(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109960709 A
(43)申请公布日 2019.07.02

(21)申请号 201910312278.5

(22)申请日 2019.04.18

(71)申请人 上海达梦数据库有限公司
地址 201203 上海市浦东新区博霞路50号403室

(72)发明人 杨超 赵侃 郑靖博

(74)专利代理机构 北京品源专利代理有限公司 11332
代理人 孟金喆

(51)Int.Cl.
G06F 16/25(2019.01)

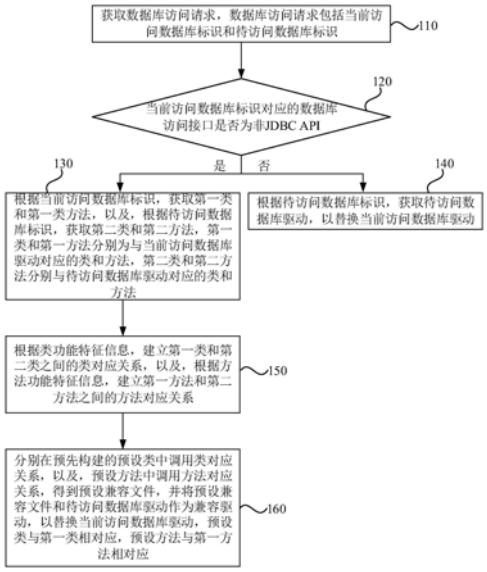
权利要求书3页 说明书20页 附图6页

(54)发明名称

一种数据库驱动的处理方法、装置、设备及存储介质

(57)摘要

本发明实施例公开了一种数据库驱动的处理方法、装置、设备及存储介质，获取数据库访问请求；如果与当前访问数据库标识对应的数据库访问接口为非JDBC API，则根据当前访问数据库标识，获取第一类和第一方法，根据待访问数据库标识，获取第二类和第二方法；根据类功能特征信息，建立第一类和第二类之间的类对应关系，根据方法功能特征信息，建立第一方法和第二方法之间的方法对应关系；在预先构建的预设类中调用类对应关系，预设方法中调用方法对应关系，得到预设兼容文件，并将预设兼容文件和待访问数据库驱动作为兼容驱动，以替换当前访问数据库驱动，预设类与第一类相对应，预设方法与第一方法相对应。本发明实施例简化了切换操作的流程。



CN 109960709 A

1.一种数据库驱动的处理方法,其特征在于,包括:

获取数据库访问请求,所述数据库访问请求包括当前访问数据库标识和待访问数据库标识;

如果与所述当前访问数据库标识对应的数据库访问接口为非Java数据库连接JDBC API,则根据所述当前访问数据库标识,获取第一类和第一方法,以及,根据所述待访问数据库标识,获取第二类和第二方法,所述第一类和所述第一方法分别为与当前访问数据库驱动对应的类和方法,所述第二类和所述第二方法分别为与待访问数据库驱动对应的类和方法;

根据类功能特征信息,建立所述第一类和所述第二类之间的类对应关系,以及,根据方法功能特征信息,建立所述第一方法和所述第二方法之间的方法对应关系;

分别在预先构建的预设类中调用所述类对应关系,以及,预设方法中调用所述方法对应关系,得到预设兼容文件,并将所述预设兼容文件和所述待访问数据库驱动作为兼容驱动,以替换所述当前访问数据库驱动,所述预设类与所述第一类相对应,所述预设方法与所述第一方法相对应。

2.根据权利要求1所述的方法,其特征在于,所述根据类功能特征信息,建立所述第一类和所述第二类之间的类对应关系,包括:

如果第一类对应的类名与第二类对应的类名一致,则建立所述第一类和所述第二类之间的类对应关系。

3.根据权利要求1所述的方法,其特征在于,所述根据类功能特征信息,建立所述第一类和所述第二类之间的类对应关系,包括:

如果第一类对应的JDBC标准信息与第二类对应的JDBC标准信息一致,则建立所述第一类和所述第二类之间的类对应关系。

4.根据权利要求1所述的方法,其特征在于,所述根据方法功能特征信息,建立所述第一方法和所述第二方法之间的方法对应关系,包括:

根据方法名,建立所述第一方法和所述第二方法之间的一对一的方法对应关系;

根据方法参数,建立所述第一方法和所述第二方法之间的一对一的方法对应关系或一对多的方法对应关系。

5.根据权利要求4所述的方法,其特征在于,所述根据方法名,建立所述第一方法和所述第二方法之间的一对一的方法对应关系,包括:

如果第一方法对应的方法名和第二方法对应的方法名一致,则建立所述第一方法和所述第二方法之间的一对一的方法对应关系。

6.根据权利要求4或5所述的方法,其特征在于,所述根据方法参数,建立所述第一方法和所述第二方法之间的一对一的方法对应关系或一对多的方法对应关系,包括:

如果第一方法对应的输入参数和第二方法对应的输入参数一致且第一方法对应的输出参数和第二方法对应的输出参数一致,则建立所述第一方法和所述第二方法之间的一对一的方法对应关系;

如果第一方法对应的输入参数和第二方法对应的输入参数之间满足包含关系或等价关系,以及,第一方法对应的输出参数和第二方法对应的输出参数一致,则建立所述第一方法和所述第二方法之间的一对一的方法对应关系;

如果根据各第一方法对应的输出参数得到的目标输出参数与第二方法对应的输出参数一致，则建立所述第二方法与各第一方法之间的一对多的方法对应关系；

如果根据各第二方法对应的输出参数得到的目标输出参数与第一方法对应的输出参数一致，则建立所述第一方法与各第二方法之间的一对多的方法对应关系。

7.根据权利要求1-5任一所述的方法，其特征在于，所述根据所述当前访问数据库标识，获取第一类和第一方法，以及，根据所述待访问数据库标识，获取第二类和第二方法，包括：

根据所述当前访问数据库标识对应的文档信息，获取第一类和第一方法；

根据所述待访问数据库标识对应的文档信息，获取第二类和第二方法。

8.根据权利要求1-5任一所述的方法，其特征在于，所述如果与所述当前访问数据库标识对应的数据库访问接口为非Java数据库连接JDBC API，则根据所述当前访问数据库标识，获取第一类和第一方法，以及，根据所述待访问数据库标识，获取第二类和第二方法，包括：

如果与所述当前访问数据库标识对应的数据库访问接口的接口名为非Java数据库连接JDBC API接口名或与所述当前访问数据库标识对应的数据库访问接口的类名为非JDBC API类名，则根据所述当前访问数据库标识，获取第一类和第一方法，以及，根据所述待访问数据库标识，获取第二类和第二方法。

9.根据权利要求1-5任一所述的方法，其特征在于，所述将所述预设兼容文件和所述待访问数据库驱动作为兼容驱动，以替换所述当前访问数据库驱动之后，还包括：

在所述兼容驱动下，调用与所述当前访问数据库标识对应的数据库访问接口，访问待访问数据库。

10.一种数据库驱动的处理装置，其特征在于，包括：

数据库访问请求获取模块，用于获取数据库访问请求，所述数据库访问请求包括当前访问数据库标识和待访问数据库标识；

类和方法获取模块，用于如果与所述当前访问数据库标识对应的数据库访问接口为非Java数据库连接JDBC API，则根据所述当前访问数据库标识，获取第一类和第一方法，以及，根据所述待访问数据库标识，获取第二类和第二方法，所述第一类和所述第一方法分别为与当前访问数据库驱动对应的类和方法，所述第二类和所述第二方法分别为与待访问数据库驱动对应的类和方法；

关系建立模块，用于根据类功能特征信息，建立所述第一类和所述第二类之间的类对应关系，以及，根据方法功能特征信息，建立所述第一方法和所述第二方法之间的方法对应关系；

兼容驱动模块，用于分别在预先构建的预设类中调用所述类对应关系，以及，预设方法中调用所述方法对应关系，得到预设兼容文件，并将所述预设兼容文件和所述待访问数据库驱动作为兼容驱动，以替换所述当前访问数据库驱动，所述预设类与所述第一类相对应，所述预设方法与所述第一方法相对应。

11.一种设备，其特征在于，包括：

一个或多个处理器；

存储器，用于存储一个或多个程序；

当所述一个或多个程序被所述一个或多个处理器执行，使得所述一个或多个处理器实现如权利要求1-9任一所述的方法。

12.一种计算机可读存储介质，其上存储有计算机程序，其特征在于，该程序被处理器执行时实现如权利要求1-9任一所述的方法。

一种数据库驱动的处理方法、装置、设备及存储介质

技术领域

[0001] 本发明实施例涉及数据库技术，尤其涉及一种数据库驱动的处理方法、装置、设备及存储介质。

背景技术

[0002] 目前开发应用程序，越来越多的需要使用数据库，例如，B/S应用程序和C/S应用程序。尤其对于B/S应用程序(即浏览器/服务器结构)来说，这是由于现在的网页基本上都是动态网页，动态网页就意味着页面的信息是要经常发生变化的，而存储上述信息的载体主要就是数据库。上述要求掌握使用应用程序操作数据库，以及，操作数据库中数据的方法。在Java应用程序中，通常是使用JDBC(Java Database Base Connectivity，Java数据库连接)连接和操作数据库。JDBC是一套用于执行SQL语句的Java API(Application Programming Interface，应用程序接口)，它由一组用Java语言编写的类和接口组成。

[0003] 现有技术中，当需要由访问当前访问数据库切换到访问待访问数据库，并且当前访问数据库的数据库接口为非JDBC API时，则通常采用如下方式进行切换：将当前访问数据库驱动替换成待访问数据库驱动，并修改JDBC应用程序中的相关程序代码。

[0004] 然而，发明人发现现有技术中至少存在如下问题：由于在切换过程中需要修改JDBC应用程序中的程序代码，即无法实现JDBC应用程序中的程序代码无缝迁移，因此，导致切换数据库操作流程繁琐，进而影响了切换操作效率。

发明内容

[0005] 本发明实施例提供一种数据库驱动的处理方法、装置、设备及存储介质，以简化切换数据库操作流程，进而提高切换操作效率。

[0006] 第一方面，本发明实施例提供了一种数据库驱动的处理方法，该方法包括：

[0007] 获取数据库访问请求，所述数据库访问请求包括当前访问数据库标识和待访问数据库标识；

[0008] 如果与所述当前访问数据库标识对应的数据库访问接口为非Java数据库连接JDBC API，则根据所述当前访问数据库标识，获取第一类和第一方法，以及，根据所述待访问数据库标识，获取第二类和第二方法，所述第一类和所述第一方法分别为与当前访问数据库驱动对应的类和方法，所述第二类和所述第二方法分别为与待访问数据驱动对应的类和方法；

[0009] 根据类功能特征信息，建立所述第一类和所述第二类之间的类对应关系，以及，根据方法功能特征信息，建立所述第一方法和所述第二方法之间的方法对应关系；

[0010] 分别在预先构建的预设类中调用所述类对应关系，以及，预设方法中调用所述方法对应关系，得到预设兼容文件，并将所述预设兼容文件和所述待访问数据库驱动作为兼容驱动，以替换所述当前访问书库驱动，所述预设类与所述第一类相对应，所述预设方法与所述第一方法相对应。

[0011] 第二方面，本发明实施例还提供了一种数据库驱动的处理装置，该装置包括：

[0012] 数据库访问请求获取模块，用于获取数据库访问请求，所述数据库访问请求包括当前访问数据库标识和待访问数据库标识；

[0013] 类和方法获取模块，用于如果与所述当前访问数据库标识对应的数据库访问接口为非Java数据库连接JDBC API，则根据所述当前访问数据库标识，获取第一类和第一方法，以及，根据所述待访问数据库标识，获取第二类和第二方法，所述第一类和所述第一方法分别为与当前访问数据库驱动对应的类和方法，所述第二类和所述第二方法分别为与待访问数据库驱动对应的类和方法；

[0014] 关系建立模块，用于根据类功能特征信息，建立所述第一类和所述第二类之间的类对应关系，以及，根据方法功能特征信息，建立所述第一方法和所述第二方法之间的方法对应关系；

