第八篇 逻辑删除实现

一、物理删除与逻辑删除

- 物理删除：指文件存储所用到的磁存储区域被真正的擦除或清零，这样删除的文件是不可以恢复的，物理删除是计算机处理数据时的一个概念。如果在数据库中直接使用delete、drop删除了表数据，如果没有备份的话，数据就很难恢复了。
- 逻辑删除（软删除）：逻辑删除就是对要被删除的数据打上一个删除标记，通常使用一个deleted字段标示行记录是不是被删除，比如该数据有一个字段deleted，当其值为0表示未删除，值为1表示删除。那么逻辑删除就是将0变成1。在逻辑上是数据是被删除的，但数据本身是依然存在的。

两者的优劣：

- 物理删除一定程度上删除了“无用”的数据，降低了表的数据量，对性能肯定是有好处的；但是如果没有备份的话，数据很难恢复。也无法对历史数据进行数据分析。
- 逻辑删除恢复的话只要修改ideleted等类似的状态标示字段就可以了，但是表的数据量肯定会比物理删除增加了，并且查询时经常要考虑到deleted字段，对索引都会有影响。

所以一张表的数据是否采用逻辑删除，还要根据数据的重要性、数据量、查询性能以及业务需求等因素综合判断。

二、逻辑删除实现

- 首先为需要逻辑删除的表增加一个deleted字段作为逻辑删除字段，并且设置其默认值为0，如下：

```sql
CREATE TABLE `admin` (
    `id` bigint(20) NOT NULL AUTO_INCREMENT COMMENT '主键ID',
    `name` VARCHAR(30) NULL DEFAULT NULL COMMENT '姓名',
    `age` INT(11) NULL DEFAULT NULL COMMENT '年龄',
    `email` VARCHAR(50) NULL DEFAULT NULL COMMENT '邮箱',
    `deleted` TINYINT(4) NOT NULL DEFAULT '0' COMMENT '逻辑删除标记',
    PRIMARY KEY (`id`)
);
```

- 给数据库表对应的实体类字段上加上@TableLogic注解：

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
@EqualsAndHashCode(callSuper = true)
public class Admin extends Model<Admin> {
    private static final long serialVersionUID = 62973979471089565 92L;
    private Long id;
    private String name;
    private Integer age;
    private String email;
    @TableLogic
    private Integer deleted;
}
```

为Admin创建AdminMapper，继承BaseMapper

三、API使用方法

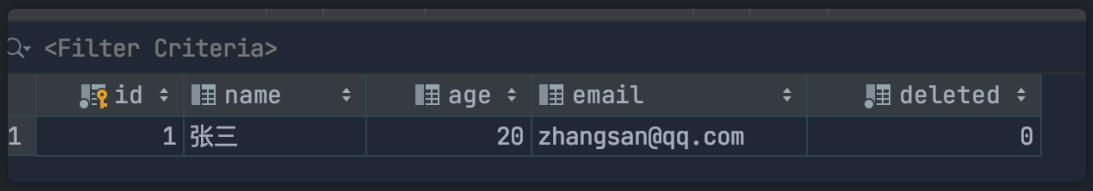
> 测试AdminMapper

3.1.插入一条数据

插入数据的时候，不需要为deleted字段赋值

```java
@Test
public void testInsert() {
  Admin admin = Admin.builder()
    .name("张三")
    .age(20)
    .email("zhangsan@qq.com")
    .build();
  int row = adminMapper.insert(admin);
  assertEquals(1, row);
}
```

deleted采用默认值0（未删除），新插入的数据都是未删除的数据

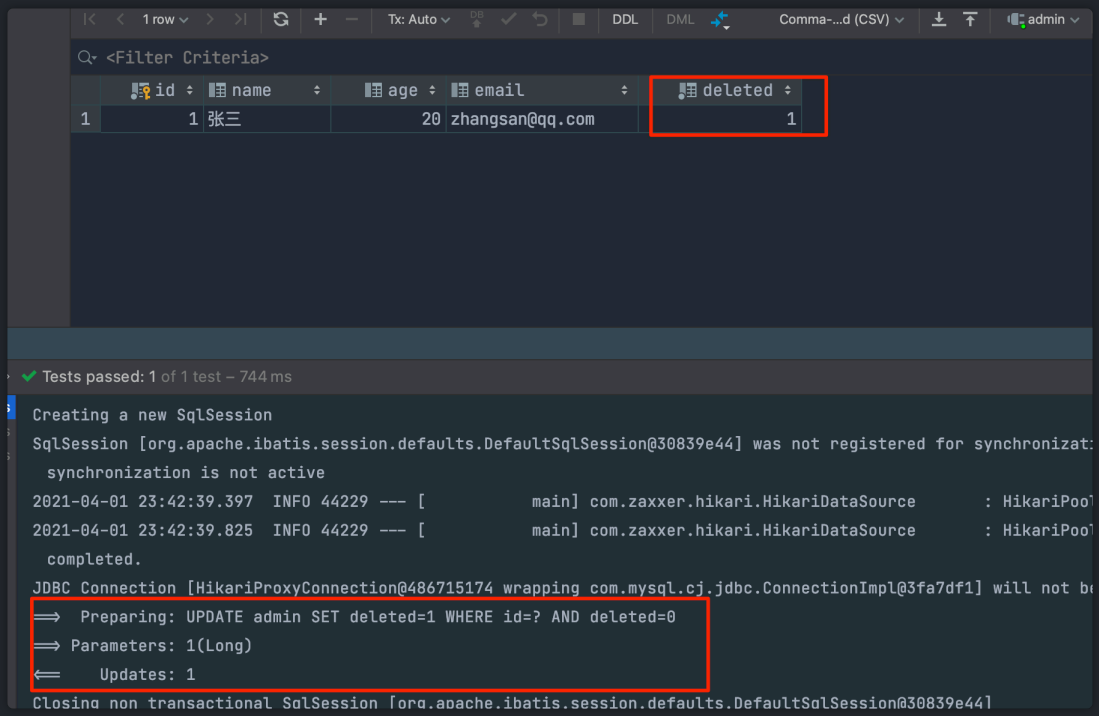


3.2.删除一条记录：

执行如下Mybatis Plus API删除操作

