

一种数据同步的方法、装置及设备

技术领域

[01] 本说明书涉及计算机技术领域，尤其涉及一种数据同步的方法、装置及设备。

背景技术

5 [02] 为了满足用户日益增长的数据查询需求，当前，可以将一些常用的用户数据保存在缓存中，这样不仅能够提高数据的查询效率，同时还能有效的缓解数据库的查询压力，如图 1 所示。

[03] 图 1 为本说明书提供的现有技术中用户通过服务端进行数据查询的示意图。

[04] 对于存储在缓存中的数据 value1 来说，服务端可以在接收到携带有 key1 的数据查
10 询请求后，直接从缓存中查询到 key1 对应的数据 value1。而对于未存储在缓存中的数
据 value4 来说，服务端可以通过接收到数据查询请求中携带的 key4，从缓存中查询是
否存在 key4 对应的数据。若确定缓存中未存储有 key4 对应的数据，则可以向数据库查
询 key4 对应的数据，进而从数据库中获取到 key4 对应的数据 value4。同时，服务端可
15 以将 key4 和 value4 按照对应关系存储在缓存中，以便后续用户能够从缓存中快速的查
询 key4 对应的数据。

[05] 在实际应用中，可能会出现数据库存储的业务数据发生了变化，但缓存中存储的
业务数据却并未发生变化的情况出现，如图 2 所示。

[06] 图 2 为本说明书提供的数据库和缓存中存储的数据不一致的示意图。

[07] 例如，假设用户 1 起初并未设置用于进行身份验证的验证信息，则数据库中存储
20 的用户 1 的用户数据 value1 中将不包含有用户的验证信息，同时，缓存中存储的用户数
据 value1 中也将不包含有用户的验证信息。而随着用户 1 对身份验证需求的不断增加，
用户 1 可以在服务端设置用于进行身份验证的验证信息，这样，用户 1 的验证信息将进
一步作为用户 1 的用户数据进行保存，用户数据 value1 也将随着变更为 value1'。然而，
缓存中存储的依然是 value1，这样一来，服务端将无法通过从缓存中获取到的用户数据
25 value1，对用户 1 实施身份验证。

[08] 基于现有技术，需要一种有效的数据同步方式。

发明内容

[09] 本说明书提供一种数据同步的方法，用以解决现有技术中无法对数据库和缓存中的数据进行同步的问题。

[10] 本说明书提供了一种数据同步的方法，包括：

5 [11] 应用确定待查询数据；

[12] 从缓存中获取所述待查询数据对应的版本标识；

[13] 当确定所述待查询数据对应的版本标识与设定版本标识不一致时，则从数据库中查询所述待查询数据，并将从所述数据库中获取到的待查询数据替换存储在所述缓存中的所述待查询数据。

10 [14] 本说明书提供一种数据同步的装置，用以解决现有技术中无法对数据库和缓存中的数据进行同步的问题。

[15] 本说明书提供了一种数据同步的装置，包括：

[16] 确定模块，确定待查询数据；

[17] 获取模块，从缓存中获取所述待查询数据对应的版本标识；

15 [18] 处理模块，当确定所述待查询数据对应的版本标识与设定版本标识不一致时，则从数据库中查询所述待查询数据，并将从所述数据库中获取到的待查询数据替换存储在所述缓存中的所述待查询数据。

[19] 本说明书提供一种数据同步的设备，用以解决现有技术中无法对数据库和缓存中的数据进行同步的问题。

20 [20] 本说明书提供了一种数据同步的设备，包括一个或多个存储器以及处理器，所述存储器存储程序，并且被配置成由所述一个或多个处理器执行以下步骤：

[21] 确定待查询数据；

[22] 从缓存中获取所述待查询数据对应的版本标识；

25 [23] 当确定所述待查询数据对应的版本标识与设定版本标识不一致时，则从数据库中查询所述待查询数据，并将从所述数据库中获取到的待查询数据替换存储在所述缓存中的所述待查询数据。

[24] 本说明书采用的上述至少一个技术方案能够达到以下有益效果:

[25] 在本说明书一个或多个实施例中, 应用可以确定待查询数据, 并从缓存中获取该待查询数据对应的版本标识, 当确定该待查询数据对应的版本标识与设定版本标识不一致时, 则从数据库中查询该待查询数据, 并将从该数据库中获取到的待查询数据替换存储在该缓存中的该待查询数据。

[26] 从上述方法中可以看出, 由于可以通过比对缓存中存储的待查询数据对应的版本标识与设定版本标识是否一致, 来确定缓存和数据库中分别存储的该待查询数据是否一致, 并在确定出两者不一致时, 进行数据同步, 从而有效的保证了数据库和缓存中的数据一致性。