[0015] 兼容驱动模块，用于分别在预先构建的预设类中调用所述类对应关系，以及，预设方法中调用所述方法对应关系，得到预设兼容文件，并将所述与兼容文件和所述待访问数据库驱动作为兼容驱动，以替换所述当前访问数据库驱动，所述预设类与所述第一类相对应，所述预设方法与所述第一方法相对应。

[0016] 第三方面，本发明实施例还提供了一种设备，该设备包括：

[0017] 一个或多个处理器；

[0018] 存储器，用于存储一个或多个程序；

[0019] 当所述一个或多个程序被所述一个或多个处理器执行，使得所述一个或多个处理器实现如本发明实施例第一方面所述的方法。

[0020] 第四方面，本发明实施例还提供了一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现如本发明实施例第一方面所述的方法。

[0021] 本发明实施例通过获取数据库访问请求，数据库访问请求包括当前访问数据库标识和待访问数据库标识，如果与当前访问数据库标识对应的数据库访问接口为非JDBC API，则根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法，根据类功能特征类功能特征信息，建立第一类和第二类之间的类对应关系，以及，根据方法功能特征方法功能特征信息，建立第一方法和第二方法之间的方法对应关系，分别在预先构建的预设类中调用类对应关系，以及，预设方法中调用方法对应关系，得到预设兼容文件，并将预设兼容文件和待访问数据库驱动作为兼容驱动，以替换当前访问数据库驱动。上述通过构建的预设兼容文件建立起了当前访问数据库与待访问数据库之间的关联关系，基于预设兼容文件实现了无需修改JDBC应用程序的相关程序代码，即实现了JDBC应用程序中的程序代码的无缝迁移，便可由访问当前访问数据库切换到访问待访问数据库，简化了切换操作的流程，进而提高了切换操作的效率。

附图说明

[0022] 图1是传统技术中的一种数据库驱动的处理方法的应用示意图；

[0023] 图2是传统技术中的另一种数据库驱动的处理方法的应用示意图；

[0024] 图3是本发明实施例中的一种数据库驱动的处理方法的流程图；

[0025] 图4是本发明实施例中的一种数据库驱动的处理方法的应用示意图；

[0026] 图5是本发明实施例中的一种数据库驱动的处理装置的结构示意图；

[0027] 图6是本发明实施例中的一种设备的结构示意图。

具体实施方式

[0028] 下面结合附图和实施例对本发明作进一步的详细说明。可以理解的是，此处所描述的具体实施例仅仅用于解释本发明，而非对本发明的限定，实施例中记载的各个特征可进行组合，形成多个可选方案。另外还需要说明的是，为了便于描述，附图中仅示出了与本发明相关的部分而非全部结构。

[0029] 实施例

[0030] JDBC是一套用于执行SQL语句的Java API(Application Programming Interface，应用程序接口)，它由一组用Java语言编写的类和接口组成。利用它提供的类和接口，可以方便地使Java应用程序与不同数据库(如MySQL、Oracle、DB2和SQL Server等)建立连接，向数据库发送SQL语句。此外，JDBC还可以封装数据库返回的查询结果，上述实现了对数据库数据的存取操作。JDBC为应用程序开发人员提供了一个标准的API，使得应用程序开发人员在编程时可以不用绑定特定数据库厂商的API，在此基础上，可以以构建更高级的工具和接口，应用程序开发人员能够使用纯Java API编写数据库应用程序。

[0031] JDBC包括如下四个层次：Java应用程序层、JDBC API层、JDBC驱动层和数据库层。其中，Java应用程序层为应用程序开发人员进行程序开发的层次。JDBC API是由SUN公司(现在是Oracle公司)提出的访问数据库的接口标准(或接口规范)，具体接口标准指的是Java应用程序与各数据库之间数据库连接的标准。上述JDBC API由不同数据库厂商来实现，即各数据库厂商均可按照这个接口标准去开发能访问其数据库的类库。当然，既然按照接口标准去开发，那么各数据库厂商就需要去实现这些接口。JDBC驱动层是由数据库厂商基于JDBC API开发数据库JDBC驱动程序以及数据库的类和方法。JDBC驱动程序负责将标准JDBC调用转换成具体数据库的调用。即JDBC驱动程序可以起到如下作用：其一、建立与数据库的连接；其二、将查询和更新语句发送给数据库；其三、处理结果。上述表明，JDBC采用JDBC API和JDBC驱动程序分离的思想设计了Java编程的框架。JDBC将Java应用程序与具体的数据库隔离，大大简化了应用程序开发过程，提高了可移植性。

[0032] JDBC API为Java应用程序使用数据库提供了统一的编程接口，它由一组Java接口和类组成。这些接口和类位于JDK的两个包中，即java.sql包和javax.sql包。其中，javax.sql包称为扩展包，javax.sql包中包括一些关于数据库、连接池和其它的扩展的接口和类，而主要的接口和类均位于java.sql包中。java.sql包中所包括的类和接口主要对基本的数据库编程服务，如生产连接、执行SQL语句、准确SQL语句以及运行批处理查询等。此外，也有一些高级的处理，如批处理更新、事务隔离和可滚动结果集等。其中一些接口由驱动程序提供商(即数据库厂商)来实现。由于在企业级java应用中进行的数据库操作远远不止数据库的连接并执行语句，还需要考虑其它方面的要求，如使用连接池来优化资源的使用，实现分布式事务处理。上述所述的javax.sql为连接管理和分布式事务提供了更好的抽象，引入了容器管理的连接池、分布式事务和行集。

[0033] JDBC API中最常用的几个接口和类如下：DriverManager类、Driver接口、Connection接口、Statement接口和ResultSet接口。其中，DriverManager类可用于负责加

载各种不同驱动程序,并根据不同的请求,向调用者返回相应的数据库连接。其定义了三个连接数据库的方法,分别为getConnection()、getDriver()/getDrivers()和deregisterDriver()/registerDriver()。Driver接口可表示驱动程序,用于将自身加载到DriverManager中,处理相应的请求并返回相应的数据库连接。Connection接口可实现对指定数据库的连接。Statement可表示一个特定的容器,用以对一个指定数据库执行SQL语句或存储过程语句对应的对象。ResultSet接口可表示返回的结果集对象,用于控制对一个指定语句的行数据的存取。

[0034] JDBC API按照如下步骤连接和操作数据库,具体的:步骤A、注册JDBC驱动程序。即将待访问数据库的JDBC驱动程序加载到Java虚拟机中。具体可通过Class.for.Name()方法将JDBC驱动程序加载到Java虚拟机中;步骤B、连接数据库。具体可使用DriverManager类的getConnection()方法来获得。同时,需要在参数中给出所要连接的数据库的URL(Uniform Resource Locator,统一资源定位符)、用户名和密码;步骤C、使用SQL语句进行数据库操作;步骤D、关闭数据库连接。数据库操作完成之后需要关闭数据库的连接以释放资源。

[0035] 需要说明的是,由于JDBC驱动程序是实现了JDBC API接口的类,即JDBC驱动程序是访问JDBC API接口的实现类,其需要由各数据库厂商根据JDBC制作得到,因此,在使用JDBC驱动程序连接数据库的时候,需要导入数据库的JDBC驱动程序。当连接不同的数据库,需要导入不同的JDBC驱动程序。而程序开发人员在编写Java应用程序时,基本上不用太关心java.sql包和javax.sql包中的类,只需用JDBC API提供的接口进行编程就可以了,即java.sql包和javax.sql包中的接口。由于这些接口的对象都是通过其它对象来获得的,因此,不需要通过新建来创建它们。这些对象的具体实现都是在JDBC驱动程序中实现的,其是JDBC驱动程序类的对象。即数据库JDBC驱动程序中可实现在JDBC API中所定义的抽象类。

[0036] Java应用程序通过JDBC API访问JDBC驱动程序管理器(即JDBC API中的DriverManager类),JDBC驱动程序管理器通过JDBC驱动程序接口(即JDBC API中的Driver接口)访问不同的JDBC驱动程序,从而实现对不同数据库的访问。由于JDBC使用JDBC API连接访问数据库后,当以后要更换操作系统或者数据库,都不需要修改连接访问数据库的程序代码,因此,JDBC能做到“一次编写,到处运行”,为应用程序开发人员带来了便利。

[0037] 为了便于理解,首先,对涉及的几个基本概念进行说明,具体的:

[0038] 其一、类和方法。类是对所有事物公共的概念进行抽象的描述。类描述了一系列在概念上有相同含义的对象,并为这些对象统一定义了Java编程语言上的属性(或状态)和行为。通常用变量来描述类的属性,用方法来实现类的行为。类通常用于封装多个方法,把有关联的方法集合在一起,便于调用。

[0039] 类的基本结构包括类声明和类体。其中,类声明由关键字和类名组成,或者,类声明由关键字、类名和类的属性组成。类体为该类的对象提供了在生存期内需要的所有代码,其包括如下几部分:构造方法、成员变量和方法的实现(即成员方法)。其中,成员变量和成员方法称为类的成员,构造方法并不是类的成员。

[0040] 方法是对实体行为的描述,实体的行为是实体对外界消息的一种反应,业务是一种功能,即实体在外界消息的触发下,按照一定的指令执行一系列动作,产生一定的结果。实体行为所需的参数及产生的结果是通过方法的参数和返回值描述的,而实体行为的处理

功能则是通过方法的方法体描述的。方法的定义指描述方法的处理过程及其所需的参数，并用一个方法名来标识这个处理过程。方法定义中的参数并没有实际值，仅仅是为了描述处理过程而引入的，因此，称为形式参数(简称形参)。方法的使用就是通过向实体发送消息执行方法所定义的处理功能。使用方法时给出参数的实际值，这些实际值称为实际参数(简称实参)。

[0041] 其二、接口。接口即是抽象方法和常量值组成的集合。其中，抽象方法是指只有方法名和参数而没有方法体的一种方法。接口可以体现Java的封装性，示例性的，如定义了一个类，类的属性是私有的，即外界不能访问，而外界可以通过公有方法来访问这个类。通常提及的一个类的公有方法就是这个类的对外接口，只不过是先把这些属性和方法封装起来，可以通过接口来访问一些功能。接口是一种数据类型，在方法的参数列表和变量的声明中可用接口作数据类型，但不能实例化。上述接口的作用可以体现在如下几个方面，具体的：第一方面、通过接口可以实现不相关类的相同行为，而不需要考虑这些类之间的层次关系；第二方面、通过接口可以声明多个类需要实现的方法；第三方面、通过接口可以了解对象的交互界面，而不需要了解对象所对应的类。接口中的方法均默认为是公共的和抽象的方法，其前默认加public和abstract。接口由接口声明和接口体两部分组成。

[0042] 针对接口声明：接口声明的格式如下：[public]interface接口名[extends listOfSuperInterface]{……}。其中，public指明任意类可以使用这个接口，缺省情况下，只有与该接口定义在同一个包中的类才可以访问这个接口。

[0043] 针对接口体：接口体包括由常量定义的接口体和由方法定义的接口体。如果在子接口中定义了和父接口同名的常量或相同的方法，则父接口中的常量将被隐藏，方法将被重写。

[0044] 基于上述，可以理解到，可将接口理解为一种特殊的类，即是由常量值和抽象方法组成的特殊类。这是由于：如果类中的所有方法均为抽象方法时，则可将该类理解为接口；如果类中的部分方法为抽象方法时，则可将该类理解为抽象类；如果类中的所有方法均为非抽象方法时，则可将该类理解为普通类。由此可见，上述所述的抽象类和接口虽然具有一定的相似性，但也存在区别，主要区别在于：接口是公开的，里面不能有私有的方法和变量，而类是可以有私有方法或私有变量的。实现接口的一定要实现接口里定义的所有方法，而实现抽象类可以有选择地重写需要用到的方法。

[0045] 基于上述，接口的具体实现是提供了类和方法。现可设定JDBC API提供了类A1和方法A1_FUN1，数据库厂商B提供了类B1和方法B1_FUN1，数据库厂商C提供了类C1和方法C1_FUN1。可以理解到，由于类A1和方法A1_FUN1是JDBC API的类以及类的方法，因此，对于全部JDBC应用程序均是通用的。由于类B1和方法B1_FUN1，以及，类C1和方法C1_FUN1分别是对应的数据库厂商自主开发的类和方法，因此，对于全部JDBC应用程序并不是通用的。可将非JDBC(如数据库厂商B和数据库厂商C)提供的数据访问接口称为非JDBC API。

