第5章 MyBatis 缓存机制

学习目标

- 理解 MyBatis 一级缓存机制，能够描述一级缓存机制的概念和特点。
- 理解 MyBatis 二级缓存机制，能够描述二级缓存机制的概念和特点。
- 掌握 MyBatis 处理一级缓存的方法，能够灵活运用一级缓存机制。
- 掌握 MyBatis 处理二级缓存的方法，能够灵活运用二级缓存机制。
- 掌握 MyBatis 集成 EhCache 缓存的方法，能够准确配置 EhCache 缓存。


视频讲解

为了降低高并发访问给数据库带来的压力，大型企业项目中都会使用缓存。使用缓存可以降低磁盘 I/O 操作、减少程序与数据库的交互，帮助程序迅速获取所需数据，提升系统响应速度，对优化系统整体性能具有重要意义。本章将对 MyBatis 的缓存种类和 EhCache 缓存进行讲解。

5.1 MyBatis 缓存分类

在实际业务场景中，如果经常执行相同 SQL 语句的查询操作，会导致频繁与数据库交互，从而造成磁盘性能下降和资源浪费的情况。MyBatis 的缓存机制很好地解决了这一类问题，MyBatis 缓存将 SQL 语句或结果保存在内存中，以便下次执行相同的 SQL 时直接从缓存中读取，从而不用再次访问数据库。本节将对 MyBatis 的一级缓存和二级缓存的概念、特点及应用进行讲解。

5.1.1 一级缓存

1. MyBatis 一级缓存的概念及特点

MyBatis 一级缓存机制是指在同一个 SqlSession 中，对相同的 SQL 语句和参数，只执行一次数据库查询，第一次查询的结果会被缓存起来，后续的查询会直接从缓存中获取，从而提高查询效率。

MyBatis 一级缓存机制默认开启，它使用一个 HashMap 对象来存储缓存数据，key 为 hashCode+sqlId+SQL 语句，value 为查询结果对象。

MyBatis 一级缓存机制有以下特点。

- 一级缓存是 SqlSession 级别的，不同的 SqlSession 之间的缓存是相互隔离的。
- 一级缓存只对 select 语句有效，对 insert、update、delete 语句无效。

MyBatis 一级缓存也会有缓存失效或清空的情况，具体场景如下。

- 当执行任何 insert、update、delete 语句时,会清空当前 SqlSession 的所有缓存。
- 当执行不同的 SQL 语句或参数时,会重新生成新的 key 值,从而导致缓存失效。
- 当手动调用 sqlSession.clearCache()方法时,会清空当前 SqlSession 的所有缓存。
- 当 SqlSession 关闭或提交时,会清空当前 SqlSession 的所有缓存。

需要注意的是,sqlSession 是 SqlSession 类的对象。

2. MyBatis 一级缓存的应用

在深入理解 MyBatis 一级缓存的概念、特点和应用场景后,这里通过一个示例演示 MyBatis 一级缓存的应用,从而掌握 MyBatis 一级缓存的机制和使用,具体步骤如下。

(1) 使用第 2 章 2.5 节中的数据库 textbook 和数据表 education。

(2) 在 IDEA 中新建 chapter05 项目,将 MyBatis 的 JAR 包和 MySQL 驱动的 JAR 包添加到 WEB-INF 下的 lib 文件夹中。

(3) 创建 Education 类,该类和例 2-1 所示内容相同,此处不再赘述。

(4) 创建 mybatis-config.xml 配置文件,具体代码可参照例 2-2,此处不再赘述。

(5) 创建映射文件 EducationMapper.xml,具体代码如例 5-1 所示。

例 5-1 EducationMapper.xml。

```
1  <?xml version = "1.0" encoding = "UTF - 8"?>
2  <! DOCTYPE mapper PUBLIC " - //mybatis.org//DTD Mapper 3.0//EN"
3          "http://mybatis.org/dtd/mybatis - 3 - mapper.dtd">
4  < mapper namespace = "education">
5      < select id = "findEducationById"
6      resultType = "com.qfedu.pojo.Education">
7          select * from education where id = #{id}
8      </select >
9      < update id = "updateEducation"
10     parameterType = "com.qfedu.pojo.Education">
11         update education set price = #{price} where id = #{id}
12     </update >
13 </mapper >
```