```java
@Test
void testDelete() {
  int row = adminMapper.deleteById(1L);
  assertEquals(1, row);
}
```

查看数据库可以发现这条数据仍然存在，只不过逻辑删除字段值被设置为1：



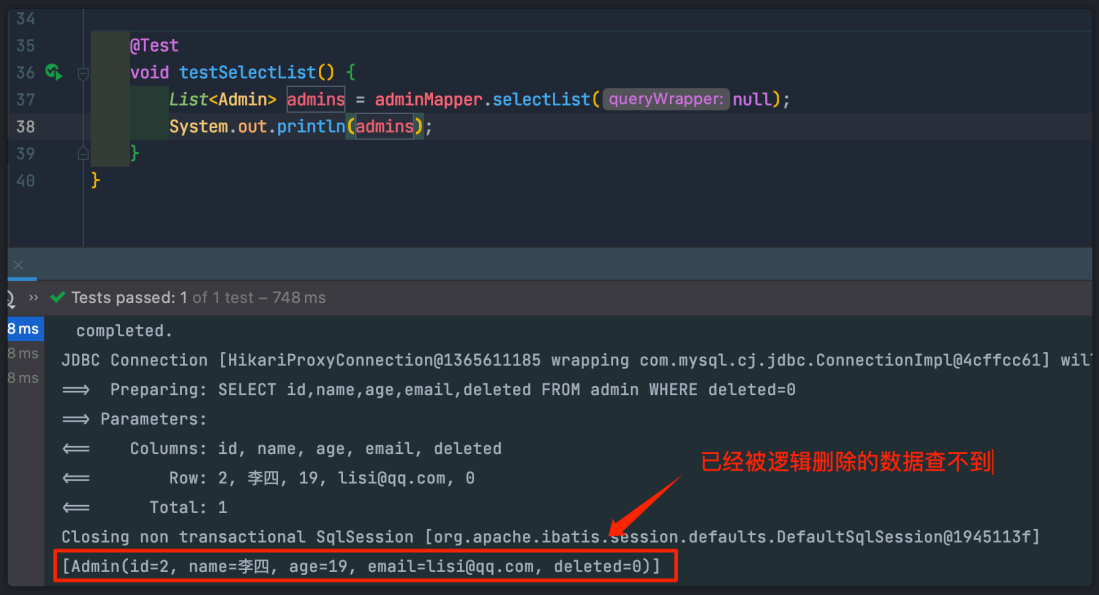
3.3.查询一条记录

- 当我们使用MP逻辑删除的功能之后，比如执行查询、修改的方法，MP会为我们自动加上未删除的条件。是不会查到被逻辑删除的记录：

```
adminMapper.selectList(null);
```

会自动添加过滤条件`WHERE deleted=0`

```
SELECT id,name,age,email,deleted
FROM user
WHERE deleted=0
```



- 当我们查询数据时，查询结果不希望包含逻辑删除字段，可以加如下的注解

```
@TableLogic
@TableField(select = false)
private Integer deleted;
```

执行的SQL如下（注意查询结果不包含deleted字段）：

```
SELECT id,name,age,email
FROM user
WHERE deleted=0
```

四、全局配置参数

通常在一个比较正规的管理项目中，逻辑删除字段不允许随意命名，所有表的逻辑删除字段使用相同的名称（比如：deleted）。我们可以在application.yml中添加全局配置，这样就不需要在每一个实体类上面都添加 @TableLogic注解了：

> **注意：当全局配置和 `@TableLogic` 局部配置同时存在，则以实体上注解为准，优先级更高。**

```yaml
#全局逻辑删除字段值
mybatis-plus:
  global-config:
    db-config:
      logic-delete-field: deleted
```

默认情况下，逻辑已删除值为1，逻辑未删除值为0。我们也可以在application.yml中进行修改：

```yaml
#逻辑已删除值(默认为 1)
#逻辑未删除值(默认为 0)
mybatis-plus:
  global-config:
    db-config:
      logic-delete-value: 1
      logic-not-delete-value: 0
```