符合数据库设计规范
3. 在做数据库分表后，后一张表的id需要根据前一张表最后一行数据中的id来选取，即需要访问前面的表，不方便。

### UUID策略

通过时间序列每次生成一个随机且唯一的值

优点：不需要关注上一张表的数据，随机生成UID

缺点：无序性：无法排序

### Redis生成ID

通过Redis原子操作来实现。比如一个Redis集群中有5台redis主机，初始化值为1，2，3，4，5，每个redis值自增步长为5：

A 1 6 11 16...

B 2 7 12 17…

C 3 8 13 18

D 4 9 14 19

E 5 10 15 20

优点： ID只要根据主机值来选取，这样避免了单点故障，并且不依赖于数据库，数字ID天然排序，对分页和排序很有帮助

缺点： 配置和编码麻烦

### Mybatis-plus自带策略（雪花算法）

使用41bit作为毫秒数，10bit做机器id（5个bit的数据中心，5个bit的机器id），12bit作为毫秒内的流水号，最后一个符号位永远为0.

## 主键生成策略配置

如果不配置，mybatis-plus自动使用自带的雪花算法，随机生成一个19位的id。

如果需要配置, 可以使用注解：@TableId(type = IdType.AUTO)

AUTO: 自动增长

ID\_WORKER：生成19位的数字类型作为主键。 3.3版本以后使用ASSIGN\_ID

ID\_WORKER\_STR：生成19位的字符串类型作为主键. 3.3版本以后使用ASSIGN\_ID\_STR

INPUT：开发者输入主键

NONE：无策略

UUID：随机生成

## 自动填充

在除去主键，其他字段没有手动输入时，Mybatis也提供了自动填充方法。当然这需要开发者先手动实现一次，之后需要填充的字段加上注解便可。

比如当使用create\_time和update\_time时，数据库如果没有设置默认值current\_TimeStamp时，则可以通过实现接口MetaObjectHandle设置。

在model层:

@TableFill(type=FieldFill.INSERT)

Date createTime;

@TableFill(type=FiledFill.INSERT\_UPDATE)

Date updateTime;

然后创建一个handle包，在里面实现这个接口，如

@Component //这里得记得把handle类交给Spring管理

public class MyMetaObjectHandle implements MetaObjectHandler {

@Override

public void insertFill(MetaObject metaObject) {

this.setFieldValByName("createTime",new Date(),metaObject);

this.setFieldValByName("updateTime",new Date(),metaObject);

}

@Override

public void updateFill(MetaObject metaObject) {

this.setFieldValByName("updateTime",new Date(),metaObject);

}

}

## 乐观锁

主要用来解决丢失更新问题。

当不考虑事务的隔离性（也就是发生并发时），可能会产生一些问题。

在“读“数据时，可能会产生 脏读 幻读 不可重复读

在“写“数据时，可能会产生 丢失更新问题。

**丢失更新问题**：发生并发时，多个人对同一条数据修改（每个人都开始事务，修改数据，提交事务），最后一个提交者会覆盖掉前面所有的更新。

丢失更新问题的解决：

悲观锁：同一时间只能一个线程操作数据。

串行操作：效率低，多个线程需要一个个排队等待操作，基数一大效率就低。

乐观锁：更新一条记录时，希望别人没有进行更新，也就是实现线程安全的数据更新。

实现方式：根据version版本号实现。

如：

Id money version

|  |  |  |
| --- | --- | --- |
| 1 | 500 | 1 |

当一个线程进入开始操作，开始事务，读取版本号version=1

1. 进行操作前判断：如果当前版本号和数据库版本号相同，允许更新，否则不允许
2. 假设版本相同，set money =8000, version = version+1;版本号会自动++
3. 提交事务

如果此时发生并行，另一个线程也在尝试修改，但是这个线程首先提交，则数据库的version会变为2，而第二个线程拿到的version为1，不相等则更新失败。

如何用mybatis-plus实现乐观锁？

1. 在数据库表中加version字段，对应实体类加上属性
2. 在version字段上标记注解@Version
3. 配置乐观锁插件 OptimisticLockerInnerInterceptor, 建议创建一个配置类，写在配置类里。

## 解决的Bug：

1. java.lang.IllegalStateException:Error processing condition on org.springframework.boot.autoconfigure.transaction.TransactionAutoConfiguration$EnableTransactionManagementConfiguration

答：jar包冲突了，**在pom.xml中去掉spring-jdbc的版本**

1. java.lang.IllegalArgumentException: Property 'sqlSessionFactory' or 'sqlSessionTemplate' are required

在sqlSessionFactory和sqlSessionTemplate两个工厂方式中不知道选什么，在使用mybatis时，添加mybatis启动器就好了，并且需要记得在application.yml或者application.properties中添加**mybatis-mapper-locations=classpath\*:mapper/\*.xml**

**<dependency>**

**<groupId>org.mybatis.spring.boot</groupId>**

**<artifactId>mybatis-spring-boot-starter</artifactId>**

**<version>1.3.2</version>**

**</dependency>**

1. org.springframework.beans.factory.BeanDefinitionStoreException: Failed to read WebprojectAppliIndexOutOfBoundsException: Index 0 out of bounds for length 0 at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64) at java.base/

出现这种问题可能是jdk版本不匹配，或者jar包冲突，点击maven dependence视察。

1. Mapper出现注入失败，则需要添加@MapperScan在正确的路径下，或者在Mapper文件上添加注释@Mapper，此时mybatis-plus自动会进行强化。