[0046] 上述将导致在不同数据库之间进行切换访问时，当前访问数据库的数据库访问接口是JDBC API与当前访问数据库的数据库访问接口是非JDBC API，形成的是不同的处理流程，具体的：现设定当前访问数据库为Oracle数据库，待访问数据库为MySQL数据库，即现需要由访问Oracle数据库切换到访问MySQL数据库。

[0047] 如果Oracle数据库的数据库访问接口是JDBC API，则为了实现由访问Oracle数据

库切换到访问MySQL数据库，可通过如下方式实现：只需要将Oracle数据库JDBC驱动程序替换成MySQL数据库JDBC驱动程序即可，而无需修改JDBC应用程序的相关程序代码。如图1所示，即给出了传统技术中的一种数据库驱动的处理方法的应用示意图。图1中显示只需将Oracle数据库JDBC驱动程序替换成MySQL数据库JDBC驱动程序即可。

[0048] 如果Oracle数据库的数据库访问接口是非JDBC API，则为了实现由访问Oracle数据库切换到访问MySQL数据库，可通过如下方式实现：不仅将Oracle数据库JDBC驱动程序替换成MySQL数据库JDBC驱动程序，而且还需要修改JDBC应用程序中引用非JDBC API的相关程序代码。这里所述的修改可以是将Oracle数据库JDBC API替换为JDBC API，还可以是将Oracle数据库JDBC API替换为MySQL数据库JDBC API。如图2所示，即给出了传统技术中的一种数据库驱动的处理方法的应用示意图。图2中不仅需要将Oracle数据库JDBC驱动程序替换成MySQL数据库JDBC驱动程序，而且还需要修改JDBC应用程序的相关程序代码。

[0049] 需要说明的是，上述所述的无需修改JDBC应用程序的相关程序代码的方式，实现由访问当前访问数据库切换到访问待访问数据库，体现出的是能够实现JDBC应用程序中的程序代码的无缝迁移。

[0050] 可以理解到，当需要由访问当前访问数据库切换到访问待访问数据库，并且当前访问数据库的数据库接口为非JDBC API时，则传统技术中存在着由于无法实现JDBC应用程序中的程序代码的无缝迁移而导致的切换数据库操作流程繁琐的问题。

[0051] 基于上述，为了解决传统技术中存在的问题，可作如下考虑：不同数据库之间所提供的类和方法通常具有对应关系，所述对应关系体现在位于不同数据库中的类或方法可以实现相似功能，如E数据库中存在类1，F数据库中存在类2，虽然类1和类2位于不同数据库，但类1和类2所实现的功能相似，则可基于两者所实现的功能相似，建立两者之间的对应关系。上述对应关系可以成为关联两个数据库的桥梁。当前访问数据库为E数据库，待访问数据库为F数据库，并且E数据库的数据库访问接口为非JDBC API。为了实现仅需要将E数据库JDBC驱动程序替换成F数据库JDBC驱动程序，而无需修改JDBC应用程序中引用非JDBC API的相关程序代码，便可由访问E数据库切换到访问F数据库，可考虑基于上述所述的对应关系实现，即虽然E数据库驱动调用的是E数据库中的类1，但基于该对应关系，可以实现调用F数据库中的类2。

[0052] 基于上述，解决传统技术中存在的上述问题的关键在于：如何构建上述对应关系。下面将通过具体实施例进行说明。

[0053] 图3为本发明实施例提供的一种数据库驱动的处理方法的流程图，本实施例可适用于简化切换操作流程的情况，该方法可以由数据库驱动的处理装置来执行，该装置可以采用软件和/或硬件的方式实现，该装置可以配置于设备中，例如典型的是服务器等。如图3所示，该方法具体包括如下步骤：

[0054] 步骤110、获取数据库访问请求，数据库访问请求包括当前访问数据库标识和待访问数据库标识。

[0055] 在本发明的实施例中，数据库访问请求可用于作为是否切换数据库的触发条件。数据库访问请求可以包括当前访问数据库标识和待访问数据库标识。其中，当前访问数据库标识可用于标识当前正在访问的数据库。换句话说，当前访问数据库标识可用于作为当前访问数据库的身份标识，以与其它数据库进行区分。待访问数据库标识可用于标识将要

访问的数据库。换句话说，待访问数据库标识可用于作为将要访问的数据库的身份标识，以与其它数据库进行区分。

[0056] 示例性的，如当前访问数据库标识为G，待访问数据库标识为H。G用于标识当前访问数据库，H用于标识待访问数据库。

[0057] 步骤120、当前访问数据库标识对应的数据库访问接口是否为非JDBC API；若是，则执行步骤130；若否，则执行步骤140。

[0058] 在发明的实施例中，当前访问数据库标识对应的数据库为当前访问数据库。确定当前访问数据库的数据库访问接口是否为非JDBC API，如果当前访问数据库的数据库访问接口为非JDBC API，则可以说明无法采用直接将当前访问数据库JDBC驱动程序替换成待访问数据库JDBC驱动程序，而不修改JDBC应用程序中引用非JDBC API的相关程序代码的方式，实现由访问当前访问数据库切换为访问待访问数据库。如果当前访问数据库的数据库访问接口为JDBC API，则可以说明可以采用将当前访问数据库JDBC驱动程序替换成待访问数据库JDBC驱动程序，而无需修改JDBC应用程序中引用非JDBC API的相关程序代码的方式，实现由访问当前访问数据库切换为访问待访问数据库。

[0059] 如果当前访问数据库的数据库访问接口为非JDBC API，则后续需要采用本发明实施例所提供的技术方案，即需要构建兼容驱动，以实现由访问当前访问数据库切换为访问待访问数据库时，只需对JDBC驱动程序进行替换而无需修改JDBC应用程序中引用JDBC API的相关程序代码。即实现与当前访问数据库的访问接口为JDBC API相同的无需修改JDBC应用程序中引用JDBC API的相关程序代码的方式，实现由访问当前访问数据库切换为访问待访问数据库。

[0060] 步骤130、根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法，第一类和第一方法分别为与当前访问数据库驱动对应的类和方法，第二类和所述第二方法分别为与待访问数据库驱动对应的类和方法。并执行步骤150。

[0061] 步骤140、根据待访问数据库标识，获取待访问数据库驱动，以替换当前访问数据库驱动。

[0062] 在本发明的实施例中，如果与当前访问数据库标识对应的数据库访问接口为非JDBC API，即如果当前访问数据库的数据库访问接口为非JDBC API，则可以根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法。其中，第一类是当前访问数据库驱动对应的类，第一方法是当前访问数据库驱动对应的方法。第二类是待访问数据库驱动对应的类，第二方法是待访问数据库驱动对应的方法。通过上述操作，以获得与当前访问数据库驱动对应的类和方法，以及，与待访问数据库驱动对应的类和方法，为后续构建两个数据库驱动的关联关系提供依据。

[0063] 如果与当前访问数据库标识对应的数据库访问接口为非JDBC API，即如果当前访问数据库的数据库访问接口为非JDBC API，则可以通过如下方式确定第一类和第一方法，以及，第二类和第二方法，具体的：通常各数据库厂商在开发数据库时，会撰写相应的使用文档，使用文档中记录有该数据库所提供的类和方法等相关内容。基于上述，如果当前访问数据库的数据库访问接口为非JDBC API，则可根据当前访问数据库标识对应的文档信息，即可根据当前访问数据库对应的文档信息，获取第一类和第一方法。以及，根据待访问数据

库标识对应的文档信息，即可根据待访问数据库对应的文档信息，获取第二类和第二方法。其中，文档信息可以理解为使用文档所提供的相关内容。此外，还可通过查看当前访问数据库对应的程序代码，获取第一类和第一方法。即可通过当前访问数据库提供的相关的程序代码，获取第一类和第一方法。同样的，可通过查看待访问数据库对应的程序代码，获取第二类和第二方法。即可通过待访问数据库提供的相关的程序代码，获取第二类和第二方法。

[0064] 如果与当前访问数据库标识对应的数据库访问接口为JDBC API，则可根据待访问数据库标识，获取待访问数据库驱动，以替换当前访问数据库访问驱动。后续可直接根据待访问数据库驱动，访问待访问数据库。

[0065] 步骤150、根据类功能特征信息，建立第一类和第二类之间的类对应关系，以及，根据方法功能特征信息，建立第一方法和第二方法之间的方法对应关系。

[0066] 在本发明的实施例中，为了构建当前访问数据库和待访问数据库之间的对应关系，可从如下方面考虑，具体的：针对数据库的类来说，如果某个数据库中某个类所实现的功能与另一个数据库中某个类所实现的功能相似，则可以说明这两个类之间可以建立对应关系。换句话说，不同数据库所提供的两个类之间能否建立对应关系的条件是：这两个类所实现的功能是否相似。如果两个类所实现的功能相似，则这两个类之间可以建立对应关系。如果两个类所实现的功能不相似，则这两个类之间无法建立对应关系。基于上述，需要考虑如何确定不同数据库所提供的两个类所实现的功能是否相似。即需要选择确定不同数据库所提供的两个类所实现的功能是否相似的依据。换句话说，该依据可以用于确定不同数据库所提供的两个类所实现的功能是否相似。本发明实施例所提供的技术方案中选择类功能特征信息作为上述所述的依据，即类功能特征信息可用于确定不同数据库所提供的两个类所实现的功能是否相似的依据，也即类功能特征信息可用于作为不同数据库所提供的两个类之间能否建立类对应关系的依据。类所对应的类功能特征信息可以体现该类所实现的功能。具体如下：如果第一类对应的类功能特征信息与第二类对应的类功能特征信息一致，则可以说明第一类所实现的功能与第二类所实现的功能一致，在此情况下，可建立第一类和第二类之间的类对应关系。简而言之，可通过类功能特征信息，建立第一类和第二类之间的类对应关系。类功能特征信息可包括类名和JDBC标准信息。其中。JDBC标准信息可以指由JDBC API所提供的信息。

[0067] 根据类功能特征信息，建立第一类和第二类的类对应关系，可作如下理解：根据类名，建立第一类和第二类的对应关系。或者，根据JDBC标准信息，建立第一类和第二类的对应关系。更为具体的，如果第一类对应的类名与第二类对应的类名一致，则可以说明第一类和第二类所实现的功能相似。基于此，可建立第一类和第二类之间的类对应关系。如果第一类对应的JDBC标准信息与第二类对应的JDBC标准信息一致，则也可以说明第一类和第二类所实现的功能相似。基于此，可建立第一类和第二类之间的类对应关系。可以理解到，第一类和第二类之间的类对应关系是一对一的类对应关系。

[0068] 针对数据库的方法来说，方法对应关系可能存在如下情况：某个数据库中的某个方法所实现的功能与另一个数据库中的某个方法所实现的功能相似，也可能某个数据库中的某几个方法所实现的功能进行结合后得到的功能与另一个数据库中的某个方法所实现的功能相似，还可能某个数据库中的某个方法所实现的功能与另一个数据库中的某几个方法所实现的功能进行结合后得到的功能相似。基于上述，可以理解到，上述所述的某个数据

库中的某几个方法所实现的功能进行结合后得到的功能与另一个数据库中的某个方法所实现的功能相似，某个数据库中的某个方法所实现的功能与另一个数据库中的某几个方法所实现的功能进行结合后得到的功能相似，体现出的并不是某个方法与另一方法之间的一对一的方法对应关系，而是多对一或一对多的方法对应关系。简而言之，方法对应关系可以包括一对一的方法对应关系、多对一的方法对应关系和一对多的方法对应关系。