在例 5-1 中,第 7 行代码表示根据 id 查询教材的 SQL 语句,第 11 行代码表示根据 id 修改教材的 SQL 语句。

(6) 创建 TestCache01 类用于测试 MyBatis 一级缓存机制,具体代码如例 5-2 所示。

例 5-2 TestCache01.java。

```
1  public class TestCache01 {
2      public static void main(String[] args) {
3          /* 创建输入流 */
4          InputStream inputStream = null;
5          /* 将 MyBatis 配置文件转换为输入流 */
6          try {
7              inputStream =
8                      Resources.getResourceAsStream("mybatis - config.xml");
9          } catch (IOException e) {
10             e.printStackTrace();
11         }
12         /* 通过 SqlSessionFactoryBuilder 创建 SqlSessionFactory 对象 */
13         SqlSessionFactory build =
```

```
                new SqlSessionFactoryBuilder().build(inputStream);
        /* 通过 SqlSessionFactory 创建 SqlSession 对象 */
        SqlSession sqlSession = build.openSession(true);
        Education education1 = sqlSession.selectOne(
                                    "education.findEducationById", 1);
        System.out.println(education1.toString());
        System.out.println("------------------------------");
        Education education2 = sqlSession.selectOne(
                                    "education.findEducationById", 1);
        System.out.println(education2.toString());
        /* 关闭事务 */
        sqlSession.close();
    }
}
```

执行 TestCache01 类，MyBatis 一级缓存测试结果如图 5-1 所示。

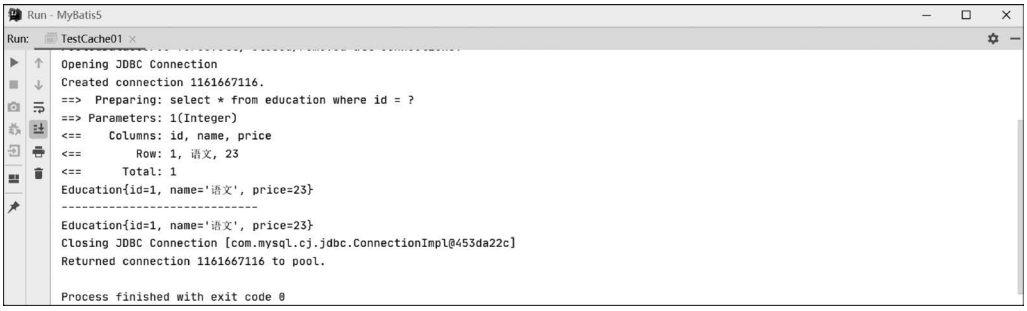

图 5-1　MyBatis 一级缓存测试结果

从图 5-1 的控制台打印结果可以看出，执行两次相同的查询语句，这是因为使用了 MyBatis 一级缓存机制。当程序第二次执行相同的查询语句时，日志并没有发出 SQL 语句，而是直接从一级缓存中获取了数据。

（7）新建 TestCache02 类，该类用于测试因修改数据信息导致一级缓存失效的场景，具体代码如例 5-3 所示。

例 5-3　TestCache02.java。

```
public class TestCache02 {
    public static void main(String[] args) {
        /* 创建输入流 */
        InputStream inputStream = null;
        /* 将 MyBatis 配置文件转换为输入流 */
        try {
            inputStream =
                    Resources.getResourceAsStream("mybatis-config.xml");
        } catch (IOException e) {
            e.printStackTrace();
        }
        /* 通过 SqlSessionFactoryBuilder 创建 SqlSessionFactory 对象 */
        SqlSessionFactory build =
                new SqlSessionFactoryBuilder().build(inputStream);
        /* 通过 SqlSessionFactory 创建 SqlSession 对象 */
        SqlSession sqlSession = build.openSession(true);
```

```
        //查询 Id 为 1 的数据信息
        Education education1 =
        sqlSession.selectOne("education.findEducationById", 1);
        System.out.println(education1.toString());
        //修改 Id 为 2 的价格信息
        Education education2 = new Education();
        education2.setId(2);
        education2.setPrice(19);
        sqlSession.update("education.updateEducation",education2);
        //再次查询 Id 为 1 的数据信息
        Education education3 =
        sqlSession.selectOne("education.findEducationById", 1);
        System.out.println(education3.toString());
        /*关闭事务*/
        sqlSession.close();
    }
}
```