10 附图说明

[27] 此处所说明的附图用来提供对本说明书的进一步理解, 构成本说明书的一部分, 本说明书的示意性实施例及其说明用于解释本说明书, 并不构成对本说明书的不当限定。在附图中:

[28] 图 1 为本说明书提供的现有技术中用户通过服务端进行数据查询的示意图;

15 [29] 图 2 为本说明书提供的数据库和缓存中存储的数据不一致的示意图;

[30] 图 3 为本说明书提供的数据同步的过程示意图;

[31] 图 4 为本说明书提供的应用从缓存中获取待查询数据对应的版本标识的示意图;

[32] 图 5 为本说明书提供的数据同步的装置示意图;

[33] 图 6 为本说明书提供的数据同步的设备示意图。

20 具体实施方式

[34] 为了使本技术领域的人员更好地理解本说明书一个或多个实施例中的技术方案, 下面将结合本说明书一个或多个实施例中的附图, 对本说明书一个或多个实施例中的技术方案进行清楚、完整地描述, 显然, 所描述的实施例仅仅是本说明书一部分实施例, 而不是全部的实施例。基于本说明书中的实施例, 本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例, 都应当属于本说明书保护的范围。

[35] 图 3 为本说明书提供的数据同步的过程示意图, 具体包括以下步骤:

[36] S300: 应用确定待查询数据。

[37] 在本说明书中, 应用可以在业务执行过程中确定待查询数据。其中, 该待查询数据可以是应用根据用户发送的数据查询请求确定出的, 即, 用户需要通过该应用执行数据查询业务。

5 [38] 当然, 该待查询数据也可以是应用对用户发送的业务请求进行处理过程中, 确定出的执行该业务时所查询的数据。例如, 应用可以根据用户发送的支付请求(即上述提到的业务请求), 确定在对该支付请求对应的支付业务进行处理的过程中, 需要查询该用户的账户数据, 这里提到的账户数据即为上述的待查询数据。

10 [39] 这里提到的应用可以指能够对用户执行的业务进行处理的服务端, 当然也可以是指安装在诸如手机、电脑等终端设备上的客户端。随着业务的不断更新, 该应用在执行业务过程的业务逻辑、数据格式等也将会发生相应的变化。相应的, 该应用每更新一次, 其对应的版本信息也将随之发生变化。在本说明书中, 当监测到该应用的版本信息发生变更时, 可以根据变更后的版本信息确定出设定版本标识, 以通过该设定版本标识, 对后续从缓存中获取到的待查询数据对应的版本标识进行比对。换句话说, 该设定版本标识将随着该应用的更新而发生改变, 变更为与该应用当前的版本信息相对应的版本标识。

[40] S302: 从缓存中获取所述待查询数据对应的版本标识。

[41] 应用在确定该待查询数据后, 可以根据确定出的该待查询数据对应的数据标识, 从缓存中查询该数据标识对应的数据。这里提到的数据标识用于标识需要查询的数据。例如, 缓存中以 key-value 形式存储数据时, key 即为数据标识, value 即为数据标识 key
20 对应的数据。这里提到的缓存可以是设置于业务系统中的分布式缓存。

[42] 应用可以通过确定出的该数据标识, 从缓存中查询与该数据标识对应的数据。若确定该分布式缓存中存储有与数据标识对应的数据(即待查询数据), 则可以获取该待查询数据对应的版本标识, 其中, 该版本标识可以版本号、版本变更的时间信息等形式出现。

25 [43] 在本说明书中, 该版本标识可以携带在缓存中存储的该待查询数据中。相应的, 应用可以根据确定出的数据标识, 从该缓存中获取与该数据标识对应的待查询数据, 进而从该待查询数据中获取该待查询数据对应的版本标识, 如图 4 所示。

[44] 图 4 为本说明书提供的应用从缓存中获取待查询数据对应的版本标识的示意图。

[45] 当缓存中以 key-value 形式存储数据时, 应用可以根据待查询数据对应的查询主键

key1, 从缓存中查询出与该 key1 对应的值: value1, 其中, 该 value1 中携带有版本标识 V1, 因此, 应用可以从该 value1 中获取出该 value1 (即待查询数据) 对应的版本标识。

[46] 当然, 在本说明书中, 该版本标识也可以携带在缓存中存储的数据所对应的标识信息, 这里可以将该标识信息称之为存储在缓存中的数据对应的实际标识。相应的, 应用在确定出待查询数据对应的数据标识后, 可以从该缓存中确定与该数据标识相匹配的实际标识, 进而从该实际标识中获取该待查询数据对应的版本标识。