[0069] 基于上述，如果某个数据库中的方法所实现的功能与另一个数据库中方法所实现的功能相似，则可以说明方法之间可以建立对应关系。换句话说，不同数据库所提供的方法之间能否建立对应关系的条件是：上述方法所实现的功能是否相似。如果不同数据库所提供的方法所实现的功能相似，则上述方法之间可以建立对应关系。如果不同数据库所提供的方法所实现的功能不相似，则上述方法之间无法建立对应关系。基于上述，需要考虑如何确定不同数据库所提供的方法所实现的功能是否相似。即需要选择确定不同数据库所提供的方法所实现的功能是否相似的依据。换句话说，该依据可以用于确定不同数据库所提供的方法所实现的功能是否相似。本发明实施例所提供的技术方案中选择方法功能特征信息作为上述所述的依据，即方法功能特征信息可用于确定不同数据库所提供的方法所实现的功能是否相似的依据，也即方法功能特征信息可用于作为不同数据库所提供的方法之间能否建立方法对应关系的依据。方法所对应的方法功能特征信息可以体现该方法所实现的功能。即可通过方法功能特征信息，建立第一类和第二类之间的方法对应关系。方法功能特征信息可包括方法名和方法参数。其中。方法参数可以包括方法的输入参数和输出参数。

[0070] 根据方法功能特征信息，建立第一方法和第二方法的方法对应关系，可作如下理解：根据方法名，建立第一方法和第二方法之间的一对一的方法对应关系。或者。根据方法参数，建立第一方法和第二方法之间的一对一的方法对应关系或一对多的方法对应关系。更为具体的：

[0071] 如果第一方法对应的方法名和第二方法对应的方法名一致，则可以说明第一方法和第二方法所实现的功能相似。基于此，可建立第一方法和第二方法之间的一对一的方法对应关系。如果第一方法对应的输入参数和第二方法对应的输入参数一致且第一方法对应的输出参数和第二方法对应的输出参数一致，则可以说明第一方法和第二方法所实现的功能相似。基于此，可建立第一方法和第二方法之间的一对一的方法对应关系。如果第一方法对应的输入参数和第二方法对应的输入参数之间满足包含关系或等价关系，以及，第一方法对应的输出参数和第二方法对应的输出参数一致，则可以说明第一方法和第二方法所实现的功能相似。基于此，可建立第一方法和第二方法之间的一对一的方法对应关系。

[0072] 如果根据各第一方法对应的输出参数得到的目标输出参数与第二方法对应的输出参数一致，则可以说明第二方法和各第一方法所实现功能进行结合的功能相似。基于此，可建立第一方法和第二方法之间的多对一的方法对应关系。如果根据各第二方法对应的输出参数得到的目标输出参数与第一方法对应的输出参数一致，则可以说明第一方法和各第二方法所实现功能进行结合的功能相似。基于此，可建立第一方法和第二方法之间的一对多的方法对应关系。

[0073] 上述通过根据类功能特征信息，建立第一类和第二类之间的类对应关系，以及，根据方法功能特征信息，建立第一方法和第二方法之间的方法对应关系，为后续生产兼容驱动提供依据。

[0074] 步骤160、分别在预先构建的预设类中调用类对应关系，以及，预设方法中调用方法对应关系，得到预设兼容文件，并将预设兼容文件和待访问数据库驱动作为兼容驱动，以替换当前访问数据库访问驱动，预设类与第一类相对应，预设方法与第一方法相对应。

[0075] 在本发明的实施例中，为了得到兼容驱动，可构建预设兼容文件，在得到预设兼容文件后，将预设兼容文件和待访问数据库驱动共同作为兼容驱动，以替换当前访问数据库驱动。即通过预设兼容文件构建起当前访问数据库和待访问数据库之间的对应关系。

[0076] 可预先构建预设类和预设方法，其中，预设类与第一类相对应，预设方法与第一方法相对应。所谓构建预设类和预设方法，可作如下理解：构建与第一类相对应的预设类，预设类的基础信息与第一类的基础信息相同，所谓基础信息可理解为类声明。构建与第一方法相对应的预设方法，预设方法的基础信息与第一方法的基础信息相同，所述基础信息可理解为方法声明。

[0077] 在预先构建的预设类中调用类对应关系，以及，在预先构建的预设方法中调用方法对应关系，得到预设兼容文件，可作如下理解：将类对应关系写入预先构建的预设类中，将方法对应关系写入预先构建的预设方法中，生成预设兼容文件。更为具体的，根据第一类的类名，将类对应关系写入预先构建的与第一类的类名一致的预设类中。根据第一方法的方法名，将方法对应关系写入预先构建的与第一方法的方法名一致的预设方法中，生成预设兼容文件。

[0078] 需要说明的是，根据前文所述可知，由于类是一组具有相同属性的实体对象抽象而来的，类是方法的一个功能模块，把方法囊括其中，类中有属性和方法，因此，存在多个方法属于同一类的情况。基于上述，存在多个第一方法属于同一第一类的情况。由于预设类与第一类相对应，预设方法与第一方法相对应，因此，存在多个预设方法属于同一预设类的情况。

[0079] 在得到预设兼容文件后，可将预设兼容文件与待访问数据库驱动共同作为兼容驱动，以替换当前访问数据库驱动。后续可在兼容驱动下，调用与当前访问数据库标识对应的数据库访问接口，访问待访问数据库。上述通过将预设兼容文件与待访问数据库驱动共同作为兼容驱动替换当前访问数据库驱动，解决了后续在访问待访问数据库时，不必考虑JDBC应用程序对当前访问数据库驱动的调用问题。

[0080] 示例性的，如当前待访问数据库为Oracle数据库，待访问数据库为DB2数据库。Oracle数据库的数据库接口为非JDBC API。通过上述步骤110-步骤160得到预设兼容文件。其中，Oracle数据库中的类OracleDriver和DB2数据库中的类DB2Driver相对应，Oracle数据库中的方法OracleDriver和DB2数据库中的方法DB2Driver相对应。

[0081] 基于上述，Oracle数据库驱动中DriverManager.registerDriver(new OracleDriver())所实现的功能，DB2数据库驱动中DriverManager.registerDriver(new DB2Driver())所实现的功能，以及，预设兼容文件中DriverManager.registerDriver(new OracleDriver())，三者所实现的功能均相同。可以理解到，预设兼容文件实现了在Oracle数据库驱动中增加了DB2数据库驱动的实例化。后续Oracle数据库驱动中方法OracleDriver的实现就是对DB2数据库驱动中方法OracleDriver的调用。上述实现了虽然Oracle数据库驱动程序中调用的是方法OracleDriver，但是预设兼容文件实际上调用的是DB2Driver。从而实现了无需修改JDBC应用程序中引用非JDBC API的相关程序代码的，便可

实现对不同数据库的访问。

[0082] 需要说明的是,本发明实施例所述的数据库驱动即指数据库JDBC驱动程序。

[0083] 当需要由访问当前访问数据库切换到访问待访问数据库,并且当前访问数据库的数据库接口为非JDBC API时,本发明实施例所提供的技术方案通过构建预设兼容文件,建立起当前访问数据库和待访问数据库之间的关联关系,实现虽然当前访问数据库驱动调用的是当前访问数据库中的类或方法,但基于预设兼容文件,可以实现调用待访问数据库中的对应的类或方法,上述实现过程只需将当前访问数据库驱动替换成待访问数据库驱动即可,实现了JDBC应用程序中的程序代码的无缝迁移,简化了切换操作的流程,进而提高了切换操作的效率。

[0084] 本实施例的技术方案,通过获取数据库访问请求,数据库访问请求包括当前访问数据库标识和待访问数据库标识,如果与当前访问数据库标识对应的数据库访问接口为非JDBC API,则根据当前访问数据库标识,获取第一类和第一方法,以及,根据待访问数据库标识,获取第二类和第二方法,根据类功能特征类功能特征信息,建立第一类和第二类之间的类对应关系,以及,根据方法功能特征方法功能特征信息,建立第一方法和第二方法之间的方法对应关系,分别在预先构建的预设类中调用类对应关系,以及,预设方法中调用方法对应关系,得到预设兼容文件,并将预设兼容文件和待访问数据库驱动作为兼容驱动,以替换当前访问数据库驱动。上述通过构建的预设兼容文件建立起了当前访问数据库与待访问数据库之间的关联关系,基于预设兼容文件实现了无需修改JDBC应用程序的相关程序代码,即实现了JDBC应用程序中的程序代码的无缝迁移,便可由访问当前访问数据库切换到访问待访问数据库,简化了切换操作的流程,进而提高了切换操作的效率。

[0085] 可选的,在上述技术方案的基础上,根据类功能特征信息,建立第一类和第二类之间的类对应关系,具体可以包括:如果第一类对应的类名与第二类对应的类名一致,则建立第一类和第二类之间的类对应关系。

[0086] 在本发明的实施例中,如果存在两个类的类名相同,则可以说明这两个类所实现的功能相似。基于上述,可根据类名,建立第一类和第二类之间的类对应关系。即如果第一类对应的类名与第二类对应的类名一致,则可以说明第一类和第二类所实现的功能相似,基于此,可建立第一类和第二类之间的类对应关系。可以理解到,建立类对应关系的两个类所实现的功能相似。

[0087] 示例性的,如现存在数据库I和数据库J,数据库I中存在类名为K的第一类,数据库J中存在类名为K的第二类。由于第一类对应的类名为K,第二类对应的类名为K,第一类对应的类名与第二类对应的类名一致,则可建立第一类和第二类之间的类对应关系。

[0088] 可选的,在上述技术方案的基础上,根据类功能特征信息,建立第一类和第二类之间的类对应关系,具体可以包括:如果第一类对应的JDBC标准信息与第二类对应的JDBC标准信息一致,则建立第一类和第二类之间的类对应关系。

[0089] 在本发明的实施例中,对于JDBC标准信息,可作如下理解:JDBC标准信息可以指JDBC API所设定的相关信息,各数据库厂商需要根据该相关信息进行设计。由于各数据厂商需要根据该相关信息进行设计,因此,各数据库厂商所设定的类的JDBC标准信息即是JDBC API所设定的相关信息。对于可实现同一功能的位于不同数据库中的类来说,上述类所对应的JDBC标准信息应当相同。

[0090] 基于上述，如果存在两个类的JDBC标准信息相同，两个类分属于不同的数据库，则可以说明这两个类所实现的功能相似。基于此，可根据JDBC标准信息，建立第一类和第二类之间的类对应关系。即如果第一类对应的JDBC标准信息与第二类对应的JDBC标准信息一致，则可以说明第一类和第二类所实现的功能相似，基于此，可建立第一类和第二类之间的类对应关系。可以理解到，建立类对应关系的两个类所实现的功能相似。

[0091] 示例性的，如现存在数据库I和数据库J，数据库I中存在JDBC标准信息为M的第一类，数据库J中存在JDBC标准信息为M的第二类。由于第一类对应的JDBC标准信息为M，第二类对应的JDBC标准信息为N，第一类对应的JDBC标准信息与第二类对应的JDBC标准信息一致，则可建立第一类和第二类之间的类对应关系。

[0092] 可选的，在上述技术方案的基础上，根据方法功能特征信息，建立第一方法和第二方法之间的方法对应关系，具体可以包括：根据方法名，建立第一方法和第二方法之间的一对一的方法对应关系。根据方法参数，建立第一方法和所述第二方法之间的一对一的方法对应关系或一对多的方法对应关系。

[0093] 在本发明的实施例中，如果某个数据库中的方法所实现的功能与另一个数据库中方法所实现的功能相似，则可以说明方法之间可以建立对应关系。换句话说，不同数据库所提供的方法之间能否建立对应关系的条件是：上述方法所实现的功能是否相似。如果不同数据库所提供的方法所实现的功能相似，则上述方法之间可以建立对应关系。如果不同数据库所提供的方法所实现的功能不相似，则上述方法之间无法建立对应关系。基于上述，需要考虑如何确定不同数据库所提供的方法所实现的功能是否相似。即需要选择确定不同数据库所提供的方法所实现的功能是否相似的依据。换句话说，该依据可以用于确定不同数据库所提供的方法所实现的功能是否相似。本发明实施例所提供的技术方案中选择方法功能特征信息作为上述所述的依据，即方法功能特征信息可用于确定不同数据库所提供的方法所实现的功能是否相似的依据，也即方法功能特征信息可用于作为不同数据库所提供的方法之间能否建立方法对应关系的依据。方法所对应的方法功能特征信息可以体现该方法所实现的功能。即可通过方法功能特征信息，建立第一类和第二类之间的方法对应关系。方法功能特征信息可包括方法名和方法参数。其中。方法参数可以包括方法的输入参数和输出参数。