执行 TestCache02 类,MyBatis 一级缓存失效的测试结果如图 5-2 所示。

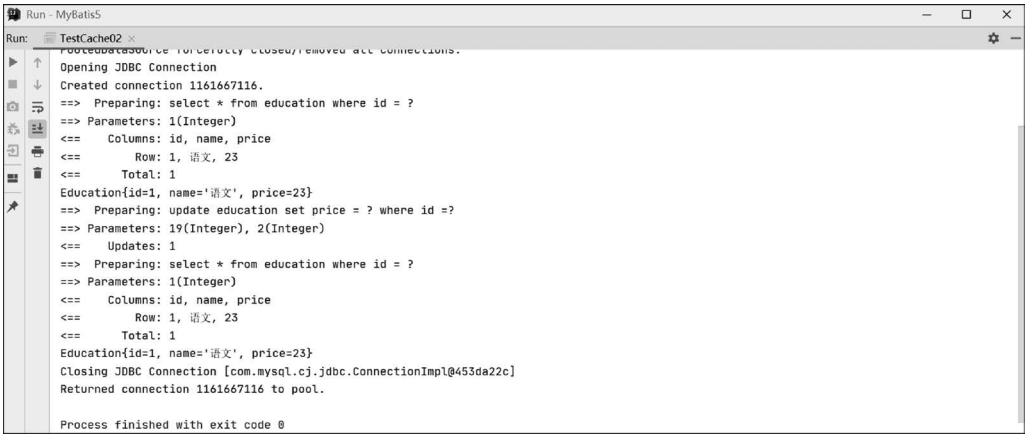

图 5-2 MyBatis 一级缓存失效的测试结果

从图 5-2 的控制台打印结果可以看出,程序首先查询 Id 为 1 的教材信息,其次更新 Id 为 2 的教材信息,然后再次查询 Id 为 1 的教材信息。由于程序执行了更新操作,符合本节介绍的 MyBaits 一级缓存失效的情况之一,所以当再次查询 Id 为 1 的教材信息时,缓存失效,只能再次执行 SQL 语句从数据库中读取结果。

5.1.2 二级缓存

1. MyBatis 二级缓存的概念及特点

MyBatis 二级缓存机制是指在不同的 SqlSession 中,对相同的 SQL 语句和参数,只执行一次数据库查询,第一次查询的结果会被缓存起来,后续的查询会直接从缓存中获取,从而提高查询效率。

MyBatis 二级缓存机制默认是关闭的,需要手动开启和配置。它使用一个 Cache 对象来存储缓存数据,同时支持自定义缓存的实现类或使用第三方缓存方案。

MyBatis 二级缓存机制有以下特点。

- 二级缓存是 Mapper 级别的缓存，多个 SqlSession 可以共享同一个 Mapper 的缓存，也称为全局缓存。
- 二级缓存只对 select 语句有效，对 insert、update、delete 语句无效。

MyBatis 二级缓存也会有失效或清空的情况，具体场景如下。

- 当执行任何 insert、update、delete 语句时，会刷新当前 Mapper 的所有缓存。
- 当执行不同的 SQL 语句或参数时，会重新生成新的 key 值，从而导致缓存失效。
- 当手动调用 sqlSession.clearCache()方法时，会清空当前 SqlSession 的所有缓存。
- 当 SqlSession 关闭或提交时，会清空当前 SqlSession 的所有缓存。

需要注意的是，sqlSession 是 SqlSession 类的对象。

2. MyBatis 二级缓存的应用

在深入理解 MyBatis 二级缓存的概念、特点和应用场景后，这里通过一个示例演示 MyBatis 二级缓存的应用，从而掌握 MyBatis 二级缓存的机制和使用，具体步骤如下。

(1) 使用本章 5.1.1 节中的 chapter05 项目。

(2) 在映射文件 EducationMapper.xml 中的< mapper >元素下添加如下代码。

```
< cache ></cache >
```

上述代码表示开启二级缓存机制。

(3) Education 类需要实现序列化接口，具体代码如下。

```
public class Education implements Serializable {}
```