[47] 例如, 假设待查询数据 value1 在缓存中以 key1_V1-value1 的形式存储, 该待查询数据 value1 对应的实际标识为 key1_V1, V1 即为携带在该实际标识中的版本标识。因此, 应用可以根据待查询数据对应的数据标识 key1, 从缓存中确定与该数据标识 key1 相匹配的实际标识 key1_V1, 继而可以从该实际标识 key1_V1 中确定出该待查询数据 value1 对应的版本标识 V1。

[48] S304: 当确定所述待查询数据对应的版本标识与设定版本标识不一致时, 则从数据库中查询所述待查询数据, 并将从所述数据库中获取到的待查询数据替换存储在所述缓存中的待查询数据。

[49] 应用从缓存中获取到待查询数据对应的版本标识后, 可以将该版本标识与设定版本标识进行比对, 当确定两者不一致时, 则可以确定数据库存储的该待查询数据和缓存中存储的该待查询数据不同步, 进而将数据库中存储的该待查询数据同步至缓存中。

[50] 具体的, 当确定从缓存中获取到的待查询数据对应的版本标识与设定版本标识不一致时, 则可以失效该缓存中存储的待查询数据, 并将从数据库中获取到的待查询数据存储在缓存中。

[51] 例如, 在图 4 中, 应用当前的版本标识为 V2, 可以将该版本标识 V2 作为设定版本标识。并且, 数据库中存储的各数据对应的版本标识均为 V2。应用从缓存中获取到待查询数据 value1 对应的版本标识 V1 后, 确定出版本标识 V1 与该设定版本标识 V2 不符, 则可以通过预设的指令, 强制失效缓存中存储的待查询数据 V1。同时, 该应用可以通过该待查询数据对应的数据标识 key1, 从数据库中查询出该数据标识 key1 对应的数据 value1', 以获取数据 value1', 并将 value1' 存储在缓存中。

[52] 即, 相当于在确定缓存中存储的待查询数据对应的版本标识与设定版本标识不一致时, 通过数据库中存储的版本标识与设定版本标识相一致的数据替换存储在缓存中的待查询数据, 从而完成数据库和缓存中的数据同步。

[53] 当然，若是确定待查询数据对应的版本标识与该设定版本标识相一致，则可以确定出数据库中存储的该待查询数据与缓存中存储的待查询数据相同步，进而可直接获取缓存中存储的待查询数据。

5 [54] 从上述可以看出，在本说明书提供的数据库同步方法中，相当于以数据库中存储的数据的版本标识为基准，来判断缓存中存储的数据的版本标识与数据库中存储的数据的版本标识是否一致，以此来实现数据库和缓存中的数据同步，从而有效的保证了数据库和缓存中的数据一致性。

10 [55] 需要说明的是，上述提到的设定版本标识可以是根据应用当前的版本信息确定出的版本标识，也可以是根据当前数据库中存储的数据对应的版本标识。应用当前的版本信息所对应的版本标识可以与数据库中存储的数据对应的版本标识相一致，当然也可不一致。但无论是以应用当前的版本信息确定出的版本标识作为设定版本标识，还是以数据库中存储的数据对应的版本标识为设定版本标识，若是从缓存中获取到的待查询数据的版本标识与该设定版本标识不一致，即可将数据库中存储的待查询数据替换缓存中存储的待查询数据，以实现该待查询数据在数据库和缓存中的数据同步。

15 [56] 以上为本说明书的一个或多个实施例提供的数据库同步的方法，基于同样的思路，本说明书还提供了相应的数据库同步的装置，如图 5 所示。

[57] 图 5 为本说明书提供的数据库同步的装置示意图，具体包括：

[58] 确定模块 501，确定待查询数据；

[59] 获取模块 502，从缓存中获取所述待查询数据对应的版本标识；

20 [60] 处理模块 503，当确定所述待查询数据对应的版本标识与设定版本标识不一致时，则从数据库中查询所述待查询数据，并将从所述数据库中获取到的待查询数据替换存储在所述缓存中的所述待查询数据。

[61] 所述处理模块 503，当确定所述待查询数据对应的版本标识与所述设定版本标识相一致时，则从所述缓存中获取所述待查询数据。

25 [62] 所述确定模块 501，确定所述待查询数据对应的数据标识；

[63] 所述获取模块 502，根据确定出的所述数据标识，从所述缓存中获取与所述数据标识对应的待查询数据；从所述待查询数据中获取所述待查询数据对应的版本标识。

[64] 所述确定模块 501，确定所述待查询数据对应的数据标识；

[65] 所述获取模块 502, 从所述缓存中确定与所述数据标识相匹配的实际标识; 从所述实际标识中获取所述待查询数据对应的版本标识。

[66] 所述处理模块 503, 失效所述缓存中存储的所述待查询数据; 将从所述数据库中获取到的待查询数据存储到所述缓存中。

5 [67] 所述装置还包括:

[68] 监测模块 504, 当监测到所述装置的版本信息发生变更时, 根据变更后的版本信息确定所述设定版本标识。

[69] 基于上述说明的数据同步的方法, 本说明书还对应提供了一种用于数据同步的设备, 如图 6 所示。该设备包括一个或多个存储器以及处理器, 所述存储器存储程序, 并且被配置成由所述一个或多个处理器执行以下步骤:

[70] 确定待查询数据;

[71] 从缓存中获取所述待查询数据对应的版本标识;

[72] 当确定所述待查询数据对应的版本标识与设定版本标识不一致时, 则从数据库中查询所述待查询数据, 并将从所述数据库中获取到的待查询数据替换存储在所述缓存中的所述待查询数据。

[73] 在本说明书的一个或多个实施例中, 应用可以确定待查询数据, 并从缓存中获取该待查询数据对应的版本标识, 当确定该待查询数据对应的版本标识与设定版本标识不一致时, 则从数据库中查询该待查询数据, 并将从该数据库中获取到的待查询数据替换存储在该缓存中的该待查询数据。

20 [74] 从上述方法中可以看出, 由于可以通过比对缓存中存储的待查询数据对应的版本标识与设定版本标识是否一致, 来确定缓存和数据库中分别存储的该待查询数据是否一致, 并在确定出两者不一致时, 进行数据同步, 从而有效的保证了数据库和缓存中的数据一致性。

25 [75] 在 20 世纪 90 年代, 对于一个技术的改进可以很明显地区分是硬件上的改进 (例如, 对二极管、晶体管、开关等电路结构的改进) 还是软件上的改进 (对于方法流程的改进)。然而, 随着技术的发展, 当今的很多方法流程的改进已经可以视为硬件电路结构的直接改进。设计人员几乎都通过将改进的方法流程编程到硬件电路中来得到相应的硬件电路结构。因此, 不能说一个方法流程的改进就不能用硬件实体模块来实现。例如,

- 可编程逻辑器件 (Programmable Logic Device, PLD) (例如现场可编程门阵列 (Field Programmable Gate Array, FPGA)) 就是这样一种集成电路, 其逻辑功能由用户对器件编程来确定。由设计人员自行编程来把一个数字系统“集成”在一片 PLD 上, 而不需要请芯片制造厂商来设计和制作专用的集成电路芯片。而且, 如今, 取代手工地制作集成电路芯片, 这种编程也多半改用“逻辑编译器 (logic compiler)”软件来实现, 它与程序开发撰写时所用的软件编译器相类似, 而要编译之前的原始代码也得用特定的编程语言来撰写, 此称之为硬件描述语言 (Hardware Description Language, HDL), 而 HDL 也并非仅有一种, 而是有许多种, 如 ABEL (Advanced Boolean Expression Language)、AHDL (Altera Hardware Description Language)、Confluence、CUPL (Cornell University Programming Language)、HDCal、JHDL (Java Hardware Description Language)、Lava、Lola、MyHDL、PALASM、RHDH (Ruby Hardware Description Language) 等, 目前最普遍使用的是 VHDL (Very-High-Speed Integrated Circuit Hardware Description Language) 与 Verilog。本领域技术人员也应该清楚, 只需要将方法流程用上述几种硬件描述语言稍作逻辑编程并编程到集成电路中, 就可以很容易得到实现该逻辑方法流程的硬件电路。
- 15 [76] 控制器可以按任何适当的方式实现, 例如, 控制器可以采取例如微处理器或处理器以及存储可由该 (微) 处理器执行的计算机可读程序代码 (例如软件或固件) 的计算机可读介质、逻辑门、开关、专用集成电路 (Application Specific Integrated Circuit, ASIC)、可编程逻辑控制器和嵌入微控制器的形式, 控制器的例子包括但不限于以下微控制器: ARC 625D、Atmel AT91SAM、Microchip PIC18F26K20 以及 Silicone Labs C8051F320,
- 20 存储器控制器还可以被实现为存储器的控制逻辑的一部分。本领域技术人员也知道, 除了以纯计算机可读程序代码方式实现控制器以外, 完全可以通过将方法步骤进行逻辑编程来使得控制器以逻辑门、开关、专用集成电路、可编程逻辑控制器和嵌入微控制器等的形式来实现相同功能。因此这种控制器可以被认为是一种硬件部件, 而对其内包括的用于实现各种功能的装置也可以视为硬件部件内的结构。或者甚至, 可以将用于实现各种功能的装置视为既可以是实现方法的软件模块又可以是硬件部件内的结构。
- 25 [77] 上述实施例阐明的系统、装置、模块或单元, 具体可以由计算机芯片或实体实现, 或者由具有某种功能的产品来实现。一种典型的实现设备为计算机。具体的, 计算机例如可以为个人计算机、膝上型计算机、蜂窝电话、相机电话、智能电话、个人数字助理、媒体播放器、导航设备、电子邮件设备、游戏控制台、平板计算机、可穿戴设备或者这些设备中的任何设备的组合。
- 30