[0094] 根据方法功能特征信息，建立第一方法和第二方法的方法对应关系，可作如下理解：根据方法名，建立第一方法和第二方法之间的一对一的方法对应关系。或者。根据方法参数，建立第一方法和第二方法之间的一对一的方法对应关系或一对多的方法对应关系。

[0095] 上述所述的一对一的方法对应关系和一对多的方法对应关系，可作如下理解：由于方法对应关系可能存在如下情况：某个数据库中的某个方法所实现的功能与另一个数据库中的某个方法所实现的功能相似，也可能某个数据库中的某几个方法所实现的功能进行结合后得到的功能与另一个数据库中的某个方法所实现的功能相似，还可能某个数据库中的某个方法所实现的功能与另一个数据库中的某几个方法所实现的功能进行结合后得到的功能相似。基于上述，可以理解到，上述所述的某个数据库中的某几个方法所实现的功能进行结合后得到的功能与另一个数据库中的某个方法所实现的功能相似，某个数据库中的某个方法所实现的功能与另一个数据库中的某几个方法所实现的功能进行结合后得到的功能相似，体现出的并不是某个方法与另一方法之间的一对一的方法对应关系，而是多对

一或一对多的方法对应关系。因此，方法对应关系可以包括一对一的方法对应关系、多对一的方法对应关系和一对多的方法对应关系。其中，多对一的方法对应关系和一对多的方法对应关系是相对而言的。

[0096] 本发明实施例所述的方法名，可用于确定两个方法是否具有一对一的方法对应关系。所述的方法参数，可用于确定方法的对应关系，对应关系可以包括一对一的对应关系和一对多的对象关系(即多对一的对应关系)。

[0097] 示例性的，如现存在数据库I和数据库J，数据库I中存在方法名为N的第一方法，数据库J中存在方法名为N的第二方法。由于第一方法对应的方法名为N，第二方法对应的方法名为N，第一方法对应的方法名与第二方法对应的方法名一致，则可建立第一方法和第二方法之间的一对一的方法对应关系。

[0098] 又如现存在数据库I和数据库J，数据库I中存在方法参数中输入参数为0，输出参数为P的第一方法，数据库J中存在方法参数中输入参数为0，输出参数为P的第二方法。由于第一方法对应的输入参数为0，输出参数为P，第二方法对应的输入参数为0，输出参数为P，第一方法对应的输入参数与第二方法对应的输入参数一致且第一方法的输出参数与第二方法的输出参数一致，即第一方法对应的方法参数与第二方法对应的方法名一致，则可建立第一方法和第二方法之间的一对一的方法对应关系。

[0099] 可选的，在上述技术方案的基础上，根据方法名，建立第一方法和第二方法之间的一对一的方法对应关系，具体可以包括：如果第一方法对应的方法名和第二方法对应的方法名一致，则建立第一方法和第二方法之间的一对一的方法对应关系。

[0100] 在本发明的实施例中，如果存在两个方法的方法名相同，则可以说明这两个方法所实现的功能相似。基于上述，可根据方法名，建立第一方法和第二方法之间的方法对应关系，且该方法对应关系为一对一的方法对应关系。即如果第一方法对应的方法名与第二类对应的方法名一致，则可以说明第一方法和第二方法所实现的功能相似，基于此，可建立第一方法和第二方法之间的一对一的方法对应关系。可以理解到，建立方法对应关系的两个方法所实现的功能相似。

[0101] 可选的，在上述技术方案的基础上，根据方法参数，建立第一方法和第二方法之间的一对一的方法对应关系或一对多的方法对应关系，具体可以包括：如果第一方法对应的输入参数和第二方法对应的输入参数一致且第一方法对应的输出参数和第二方法对应的输出参数一致，则建立第一方法和第二方法之间的一对一的方法对应关系。如果第一方法对应的输入参数和第二方法对应的输入参数之间满足包含关系或等价关系，以及，第一方法对应的输出参数和第二方法对应的输出参数一致，则建立第一方法和第二方法之间的一对一的方法对应关系。如果根据各第一方法对应的输出参数得到的目标输出参数与第二方法对应的输出参数一致，则建立第二方法与各第一方法之间的一对多的方法对应关系。如果根据各第二方法对应的输出参数得到的目标输出参数与第一方法对应的输出参数一致，则建立第一方法与各第二方法之间的一对多的方法对应关系。

[0102] 在本发明的实施例中，方法参数可以包括输入参数和输出参数。根据方法参数，可建立方法之间的一对一的方法对应关系或者一对多的方法对应关系(即多对一的方法对应关系)。具体的：

[0103] 如果第一方法对应的输入参数与第二方法对应的输入参数一致且第一方法的输

出参数与第二方法对应的输出参数一致，则可以说明第一方法和第二方法所实现的功能相似，基于此，可建立第一方法和第二方法之间的一对一的方法对应关系。上述可作如下理解：通常如果两个方法的输入参数一致，且输出参数一致，则可说明这两个方法所实现的功能相似。

[0104] 如果第一方法对应的输入参数与第二方法对应的输入参数之间满足包含关系或等价关系，以及，第一方法对应的输出参数和第二方法对应的输出参数一致，则可以说明第一方法和第二方法所实现的功能相似，基于此，可建立第一方法和第二方法之间的一对一的方法对应关系。上述可作如下理解：通常如果两个方法的输出参数一致，两个方法的输入参数之间满足包含关系或等价关系，则可说明这两个方法所实现的功能相似。两个方法的输入参数之间满足包含关系指的是两个方法的输入参数并不完全一致，但两个方法的输入参数之间具有包含关系。示例性的，如方法Q的输入参数为a、b和c，方法R的输入参数为a和b，则可认为方法Q和方法R的输入参数之间具有包含关系。两个方法的输入参数之间满足等价关系指的是两个方法的输入参数并不完全一致，但两个方法的输入参数之间具有等价关系。需要说明的是，等价关系的构建可根据实际情况进行设定，在此不作具体限定。示例性的，如方法Q的输入参数为a，方法R的输入参数为a1和a2，其中，a＝a1+a2，则可认为方法Q和方法R的输入参数之间具有等价关系。

[0105] 示例性的，如现存在数据库I和数据库J，数据库I中存在方法参数中输入参数为a、b和c，以及，输出参数为P的第一方法，数据库J中存在方法参数中输入参数为a和b，以及，输出参数为P的第二方法。由于第一方法对应的输入参数为a、b和c，输出参数为P，第二方法对应的输入参数为a和b，输出参数为P，第一方法的输出参数与第二方法的输出参数一致，且第一方法对应的输入参数与第二方法对应的输入参数之间具有包含关系，则可建立第一方法和第二方法之间的一对一的方法对应关系。

[0106] 又如现存在数据库I和数据库J，数据库I中存在方法参数中输入参数为0，和输出参数为P的第一方法，数据库J中存在方法参数中输入参数为o1和o2，以及，输出参数为P的第二方法，其中0＝o1+o2。由于第一方法对应的输入参数为0＝o1+o2，输出参数为P，第二方法对应的输入参数为o1和o2，输出参数为P，第一方法的输出参数与第二方法的输出参数一致，且第一方法对应的输入参数与第二方法对应的输入参数之间具有等价关系，则可建立第一方法和第二方法之间的一对一的方法对应关系。

[0107] 如果根据各第一方法对应的输出参数得到的目标输出参数与第二方法对应的输出参数一致，则可以说明第二方法和各第一方法进行结合后所实现的功能相似，基于此，可建立第二方法和各第一方法之间的一对多的方法对应关系。上述可作如下理解：虽然单个第一方法和单个第二方法所实现的功能并不相似，但多个第一方法进行结合后所实现的功能与单个第二方法所实现的功能相似，则可以建立第二方法和各第一方法之间的一对多的方法对应关系。

[0108] 示例性的，如现存在数据库I和数据库J，数据库I中存在输入参数为0和输出参数为P的第一方法，以及，数据库I中还存在输入参数为P和输出参数为T的第一方法，数据库J中存在输入参数0和输出参数为T的第二方法。可以理解到，数据库I中存在的输入参数为0和输出参数为P的第一方法，以及，数据库I中存在的输入参数为P和输出参数为T的第一方法，这两个第一方法结合后得到的目标输出参数为T，其与数据库J中存在的输出参数为T的

第二方法的输出参数一致，基于上述，可建立第二方法与这两个第一方法之间的方法对应关系。并且可以理解到，这两个第一方法之间的关联关系体现在一个第一方法的输出参数为另一个第一方法的输入参数。

[0109] 又如现存在数据库I和数据库J，数据库I中存在输入参数为0，以及，输出参数为p1和p2的第一方法，以及，数据库I中还存在输入参数为P和输出参数为T的第一方法，数据库J中存在输入参数0和输出参数为T的第二方法，其中，P＝p1+p2。可以理解到，数据库I中存在的输入参数为0，以及，输出参数为p1和p2的第一方法，同时，数据库I中存在的输入参数为P＝p1+p2和输出参数为T的第一方法，这两个第一方法结合后得到的目标输出参数为T，其与数据库J中存在的输出参数为P的第二方法的输出参数一致，基于上述，可建立第二方法与这两个第一方法之间的方法对应关系。并且可以理解到，这两个第一方法之间的关联关系体现在一个第一方法的输出参数与另一个第一方法的输入参数之间具有等价关系。

[0110] 再如现存在数据库I和数据库J，数据库I中存在输入参数为0和输出参数为P的第一方法，以及，数据库I中还存在输入参数为R和输出参数为T的第一方法，数据库J中存在输入参数0和输出参数为U的第二方法，其中，构建一个第一方法的输出参数P和另一个第一方法的输出参数T之间的关联关系后可以得到第二方法的输出参数U，即可理解为U＝f(P,T)。可以理解到，数据库I中存在的输入参数为0和输出参数为P的第一方法，以及，数据库I中存在的输入参数为R和输出参数为T的第一方法，U＝f(P,T)，这两个第一方法结合后得到的目标输出参数为U，其与数据库J中存在的输出参数为U的第二方法的输出参数一致，基于上述，可建立第二方法与这两个第一方法之间的方法对应关系。并且可以理解到，这两个第一方法之间的关联关系体现基于构建的两个第一方法的输出参数之间的关联关系可以得到第二方法的输出参数。

[0111] 可选的，在上述技术方案的基础上，根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法，具体可以包括：根据当前访问数据库标识对应的文档信息，获取第一类和第一方法。根据待访问数据库标识对应的文档信息，获取第二类和第二方法。

[0112] 在本发明的实施例中，通过如下方式确定第一类和第一方法，以及，第二类和第二方法，具体的：通常各数据库厂商在开发数据库时，会撰写相应的使用文档，使用文档中记录有该数据库所提供的类和方法等相关内容。基于上述，如果当前访问数据库的数据库访问接口为非JDBC API，则可根据当前访问数据库标识对应的文档信息，即可根据当前访问数据库对应的文档信息，获取第一类和第一方法。以及，根据待访问数据库标识对应的文档信息，即可根据待访问数据库对应的文档信息，获取第二类和第二方法。其中，文档信息可以理解为使用文档所提供的相关内容。

[0113] 此外，还可通过查看当前访问数据库对应的程序代码，获取第一类和第一方法。即可通过当前访问数据库提供的相关的程序代码，获取第一类和第一方法。同样的，可通过查看待访问数据库对应的程序代码，获取第二类和第二方法。即可通过待访问数据库提供的相关的程序代码，获取第二类和第二方法。

[0114] 可选的，在上述技术方案的基础上，如果与当前访问数据库标识对应的数据库访问接口为非JDBC API，则根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法，具体可以包括：如果与当前访问数据库标识对应

的数据库访问接口的接口名为非JDBC API接口名或与当前访问数据库标识对应的数据库访问接口的类名为非JDBC API类名，则根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法。