(4) 新建 TestCache03 类用于测试二级缓存机制，具体代码如例 5-4 所示。

例 5-4 TestCache03.java。

```
1  public class TestCache03 {
2      public static void main(String[] args) {
3          /* 创建输入流 */
4          InputStream inputStream = null;
5          /* 将 MyBatis 配置文件转化为输入流 */
6          try {
7              inputStream =
8                      Resources.getResourceAsStream("mybatis-config.xml");
9          } catch (IOException e) {
10             e.printStackTrace();
11         }
12         /* 通过 SqlSessionFactoryBuilder 创建 SqlSessionFactory 对象 */
13         SqlSessionFactory build =
14                 new SqlSessionFactoryBuilder().build(inputStream);
15         /* 通过 SqlSessionFactory 创建 SqlSession 对象 */
16         SqlSession sqlSession1 = build.openSession(true);
17         SqlSession sqlSession2 = build.openSession(true);
18         Education education1 =
19         sqlSession1.selectOne("education.findEducationById", 1);
20         //事务提交
21         sqlSession1.commit();
22         System.out.println(education1.toString());
23         System.out.println("------------------------------");
24         Education education2 =
```

```
        sqlSession2.selectOne("education.findEducationById", 1);
        sqlSession2.commit();
        System.out.println(education2.toString());
        /* 关闭事务 */
        sqlSession1.close();
        sqlSession2.close();
    }
}
```

执行 TestCache03 类,MyBatis 二级缓存的测试结果如图 5-3 所示。

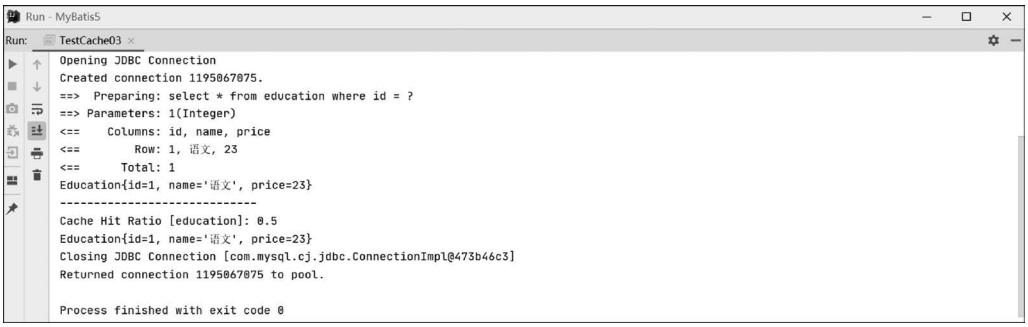

图 5-3　MyBatis 二级缓存的测试结果

从图 5-3 的控制台打印结果可以看出,在同一个 Java 程序中使用两个不同的 SqlSession 对象执行相同的查询语句,由于在配置文件中启用了 MyBatis 二级缓存,当程序第二次执行该查询语句时,日志并没有发出 SQL 语句,而是直接从二级缓存中读取了数据。

(5) 新建 TestCache04 类,该类用于测试因修改数据信息导致二级缓存失效的场景,具体代码如例 5-5 所示。

例 5-5　TestCache04.java。

```
public class TestCache04 {
    public static void main(String[] args) {
        /* 创建输入流 */
        InputStream inputStream = null;
        /* 将 MyBatis 配置文件转换为输入流 */
        try {
            inputStream =
                    Resources.getResourceAsStream("mybatis-config.xml");
        } catch (IOException e) {
            e.printStackTrace();
        }
        /* 通过 SqlSessionFactoryBuilder 创建 SqlSessionFactory 对象 */
        SqlSessionFactory build =
                new SqlSessionFactoryBuilder().build(inputStream);
        /* 通过 SqlSessionFactory 创建 SqlSession 对象 */
        SqlSession sqlSession1 = build.openSession(true);
        SqlSession sqlSession2 = build.openSession(true);
        SqlSession sqlSession3 = build.openSession(true);
        //查询 Id 为 1 的数据信息
        Education education1 =
        sqlSession1.selectOne("education.findEducationById", 1);
        sqlSession1.commit();
```

```
23          System.out.println(education1.toString());
24          //修改 Id 为 2 的价格信息
25          Education education2 = new Education();
26          education2.setId(2);
27          education2.setPrice(18);
28          sqlSession2.update("education.updateEducation",education2);
29          sqlSession2.commit();
30          //再次查询 Id 为 1 的数据信息
31          Education education3 =
32          sqlSession3.selectOne("education.findEducationById", 1);
33          sqlSession3.commit();
34          System.out.println(education3.toString());
35          /* 关闭事务 */
36          sqlSession1.close();
37          sqlSession2.close();
38          sqlSession3.close();
39      }
40  }
```