[78] 为了描述的方便,描述以上装置时以功能分为各种单元分别描述。当然,在实施本说明书时可以把各单元的功能在同一个或多个软件和/或硬件中实现。

- 5 [79] 本领域内的技术人员应明白,本说明书的实施例可提供为方法、系统、或计算机程序产品。因此,本说明书可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且,本说明书可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

- 10 [80] 本说明书是参照根据本说明书一个或多个实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器,使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

- 15 [81] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中,使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品,该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

- 20 [82] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上,使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理,从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[83] 在一个典型的配置中,计算设备包括一个或多个处理器(CPU)、输入/输出接口、网络接口和内存。

- 25 [84] 内存可能包括计算机可读介质中的非永久性存储器,随机存取存储器(RAM)和/或非易失性内存等形式,如只读存储器(ROM)或闪存(flash RAM)。内存是计算机可读介质的示例。

[85] 计算机可读介质包括永久性和非永久性、可移动和非可移动媒体可以由任何方法或技术来实现信息存储。信息可以是计算机可读指令、数据结构、程序的模块或其他数

据。计算机的存储介质的例子包括,但不限于相变内存(PRAM)、静态随机存取存储器(SRAM)、动态随机存取存储器(DRAM)、其他类型的随机存取存储器(RAM)、只读存储器(ROM)、电可擦除可编程只读存储器(EEPROM)、快闪记忆体或其他内存技术、只读光盘只读存储器(CD-ROM)、数字多功能光盘(DVD)或其他光学存储、磁盒式磁带,磁
5 带磁磁盘存储或其他磁性存储设备或任何其他非传输介质,可用于存储可以被计算设备访问的信息。按照本文中的界定,计算机可读介质不包括暂存电脑可读媒体(transitory media),如调制的数据信号和载波。

[86] 还需要说明的是,术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含,从而使得包括一系列要素的过程、方法、商品或者设备不仅包括那些要素,而且还
10 包括没有明确列出的其他要素,或者是还包括为这种过程、方法、商品或者设备所固有的要素。在没有更多限制的情况下,由语句“包括一个……”限定的要素,并不排除在包括所述要素的过程、方法、商品或者设备中还存在另外的相同要素。

[87] 本说明书可以在由计算机执行的计算机可执行指令的一般上下文中描述,例如程序模块。一般地,程序模块包括执行特定任务或实现特定抽象数据类型的例程、程序、
15 对象、组件、数据结构等等。也可以在分布式计算环境中实践本说明书的一个或多个实施例,在这些分布式计算环境中,由通过通信网络而被连接的远程处理设备来执行任务。在分布式计算环境中,程序模块可以位于包括存储设备在内的本地和远程计算机存储介质中。

[88] 本说明书中的各个实施例均采用递进的方式描述,各个实施例之间相同相似的部分互相参见即可,每个实施例重点说明的都是与其他实施例的不同之处。尤其,对于系
20 统实施例而言,由于其基本相似于方法实施例,所以描述的比较简单,相关之处参见方法实施例的部分说明即可。

[89] 上述对本说明书特定实施例进行了描述。其它实施例在所附权利要求书的范围内。在一些情况下,在权利要求书中记载的动作或步骤可以按照不同于实施例中的顺序来执
25 行并且仍然可以实现期望的结果。另外,在附图中描绘的过程不一定要求示出的特定顺序或者连续顺序才能实现期望的结果。在某些实施方式中,多任务处理和并行处理也是可以的或者可能是有利的。

[90] 以上所述仅为本说明书的一个或多个实施例而已,并不用于限制本说明书。对于本领域技术人员来说,本说明书的一个或多个实施例可以有各种更改和变化。凡在本说
30 明书的一个或多个实施例的精神和原理之内所作的任何修改、等同替换、改进等,均应

包含在本说明书的权利要求范围之内。