[0115] 在本发明的实施例中，可通过如下方式确定当前访问数据库标识对应的数据库访问接口是否为非JDBC API，具体的：由于JDBC API为Java应用程序使用数据库提供了统一的编程接口，它由一组Java接口和类组成，因此，可根据当前访问数据库对应的数据库访问接口的接口名是否为非JDBC API接口名或者当前访问数据库对应的数据库访问接口的类名是否为非JDBC API类名，确定当前访问数据库标识对应的数据库访问接口是否为非JDBC API。上述所述的非JDBC API接口名表示并不是JDBC API所设定的接口名。所述的非JDBC API类名表示并不是JDBC API所设定的类名。更为具体的：如果与当前访问数据库标识对应的数据库访问接口的接口名为非JDBC API接口名或与当前访问数据库标识对应的数据库访问接口的类名为非JDBC API类名，则可以说明当前访问数据库的数据访问接口为非JDBC API。

[0116] 如果确定出当前访问数据库的数据库访问接口为非JDBC API，则可以根据当年访问数据库访问标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法。根据当前访问数据库访问标识，获得第一类和第一方法，以及，根据待访问数据库标识，获得第二类和第二方法，具体可以包括：根据当前访问数据库标识对应的文档信息，获取第一类和第一方法。根据待访问数据库标识对应的文档信息，获取第二类和第二方法。

[0117] 可选的，在上述技术方案的基础上，将预设兼容文件和待访问数据库驱动作为兼容驱动，以替换当前访问数据库驱动之后，具体还可以包括：在兼容驱动下，调用与当前访问数据库标识对应的数据库访问接口，访问待访问数据库。

[0118] 在本发明的实施例中，在得到兼容驱动后，便可以通过当前访问数据标识对应的数据库访问接口，即可通过当前访问数据库的数据库访问接口，访问待访问数据库。在上述过程中无需修改程序中使用的JDBC驱动类的相关代码，便可以实现由访问当前访问数据库切换到访问待访问数据库。

[0119] 为了更好地理解本发明实施例所提供的技术方案，下面通过具体示例进行说明，具体的：当前待访问数据库为Oracle数据库，待访问数据库为MySQL数据库。Oracle数据库的数据库接口为非JDBC API。

[0120] 由于Oracle数据库的数据库接口为非JDBC API，因此，需要获取Oracle数据库的第一类和第一方法，以及，获取MySQL数据库的第二类和第二方法。根据类功能特征信息，建立第一类和第二类之间的类对应关系，以及，根据方法功能特征信息，建立第一方法和第二方法之间的方法对应关系。在预先构建的预设类中调用类对应关系，以及，预先构建的预设方法调用方法对应关系，得到预设兼容文件，并将预设兼容文件和MySQL数据库驱动，作为目标驱动，以替换Oracle数据库驱动，在目标驱动下，调用Oracle数据库的数据库访问接口，访问MySQL数据库。如图4所示，给出了本发明实施例提供的一种数据库驱动的处理方法的应用示意图。图4中，显示了当需要由Oracle数据库切换到MySQL数据库时，仅需要将MySQL数据库驱动和生成的预设兼容文件共同作为兼容驱动，以替换Oracle数据库驱动，而无需修改JDBC应用程序中引用非JDBC API的相关程序代码的。需要说明的是，Oracle数据

库驱动即为Oracle数据库JDBC驱动程序,MySQL数据库驱动即为MySQL数据库JDBC驱动程序。

[0121] 图5为本发明实施例提供的一种数据库驱动的处理装置的结构示意图,本实施例可适用于简化切换操作流程的情况,该装置可以采用软件和/或硬件的方式实现,该装置可以配置于设备中,例如典型的是服务器等。如图5所示,该装置具体包括:

[0122] 数据库访问请求获取模块210,用于获取数据库访问请求,数据库访问请求包括当前访问数据库标识和待访问数据库标识。

[0123] 类和方法获取模块220,用于如果与当前访问数据库标识对应的数据库访问接口为非Java数据库连接JDBC API,则根据当前访问数据库标识,获取第一类和第一方法,以及,根据待访问数据库标识,获取第二类和第二方法,第一类和第一方法分别为与当前访问数据库驱动对应的类和方法,第二类和所述第二方法分别为与待访问数据库驱动对应的类和方法。

[0124] 关系建立模块230,用于根据类功能特征信息,建立第一类和所述第二类之间的类对应关系,以及,根据方法功能特征信息,建立第一方法和第二方法之间的方法对应关系。

[0125] 兼容驱动模块240,用于分别在预先构建的预设类中调用所述类对应关系,以及,预设方法中调用方法对应关系,得到预设兼容文件,并将与兼容文件和待访问数据库驱动作为兼容驱动,以替换当前访问数据库驱动,预设类与第一类相对应,预设方法与第一方法相对应。

[0126] 本实施例的技术方案,通过获取数据库访问请求,数据库访问请求包括当前访问数据库标识和待访问数据库标识,如果与当前访问数据库标识对应的数据库访问接口为非JDBC API,则根据当前访问数据库标识,获取第一类和第一方法,以及,根据待访问数据库标识,获取第二类和第二方法,根据类功能特征类功能特征信息,建立第一类和第二类之间的类对应关系,以及,根据方法功能特征方法功能特征信息,建立第一方法和第二方法之间的方法对应关系,分别在预先构建的预设类中调用类对应关系,以及,预设方法中调用方法对应关系,得到预设兼容文件,并将预设兼容文件和待访问数据库驱动作为兼容驱动,以替换当前访问数据库驱动。上述通过构建的预设兼容文件建立起了当前访问数据库与待访问数据库之间的关联关系,基于预设兼容文件实现了无需修改JDBC应用程序的相关程序代码,即实现了JDBC应用程序中的程序代码的无缝迁移,便可由访问当前访问数据库切换到访问待访问数据库,简化了切换操作的流程,进而提高了切换操作的效率。

[0127] 可选的,在上述技术方案的基础上,关系建立模块230,具体可以包括:

[0128] 类对应关系第一建立子模块,用于如果第一类对应的类名与第二类对应的类名一致,则建立第一类和第二类之间的类对应关系。

[0129] 可选的,在上述技术方案的基础上,关系建立模块230,具体可以包括:

[0130] 类对应关系第二建立子模块,用于如果第一类对应的JDBC标准信息与第二类对应的JDBC标准信息一致,则建立第一类和第二类之间的类对应关系。

[0131] 可选的,在上述技术方案的基础上,关系建立模块230,具体可以包括:

[0132] 方法对应关系第一建立子模块,用于根据方法名,建立第一方法和第二方法之间的一对一的方法对应关系。

[0133] 方法对应关系第二建立子模块,用于根据方法参数,建立第一方法和第二方法之

间的一对一的方法对应关系或一对多的方法对应关系。

[0134] 可选的，在上述技术方案的基础上，方法对应关系第一建立子模块，具体可以包括：

[0135] 方法对应关系第一建立单元，用于如果第一方法对应的方法名和第二方法对应的方法名一致，则建立第一方法和第二方法之间的一对一的方法对应关系。

[0136] 可选的，在上述技术方案的基础上，方法对应关系第二建立子模块，具体可以包括：

[0137] 方法对应关系第二建立单元，用于如果第一方法对应的输入参数和第二方法对应的输入参数一致且第一方法对应的输出参数和第二方法对应的输出参数一致，则建立第一方法和第二方法之间的一对一的方法对应关系。

[0138] 方法对应关系第三建立单元，用于如果第一方法对应的输入参数和第二方法对应的输入参数之间满足包含关系或等价关系，以及，第一方法对应的输出参数和第二方法对应的输出参数一致，则建立第一方法和第二方法之间的一对一的方法对应关系。

[0139] 方法对应关系第四建立单元，用于如果根据各第一方法对应的输出参数得到的目标输出参数与第二方法对应的输出参数一致，则建立第二方法与各第一方法之间的一对多的方法对应关系。

[0140] 方法对应关系第五建立单元，用于如果根据各第二方法对应的输出参数得到的目标输出参数与第一方法对应的输出参数一致，则建立第一方法与各第二方法之间的一对多的方法对应关系。

[0141] 可选的，在上述技术方案的基础上，类和方法获取模块220，具体可以包括：

[0142] 类和方法第一获取子模块，用于根据当前访问数据库标识对应的文档信息，获取第一类和第一方法。

[0143] 类和方法第二获取子模块，用于根据待访问数据库标识对应的文档信息，获取第二类和第二方法。

[0144] 可选的，在上述技术方案的基础上，类和方法获取模块220，具体可以包括：

[0145] 类和方法第三获取子模块，用于如果与当前访问数据库标识对应的数据库访问接口的接口名为非Java数据库连接JDBC API接口名或与当前访问数据库标识对应的数据库访问接口的类名为非JDBC API类名，则根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法。

[0146] 可选的，在上述技术方案的基础上，该装置具体还可以包括：

[0147] 数据库访问模块，用于在兼容驱动下，调用与当前访问数据库标识对应的数据库访问接口，访问待访问数据库。

[0148] 本发明实施例所提供的数据库驱动的处理装置可执行本发明任意实施例所提供的数据库驱动的处理方法，具备执行方法相应的功能模块和有益效果。

[0149] 图6为本发明实施例提供的一种设备的结构示意图。图6显示的设备仅仅是一个示例，不应对本发明实施例的功能和使用范围带来任何限制。如图6所示，本发明实施例提供的设备，包括处理器31、存储器32、输入装置33和输出装置34；设备中处理器31的数量可以是一个或多个，图6中以一个处理器31为例；设备中的处理器31、存储器32、输入装置33和输出装置34可以通过总线或其他方式连接，图6中以通过总线连接为例。

[0150] 存储器32作为一种计算机可读存储介质，可用于存储软件程序、计算机可执行程序以及模块，如本发明实施例中的特征点采样方法对应的程序指令/模块(例如，使用数据库驱动的处理装置中的数据库访问请求获取模块210、类和方法获取模块220、关系建立模块230和兼容驱动模块240)。处理器31通过运行存储在存储器32中的软件程序、指令以及模块，从而执行各种功能应用以及数据处理，例如实现本发明实施例所提供的应用于设备的数据库驱动的处理方法。

[0151] 存储器32可主要包括存储程序区和存储数据区，其中，存储程序区可存储操作系统、至少一个功能所需的应用程序；存储数据区可存储根据设备的使用所创建的数据等。此外，存储器32可以包括高速随机存取存储器，还可以包括非易失性存储器，例如至少一个磁盘存储器件、闪存器件、或其他非易失性固态存储器件。在一些实例中，存储器32可进一步包括相对于处理器31远程设置的存储器，这些远程存储器可以通过网络连接至设备。上述网络的实例包括但不限于互联网、企业内部网、局域网、移动通信网及其组合。

[0152] 输入装置33可用于接收用户输入的数字或字符信息，以产生与设备的用户设置以及功能控制有关的键信号输入。输出装置34可包括显示屏等显示设备。

[0153] 当然，本领域技术人员可以理解，处理器还可以实现本发明任意实施例所提供应用于设备的数据库驱动的处理方法的技术方案。该设备的硬件结构以及功能可参见实施例的内容解释。

[0154] 本发明实施例还提供了一种计算机可读存储介质，其上存储有计算机程序，该程序被处理器执行时实现如本发明实施例所提供的一种数据库驱动的处理方法，该方法包括：

[0155] 获取数据库访问请求，数据库访问请求包括当前访问数据库标识和待访问数据库标识。

[0156] 如果与当前访问数据库标识对应的数据库访问接口为非Java数据库连接JDBC API，则根据当前访问数据库标识，获取第一类和第一方法，以及，根据待访问数据库标识，获取第二类和第二方法，第一类和第一方法分别为与当前访问数据库驱动对应的类和方法，第二类和第二方法分别为与待访问数据库驱动对应的类和方法。

[0157] 根据类功能特征信息，建立第一类和第二类之间的类对应关系，以及，根据方法功能特征信息，建立第一方法和第二方法之间的方法对应关系。

[0158] 分别在预先构建的预设类中调用类对应关系，以及，预设方法中调用方法对应关系，得到预设兼容文件，并将预设兼容文件和待访问数据库驱动作为兼容驱动，以替换当前访问数据库驱动，预设类与第一类相对应，预设方法与第一方法相对应。