执行 TestCache04 类，MyBatis 二级缓存失效的测试结果如图 5-4 所示。

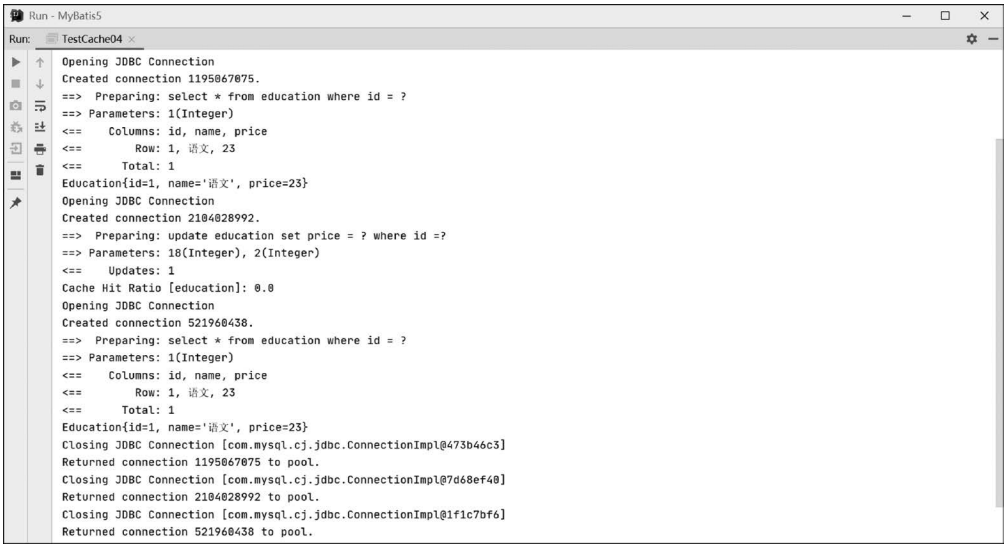

图 5-4　MyBatis 二级缓存失效的测试结果

从图 5-4 的控制台打印结果可以看出，程序首先查询 id 为 1 的教材信息，其次更新 id 为 2 的教材信息，然后再次查询 id 为 1 的教材信息。由于程序执行了更新操作，符合本节介绍的 MyBaits 二级缓存失效的情况之一，所以当再次查询 id 为 1 的教材信息时，缓存失效，只能再次执行 SQL 语句从数据库中读取结果。

5.2　EhCache 缓存

在实际开发中，很多项目会用到分布式架构。分布式架构是将项目拆分成若干个子项目并使它们协同发挥功能的解决方案。在分布式系统架构下，为了提升系统性能，通常会采用分布式缓存对缓存数据进行集中管理。由于 MyBatis 自身无法实现分布式缓存，需要整

