延迟加载

一、定义

1、延迟加载：

就是在需要用到数据时才进行加载，不需要用到数据时就不加载数据。延迟加载也称懒加载.

举个例子：如果查询订单并且关联查询用户信息。如果先查询订单信息即可满足要求，当我们需要查询用户信息时再查询用户信息。把对用户信息的按需去查询就是延迟加载。 所以延迟加载即先从单表查询、需要时再从关联表去关联查询，大大提高数据库性能，因为查询单表要比关联查询多张表速度要快。

1）好处：

先从单表查询，需要时再从关联表去关联查询，大大提高数据库性能，因为查询单表要比关联查询多张表速度要快。

2）坏处：

因为只有当需要用到数据时，才会进行数据库查询，这样在大批量数据查询时，因为查询工作也要消耗时间，所以可能造成用户等待时间变长，造成用户体验下降。

2、立即加载

不管用不用，只要一调用方法，马上发起查询。

3、使用场景

在对应的四种表关系中：一对多，多对一，一对一，多对多

一对多，多对多：通常情况下我们都是采用延迟加载。

多对一，一对一：通常情况下我们都是采用立即加载。

二、加载时机

MyBatis根据对关联对象查询的select语句的执行时机，分为三种类型：

a. 直接加载：执行完对主加载对象的 select 语句，马上执行对关联对象的 select查询。

b. 侵入式延迟： 执行对主加载对象的查询时，不会执行对关联对象的查询。但当要访问主加载对象的详情属性时，就会马上执行关联对象的select查询（例如配置文件IUserdao.xml里的collection的select语句）。

c. 深度延迟： 执行对主加载对象的查询时，不会执行对关联对象的查询。访问主加载对象的详情时也不会执行关联对象的select查询。只有当真正访问关联对象的详情时，才会执行对关联对象的 select查询。

三、mybatis延迟设置（重点）

设置名	描述	有效值	默认值
lazyLoadingEnabled	延迟加载的全局开关。当开启时，所有关联对象都会延迟加载。 特定关联关系中可通过设置 `fetchType` 属性来覆盖该项的开关状态。	true | false	false
aggressiveLazyLoading	开启时，任一方法的调用都会加载该对象的所有延迟加载属性。 否则，每个延迟加载属性会按需加载（参考 `lazyLoadTriggerMethods`)。	true | false	false （在 3.4.1

深入了解侵入式延迟和侵入式延迟

1、开启侵入式延迟

```
<!--全局参数设置-->
<settings>
   <!--延迟加载总开关-->
   <setting name="lazyLoadingEnabled" value="true"/>
   <!--侵入式延迟加载开关-->
   <!--3.4.1版本之前默认是true，之后默认是false-->
   <setting name="aggressiveLazyLoading" value="true"/>
</settings>
```

若不访问主对象users的属性什么的，

```
@Test
public void testFindAll(){
    List<User> users = userDao.findAll();
}
```

则：只执行主sql语句

```
saction  - Opening JDBC Connection
aSource  - Created connection 1728790703.
findAll  - ==>  Preparing: select * from user
findAll  - ==> Parameters:
findAll  - <==      Total: 6
saction  - Closing JDBC Connection [com.mysql.jdbc

aSource  - Returned connection 1728790703 to pool.
```

一旦访问

```
@Test
public void testFindAll(){
    List<User> users = userDao.findAll();
    for(User user : users){
        System.out.println(user);
    }
}
```

则延迟加载：

```
ion  - Opening JDBC Connection
rce  - Created connection 1728790703.
All  - ==>  Preparing: select * from user
All  - ==> Parameters:
All  - <==      Total: 6

Uid  - ==>  Preparing: select * from account where uid

Uid  - ==> Parameters: 41(Integer)
Uid  - <==      Total: 0
 CST 2018}

Uid  - ==>  Preparing: select * from account where uid

Uid  - ==> Parameters: 42(Integer)
Uid  - <==      Total: 0
5:09:37 CST 2018}
```