[0159] 本发明实施例的计算机存储介质，可以采用一个或多个计算机可读的介质的任意组合。计算机可读介质可以是计算机可读信号介质或者计算机可读存储介质。在本文件中，计算机可读存储介质可以是任何包含或存储程序的有形介质，该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。

[0160] 计算机可读的信号介质可以包括在基带中或者作为载波一部分传播的数据信号，其中承载了计算机可读的程序代码。这种传播的数据信号可以采用多种形式，包括但不限于电磁信号、光信号或上述的任意合适的组合。计算机可读的信号介质还可以是计算机可读存储介质以外的任何计算机可读介质，该计算机可读介质可以发送、传播或者传输用于

由指令执行系统、装置或者器件使用或者与其结合使用的程序。

[0161] 计算机可读介质上包含的程序代码可以用任何适当的介质传输，包括——但不限于无线、电线、光缆、射频等等，或者上述的任意合适的组合。

[0162] 可以以一种或多种程序设计语言或其组合来编写用于执行本发明操作的计算机程序代码，例如，C++、C语言和JAVA等。程序代码可以完全地在用户计算机上执行、部分地在用户计算机上执行、作为一个独立的软件包执行、部分在用户计算机上部分在远程计算机上执行、或者完全在远程计算机或服务器上执行。在涉及远程计算机的情形中，远程计算机可以通过任意种类的网络——局域网(Local Area Network，LAN)或广域网(Wide Area Network，WAN)—连接到用户计算机，或者，可以连接到外部计算机(例如利用因特网服务提供商来通过因特网连接)。

[0163] 当然，本发明实施例所提供的一种计算机可读存储介质，其计算机可执行指令不限于如上所述的方法操作，还可以执行本发明任意实施例所提供的设备的数据库驱动的处理方法的相关操作。对存储介质的介绍可参见实施例中的内容解释。

[0164] 注意，上述仅为本发明的较佳实施例及所运用技术原理。本领域技术人员会理解，本发明不限于这里所述的特定实施例，对本领域技术人员来说能够进行各种明显的变化、重新调整和替代而不会脱离本发明的保护范围。因此，虽然通过以上实施例对本发明进行了较为详细的说明，但是本发明不仅仅限于以上实施例，在不脱离本发明构思的情况下，还可以包括更多其他等效实施例，而本发明的范围由所附的权利要求范围决定。

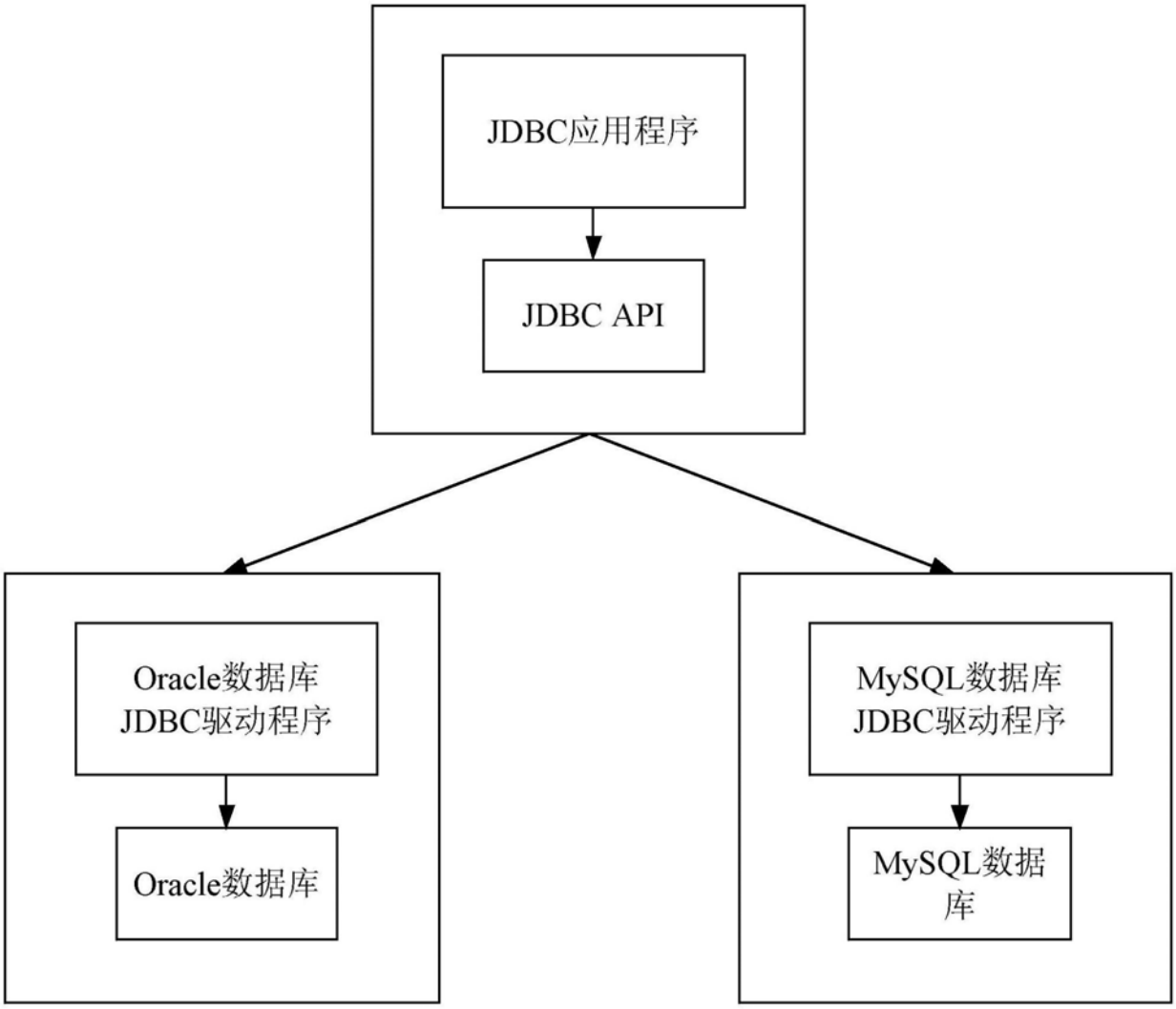

JDBC应用程序
JDBC API
Oracle数据库
JDBC驱动程序
Oracle数据库
MySQL数据库
JDBC驱动程序
MySQL数据
库

图1

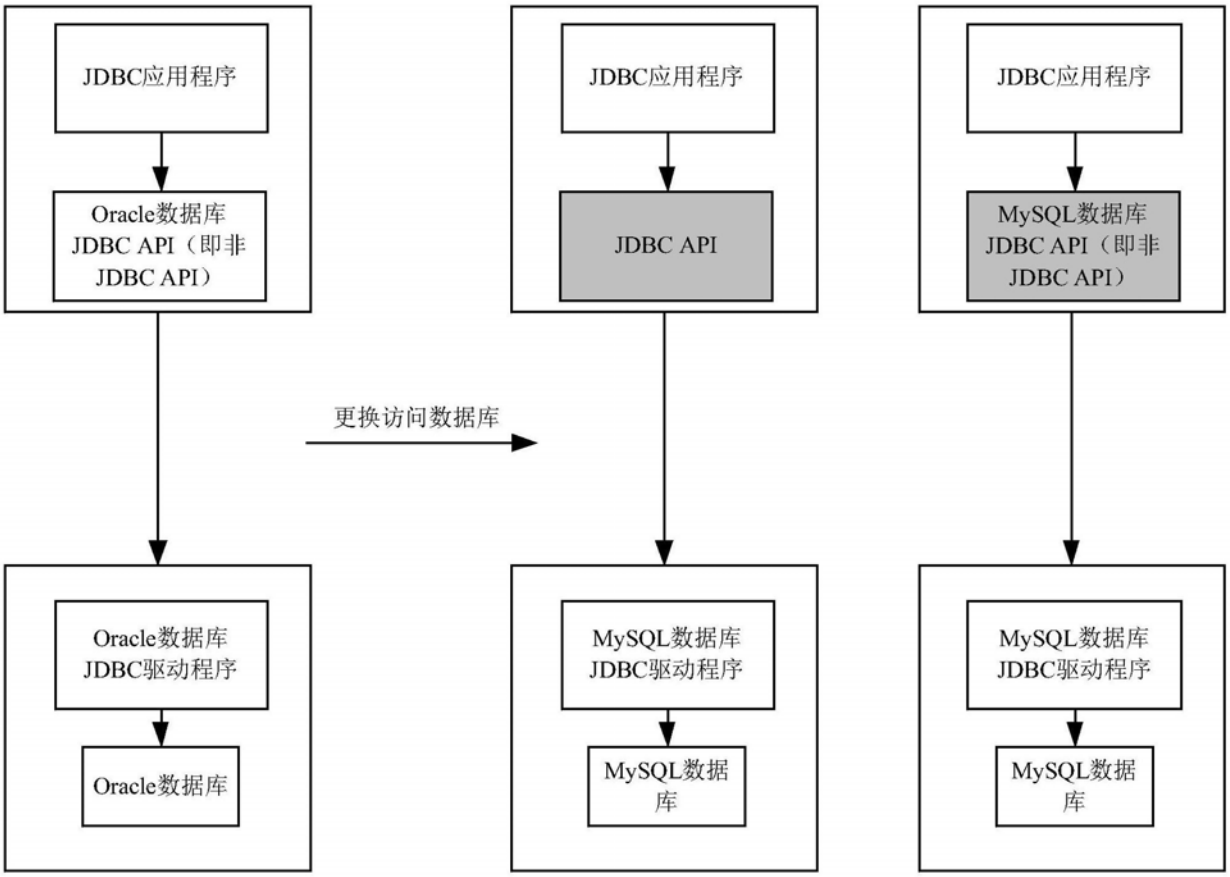

JDBC应用程序
Oracle数据库
JDBC API（即非
JDBC API）
Oracle数据库
JDBC驱动程序
Oracle数据库
更换访问数据库
JDBC应用程序
JDBC API
MySQL数据库
JDBC驱动程序
MySQL数据
库
JDBC应用程序
MySQL数据库
JDBC API（即非
JDBC API）
MySQL数据库
JDBC驱动程序
MySQL数据
库