合其他分布式缓存框架,如 EhCache 缓存。本节将对 EhCache 缓存概念、EhCache 下载和 MyBatis 整合 EhCache 缓存进行讲解。

5.2.1 EhCache 缓存简介

EhCache 缓存是一个纯 Java 进程的缓存框架,具有快速、精干等特点。EhCache 主要面向通用缓存、Java EE 和轻量级容器。

ehcache.xml 是 EhCache 的配置文件,一般存放在系统应用的 classpath 中。EhCache 的配置文件有其自身特有的层次结构,具体结构如下所示。

```
1  <?xml version = "1.0" encoding = "UTF - 8"?>
2  < ehcache >
3       < diskStore path = ""/>
4 < defaultCache
5       maxElementsInMemory = ""
6       maxElementsOnDisk = ""
7       eternal = ""
8       overflowToDisk = ""
9       diskPersistent = ""
10      timeToIdleSeconds = ""
11      timeToLiveSeconds = ""
12   diskSpoolBufferSizeMB = ""
13      diskExpiryThreadIntervalSeconds = ""
14      memoryStoreEvictionPolicy = ""
15   />
16 < cache name = "" eternal = ""
17      maxElementsInMemory = ""
18      overflowToDisk = ""
19      diskPersistent = ""
20      timeToIdleSeconds = ""
21      timeToLiveSeconds = ""
22      memoryStoreEvictionPolicy = "" />
23 </ehcache >
```

上述配置信息中,第 3 行代码的< diskStore >元素用于指定一个文件目录。当 EhCache 把数据写到硬盘上时,会把数据写到该文件目录下。第 16 行的< cache >元素可以设置自定义的配置信息。第 4 行代码中的< defaultCache >元素包含了一些属性用于定义配置信息,具体如表 5-1 所示。

表 5-1 < defaultCache >元素的属性

属性名称	说明
maxElementsInMemory	指定内存中最大缓存对象数
maxElementsOnDisk	指定磁盘中最大缓存对象数
eternal	指定缓存的 elements 是否永远不过期
overflowToDisk	指定当内存缓存溢出的时候是否将过期的 element 缓存到磁盘上
timeToIdleSeconds	指定 EhCache 中的数据前后两次被访问的时间间隔
timeToLiveSeconds	指定缓存 element 的有效生命期
diskPersistent	在 VM 重启时是否启用磁盘保存 EhCache 中的数据

表 5-1 列举出了 EhCache 配置文件中< defaultCache >元素的属性，开发人员可根据需要调整 EhCache 缓存的具体功能。

5.2.2 EhCache 的下载

（1）访问 EhCache 官网或者 EhCache 的 GitHub 网址，根据实际场景选择对应版本号进行下载。EhCache 官网下载页面如图 5-5 所示。

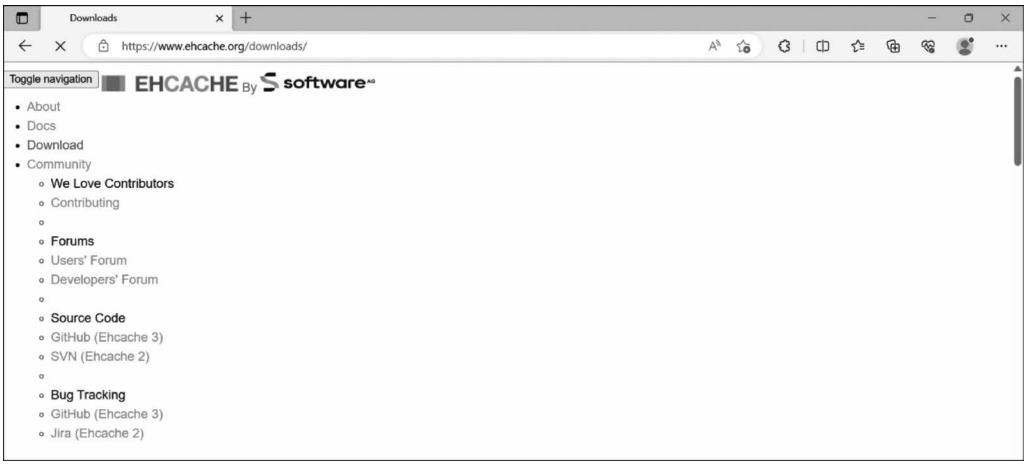

图 5-5　EhCache 官网下载页面

（2）首先将下载好的压缩包文件解压，然后打开解压后的文件夹，找到 mybatis-ehcache-1.0.3.jar 文件，最后打开 lib 目录，找到 ehcache-core-2.6.8.jar 文件。这两个文件即为 MyBatis 整合 EhCache 缓存所需的 JAR 包。EhCache 的 JAR 包目录如图 5-6 所示。