2、开启深度延迟加载

```
<!--全局参数设置-->
<settings>
    <!--延迟加载总开关-->
    <setting name="lazyLoadingEnabled" value="true"/>
    <!--侵入式延迟加载开关-->
    <!--3.4.1版本之前默认是true，之后默认是false-->
    <setting name="aggressiveLazyLoading" value="false"/>
</settings>
```

如果只访问主对象User的属性

```
@Test
public void testFindAll(){
```

```
    List<Account> accounts = accountDao.findAll();

    for(Account account : accounts){
        System.out.println("--------每个account的信息------------");
        System.out.println(account.getMoney());
    }
}
```

则立即加载

```
  .IUserDao matches criteria [is assignable to Object]
 2020-03-29 16:12:54,649 333    [           main] DEBUG ansaction.jdbc.JdbcTransaction  - Opening JDBC Con
 2020-03-29 16:12:54,862 546    [           main] DEBUG source.pooled.PooledDataSource  - Created connecti
 2020-03-29 16:12:54,864 548    [           main] DEBUG theima.dao.IAccountDao.findAll  - ==>  Preparing:
 2020-03-29 16:12:54,884 568    [           main] DEBUG theima.dao.IAccountDao.findAll  - ==> Parameters:
 2020-03-29 16:12:54,932 616    [           main] DEBUG theima.dao.IAccountDao.findAll  - <==      Total:
 --------每个account的信息------------
 1000.0
 --------每个account的信息------------
 1000.0
 --------每个account的信息------------
 2000.0
```

若访问关联对象属性

```
@Test
    public void testFindAll(){
        List<Account> accounts = accountDao.findAll();
        int i=45;
        for(Account account : accounts){
            System.out.println("--------每个account的信息------------");
            System.out.println(account.getMoney());

            System.out.println(userDao.findById(i));
            i++;
        }
    }
```

则：

```
--------每个account的信息------------
1000.0
2020-03-29 16:15:22,757 711    [           main] DEBUG .itheima.dao.IUserDao.findById  - ==>  Preparing: select * from user w
2020-03-29 16:15:22,757 711    [           main] DEBUG .itheima.dao.IUserDao.findById  - ==> Parameters: 45(Integer)
2020-03-29 16:15:22,764 718    [           main] DEBUG .itheima.dao.IUserDao.findById  - <==      Total: 1
User{id=45, username='传智播客', address='北京金燕龙', sex='男', birthday=Sun Mar 04 12:04:06 CST 2018}
--------每个account的信息------------
1000.0
2020-03-29 16:15:22,765 719    [           main] DEBUG .itheima.dao.IUserDao.findById  - ==>  Preparing: select * from user w
2020-03-29 16:15:22,765 719    [           main] DEBUG .itheima.dao.IUserDao.findById  - ==> Parameters: 46(Integer)
2020-03-29 16:15:22,770 724    [           main] DEBUG .itheima.dao.IUserDao.findById  - <==      Total: 1
User{id=46, username='老王', address='北京', sex='男', birthday=Wed Mar 07 17:37:26 CST 2018}
--------每个account的信息------------
2000.0
2020-03-29 16:15:22,770 724    [           main] DEBUG .itheima.dao.IUserDao.findById  - ==>  Preparing: select * from user w
2020-03-29 16:15:22,771 725    [           main] DEBUG .itheima.dao.IUserDao.findById  - ==> Parameters: 47(Integer)
2020-03-29 16:15:22,771 725    [           main] DEBUG .itheima.dao.IUserDao.findById  - <==      Total: 0
null
```

上面都是通过在mybatis.xml文件中统一配置的深度延迟加载，倘若只希望某些查询支持深度延迟加载的话可以在resultMap中的collection或association添加**fetchType**属性，配置为lazy之后是开启深度延迟，配置eager是不开启深度延迟。fetchType属性将取代全局配置参数lazyLoadingEnabled的设置