图2

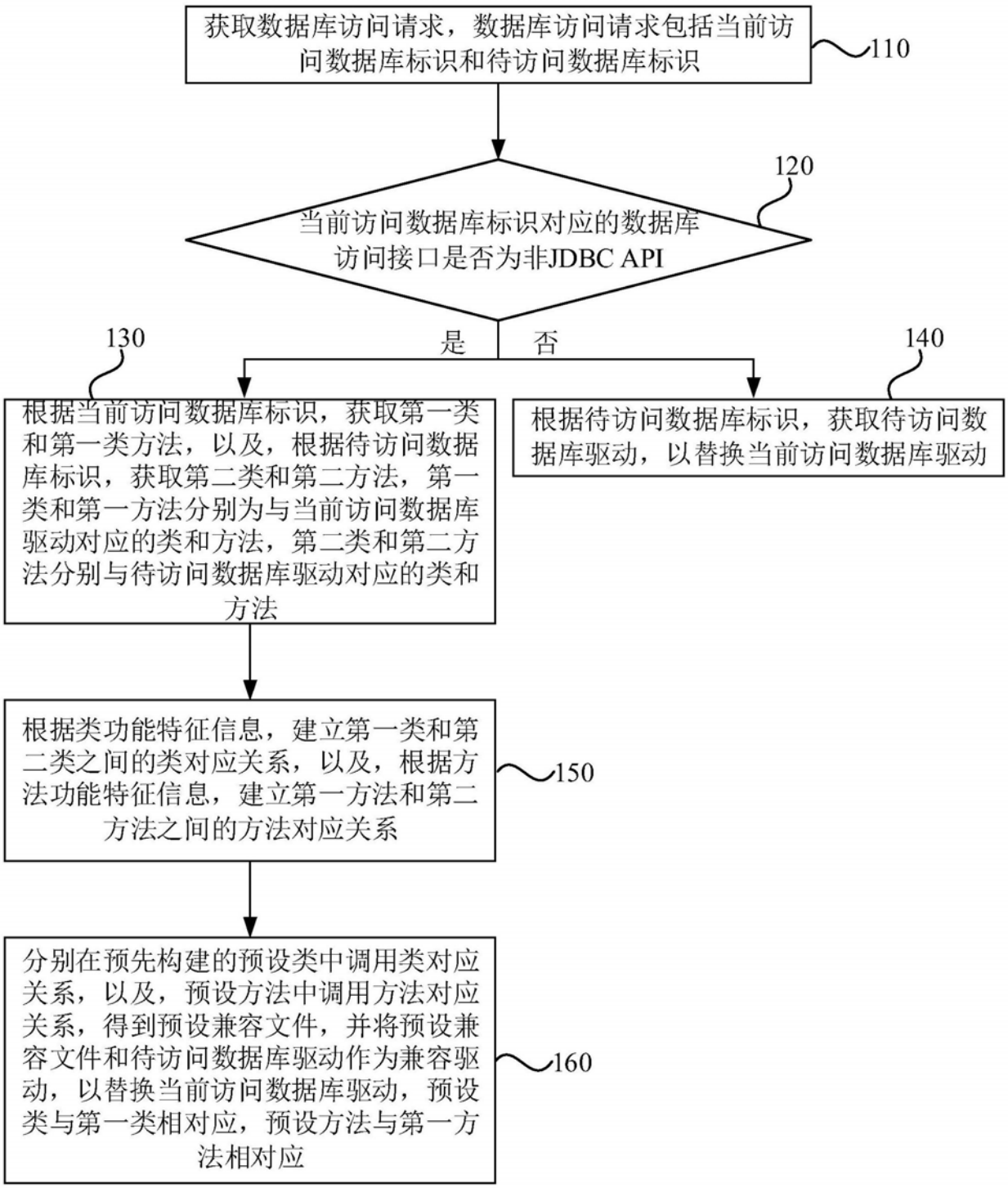

获取数据库访问请求，数据库访问请求包括当前访问数据库标识和待访问数据库标识
110
120
当前访问数据库标识对应的数据库访问接口是否为非JDBC API
是
否
130
140
根据当前访问数据库标识，获取第一类和第一类方法，以及，根据待访问数据库标识，获取第二类和第二方法，第一类和第一方法分别为与当前访问数据库驱动对应的类和方法，第二类和第二方法分别与待访问数据库驱动对应的类和方法
根据待访问数据库标识，获取待访问数据库驱动，以替换当前访问数据库驱动
根据类功能特征信息，建立第一类和第二类之间的类对应关系，以及，根据方法功能特征信息，建立第一方法和第二方法之间的方法对应关系
150
分别在预先构建的预设类中调用类对应关系，以及，预设方法中调用方法对应关系，得到预设兼容文件，并将预设兼容文件和待访问数据库驱动作为兼容驱动，以替换当前访问数据库驱动，预设类与第一类相对应，预设方法与第一方法相对应
160

图3

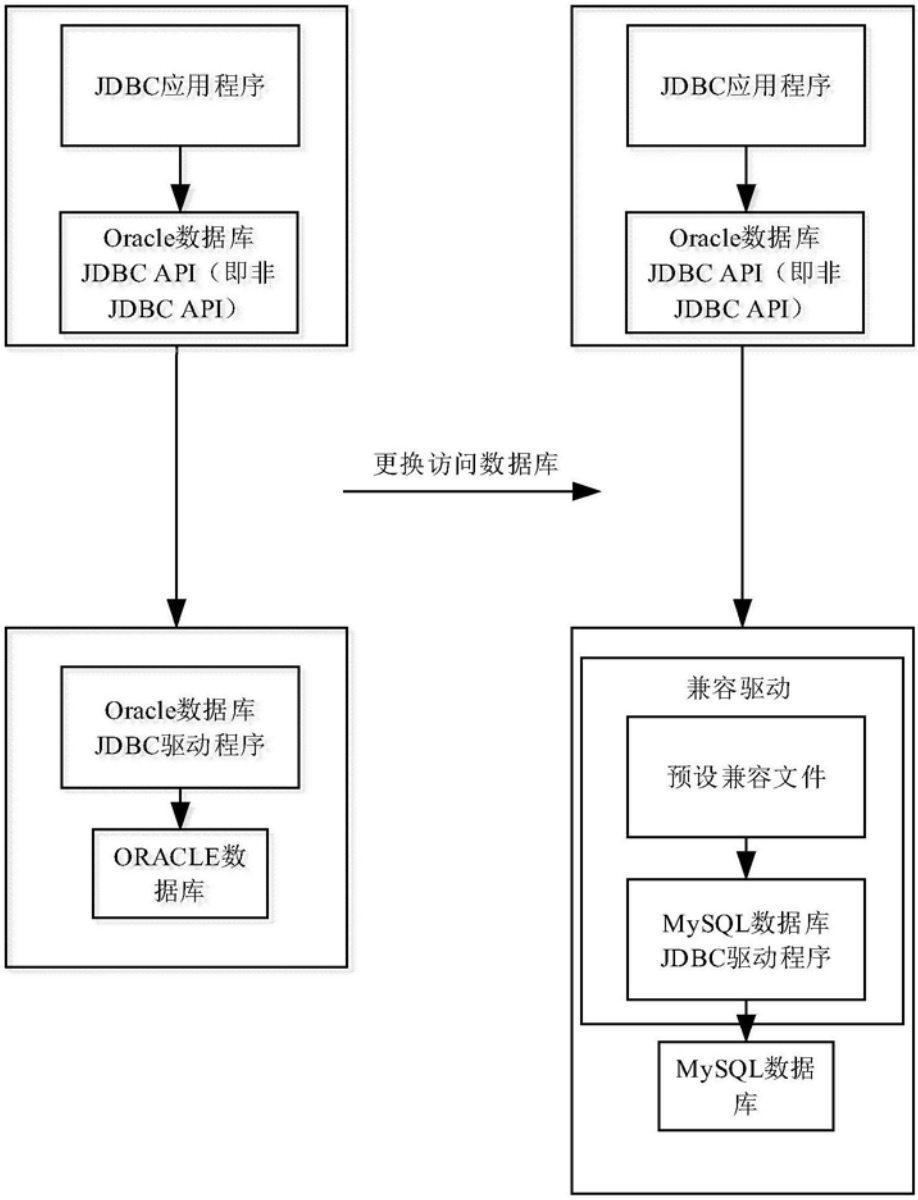

JDBC应用程序
Oracle数据库
JDBC API（即非
JDBC API）
JDBC应用程序
Oracle数据库
JDBC API（即非
JDBC API）
更换访问数据库
Oracle数据库
JDBC驱动程序
ORACLE数
据库
兼容驱动
预设兼容文件
MySQL数据库
JDBC驱动程序
MySQL数据
库

图4

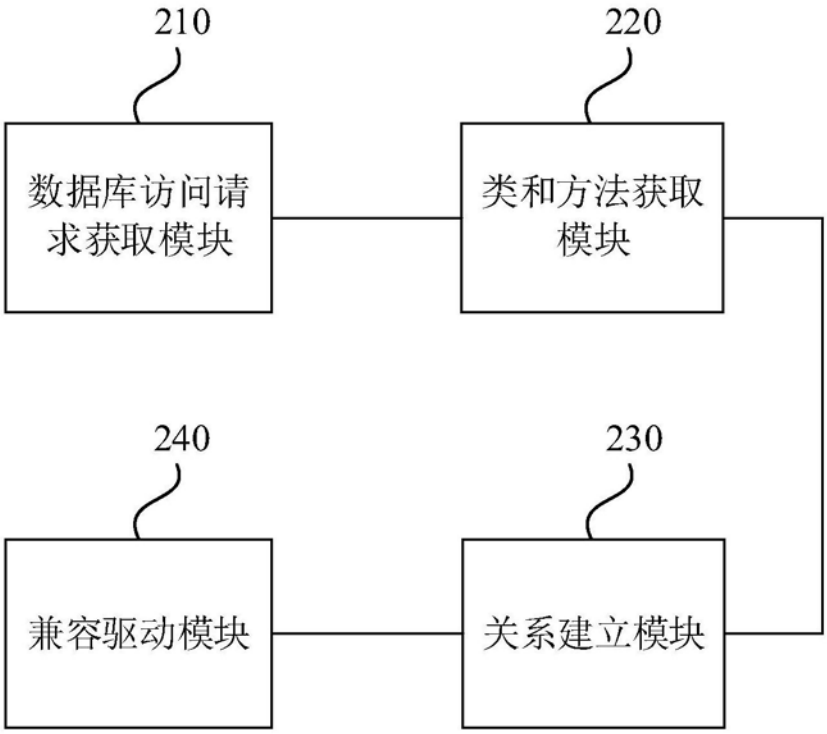


图5

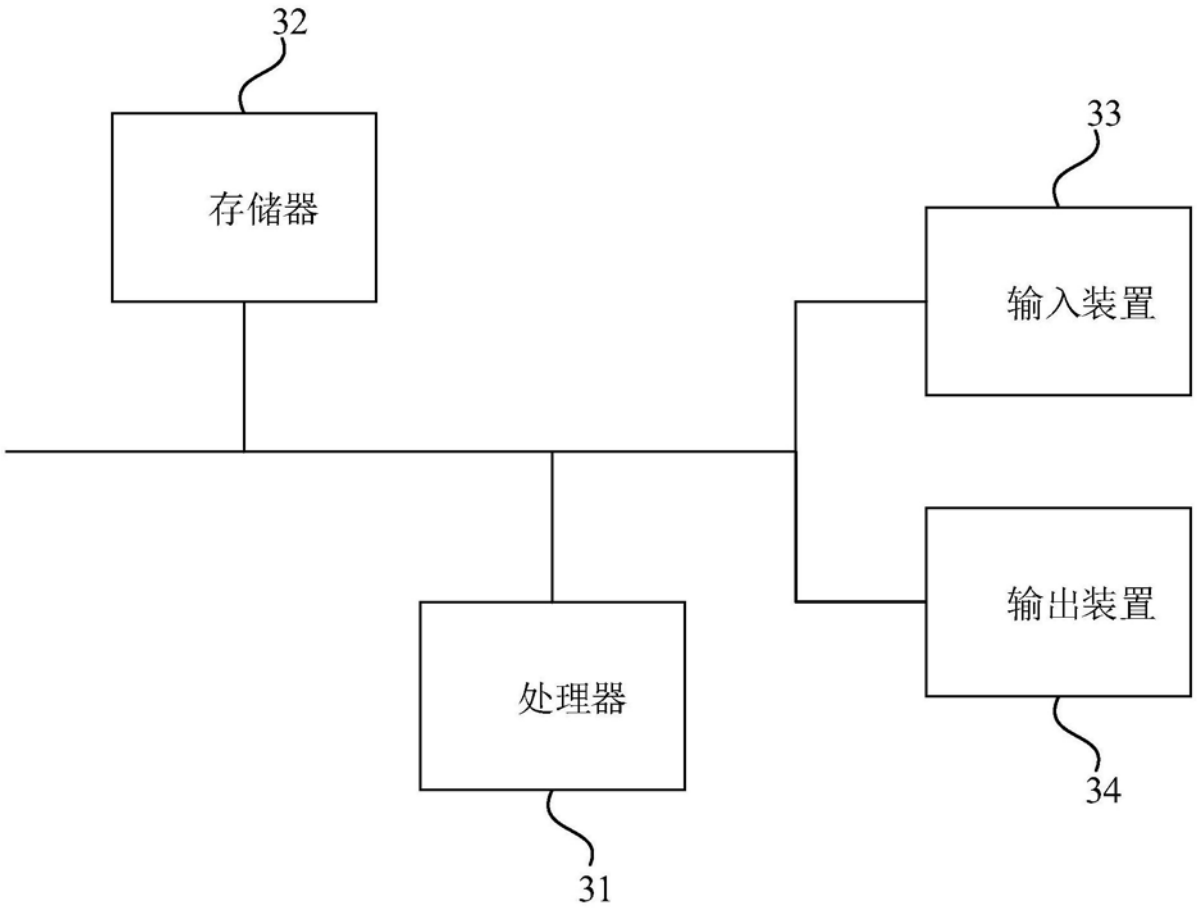


图6