图 5-6　EhCache 的 JAR 包目录

5.2.3 MyBatis 整合 EhCache 缓存

MyBatis 整合 EhCache 缓存的具体步骤如下所示。

1. 引入 EhCache 的 JAR 包

将 mybatis-ehcache-1.0.3.jar 文件和 lib 目录下的 ehcache-core-2.6.8.jar 文件复制到新项目 chapter05 的 lib 目录下，完成 JAR 包的导入。

2. 配置 EhCache 的 type 属性

在 Education.xml 映射文件中，配置< cache >标签的 type 属性，具体代码如下所示。

```
< cache type = "org.mybatis.caches.ehcache.EhcacheCache"></cache >
```

3. 配置 EhCache

在 chapter05 项目的 resource 目录下新建 EhCache 的配置文件 ehcache.xml,具体代码如例 5-6 所示。

例 5-6 ehcache.xml。

```
<?xml version = "1.0" encoding = "UTF - 8"?>
< ehcache xmlns:xsi = "http://www.w3.org/2001/XMLSchema - instance"
          xsi:noNamespaceSchemaLocation = "../config/ehcache.xsd">
    < diskStore path = "D:/ehcache/"/>
    < defaultCache
            maxElementsInMemory = "10"
            maxElementsOnDisk = "100000000"
            eternal = "false"
            overflowToDisk = "true"
            diskPersistent = "false"
            timeToIdleSeconds = "120"
            timeToLiveSeconds = "120"
            diskExpiryThreadIntervalSeconds = "120"
            memoryStoreEvictionPolicy = "LRU"/>
</ehcache >
```

在例 5-6 中,< diskStore >的 path 属性设置缓存地址为 D 盘下的 ehcache 目录,maxElementsInMemory 指定在内存中缓存元素的最大数目为 10;因为本例要演示缓存溢出后数据会存入磁盘的效果,所以此处的值需要设置得偏低,在实际开发中需要根据具体需求设置;overflowToDisk 属性值为 true,当内存缓存溢出时,过期的 element 会被缓存到磁盘上。

4. 执行结果

执行 TestCache03 类的 main()方法,EhCache 缓存执行结果如图 5-7 所示。

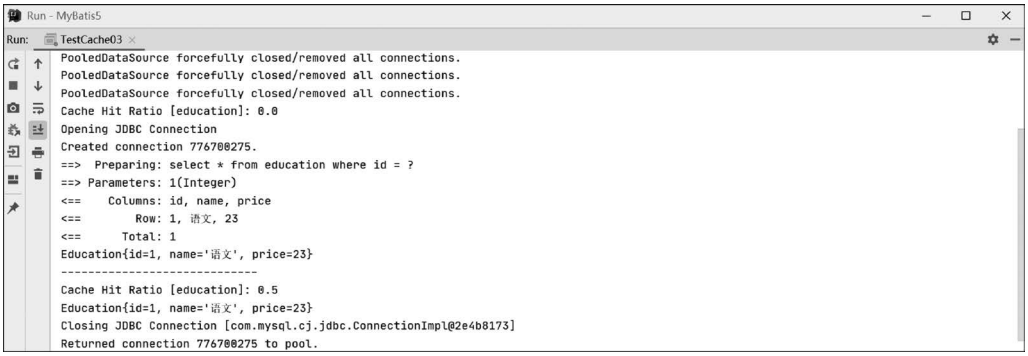

图 5-7 EhCache 缓存执行结果

从图 5-7 的执行结果可以看出,EhCache 缓存和使用 MyBatis 默认的二级缓存结果相同,当第 2 个 SqlSession 对象执行相同的查询 SQL 语句时,命中率(Cache Hit Ratio)为 0.5,程序没有发出 SQL 语句,这就说明,程序从 EhCache 缓存中获取了数据。

打开 D 盘,可以发现 D 盘中出现了 ehcache 目录,打开 D:\ehcache 目录,该目录中存有 EhCache 缓存信息,如图 5-8 所示。

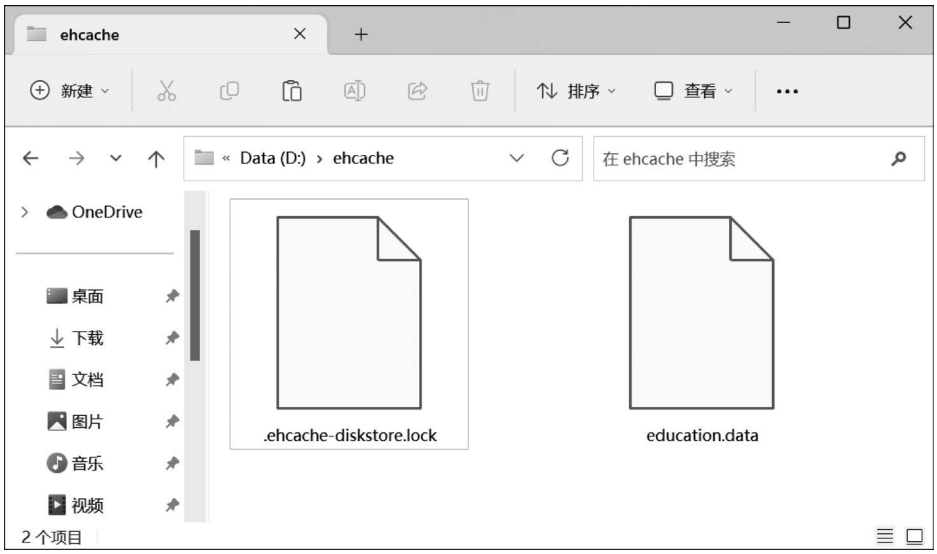

图 5-8　EhCache 缓存信息

5.3　本 章 小 结

本章首先介绍了 MyBatis 的一级缓存和二级缓存的特点及使用方法，然后对 EhCache 缓存概念、EhCache 缓存的下载和 MyBatis 整合 EhCache 缓存进行了讲解。通过本章的学习，可以使读者更好地理解和应用 MyBatis 的缓存机制以及 MyBatis 与 EhCache 缓存的整合使用，从而提高应用的性能和稳定性。

5.4　习　　题

一、填空题

1. MyBatis 的缓存分为_____和_____。
2. 一级缓存是_____级别的缓存，二级缓存是_____级别的缓存。
3. MyBatis 缓存中，一级缓存默认是开启的，二级缓存默认是_____的。
4. 当数据库发生修改、删除或新增时，MyBatis 缓存会_____。
5. 当执行两次相同的查询语句时，MyBatis 会首先从_____中读取查询结果。

二、选择题

1. 关于 MyBatis 一级缓存的描述，下列选项错误的是(　　)。

A. 当程序对数据库执行了 DML 操作，MyBatis 会删除一级缓存中的相关内容

B. 当一级缓存开启时，如果 SqlSession 第一次执行某查询语句，MyBatis 会写入一级缓存

C. 当一级缓存开启时，如果 SqlSession 第二次执行某查询语句，MyBatis 一定可以从一级缓存中读取数据

D. MyBatis 一级缓存的作用域是 SqlSession

2. 关于 MyBatis 的二级缓存,下列选项错误的是(　　)。

A. MyBatis 二级缓存的作用域是跨 Mapper 的

B. 在使用二级缓存时,MyBatis 以 namespace 区分 Mapper

C. MyBatis 二级缓存默认是开启的

D. 如果 MyBatis 二级缓存为可读写缓存,则被操作的 POJO 类需实现序列

3. 在< cache >元素的属性中,用于指定收回策略的是(　　)。

A. Eviction　　B. flushInterval

C. size　　D. readOnly

4. 关于 MyBatis 整合 EhCache 缓存,下列说法错误的是(　　)。

A. MyBatis 的二级缓存可以自定义缓存源

B. EhCache 是一种获得广泛应用的开源分布式缓存框架

C. MyBatis 自身可以实现分布式缓存

D. EhCache 通常采用名称为 ehcache. xml 的配置文件实现功能配置

5. 在 EhCache 缓存的配置文件中,下列用于指定磁盘存储的是(　　)。

A. < diskStore >　　B. < defaultCache >

C. < cache >　　D. < ehcache >

三、简答题

1. 请简述 MyBatis 一级缓存的概念和特点。

2. 请简述 MyBatis 二级缓存的概念和特点。

四、操作题

通过 MyBatis 整合 EhCache 缓存的方法,查询表 education 中所有教材的信息,将查询结果存入 EhCache 缓存中。