(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 109471879 A
(43)申请公布日 2019.03.15

(21)申请号 201811248981.6

(22)申请日 2018.10.25

(71)申请人 珠海天燕科技有限公司
地址 519085 广东省珠海市唐家湾镇哈工大路1号1栋E301-27

(72)发明人 滕朝垒 李涛

(74)专利代理机构 北京国昊天诚知识产权代理有限公司 11315
代理人 姜凤岩 南霆

(51)Int.Cl.
G06F 16/2457(2019.01)
G06F 16/2455(2019.01)
G06F 16/27(2019.01)

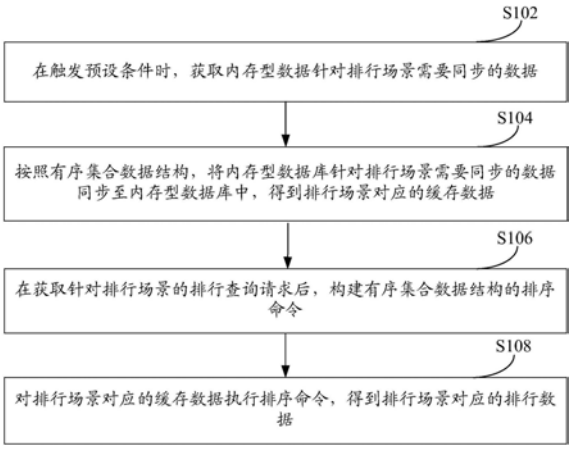
权利要求书3页 说明书11页 附图4页

(54)发明名称

一种数据排行的辅助方法及装置

(57)摘要

本申请实施例涉及一种数据排行的辅助方法及装置。方法包括:在触发预设条件时,获取内存型数据针对排行场景需要同步的数据;按照有序集合数据结构,将所述内存型数据库针对排行场景需要同步的数据同步至所述内存型数据库中,得到所述排行场景对应的缓存数据;在获取针对所述排行场景的排行查询请求后,构建有序集合数据结构的排序命令;对所述排行场景对应的缓存数据执行所述排序命令,得到所述排行场景对应的排行数据。本申请在满足高并发的排行查询要求的前提下,不依赖于业务数据库提供实时的排行数据。



CN 109471879 A

1.一种数据排行的辅助方法,其特征在于,包括:

在触发预设条件时,获取内存型数据针对排行场景需要同步的数据;

按照有序集合数据结构,将所述内存型数据库针对排行场景需要同步的数据同步至所述内存型数据库中,得到所述排行场景对应的缓存数据;

在获取针对所述排行场景的排行查询请求后,构建有序集合数据结构的排序命令;

对所述排行场景对应的缓存数据执行所述排序命令,得到所述排行场景对应的排行数据。

2.如权利要求1所述的方法,其特征在于,

所述有序集合数据结构为ZSet数据结构,所述排行场景中的排行周期作为所述ZSet数据结构的键key,所述排行场景中的排行对象作为所述ZSet数据结构的成员member,所述排行场景中的排行属性值作为所述ZSet数据结构的得分score;

其中,所述排序命令用于以key为约束条件,对member按照score的大小进行排序。

3.如权利要求2所述的方法,其特征在于,

所述内存型数据库针对排行场景所需要向业务数据库同步的数据包括目标排行对象的排行属性值对应的变量值,所述目标排行对象属于所述排行场景中的排行对象;

按照有序集合数据结构,将所述内存型数据库针对排行场景需要同步的数据同步至所述内存型数据库中,包括:

若所述内存型数据库的key存在当前排行周期的字段值,则基于所述变量值,对所述目标排行对象在当前排行周期对应所述内存型数据库的score中的字段值进行修改;

若所述内存型数据库的key不存在当前排行周期的字段值,则将当前排行周期的字段配置至所述内存型数据库的key中,并基于所述变量值,对所述目标排行对象在上一排行周期对应所述内存型数据库的score中的字段值进行计算,并将计算结果写入至所述目标排行对象在当前排行周期对应所述内存型数据库的score中。

4.如权利要求3所述的方法,其特征在于,

按照有序集合数据结构,将所述内存型数据库针对排行场景需要同步的数据同步至所述内存型数据库中,还包括:

若所述内存型数据库的member未存在所述目标排行对象的字段值,则将所述目标排行对象的字段值配置至所述内存型数据库的member中。

5.如权利要求4所述的方法,其特征在于,

所述排行场景中的排行对象均对应有业务系统配置的数值表示的编号;

将所述目标排行对象的字段值配置至所述内存型数据库的member中,包括:

构建排行对象的字段值的配置过程,所述配置过程用于使用一既定数值减去排行对象对应的编号,并将计算结果作为排行对象的字段值配置至所述内存型数据库的member中;其中,所述既定数值大于所述业务系统为排行对象配置的编号的数值上限;

基于所述配置过程,将所述目标排行对象的字段值配置至所述内存型数据库的member中。

6.如权利要求5所述的方法,其特征在于,还包括:

构建排行对象的字段值的读取过程,所述读取过程用于使用所述既定数值减去排行对象在所述内存型数据库的member中的字段值,得到排行对象对应的编号;其中排行对象对

应的编号作为排行对象的字段值的读取结果。

7.如权利要求2所述的方法,其特征在于,

构建有序集合数据结构的排序命令,包括:

若所述内存型数据库的key存在当前排行周期的字段值,则构建以当前排行周期的字段值为约束条件,对member按照score的大小进行排序的排序命令;

若所述内存型数据库的key不存在当前排行周期的字段值,则构建以上一排行周期的字段值为约束条件,对member按照score的大小进行排序的排序命令。

8.如权利要求1-7中任一项所述的方法,其特征在于,

所述预设条件包括:业务系统针对所述排行场景对应的业务数据发生变动;

获取内存型数据库针对排行场景需要同步的数据,包括:

基于从业务系统中获取到的业务系统针对所述排行场景对应的业务数据发生变动的数据,确定出内存型数据库针对排行场景需要同步的数据。

9.如权利要求1-7中任一项所述的方法,其特征在于,

所述预设条件包括:预设置的执行周期;

获取内存型数据库针对排行场景需要同步的数据,包括:

基于异步线程,从业务数据库中获取所述内存型数据库针对排行场景需要同步的数据。

10.一种数据排行的辅助装置,其特征在于,包括:

获取模块,用于在触发预设条件时,获取内存型数据针对排行场景需要同步的数据;

同步模块,用于按照有序集合数据结构,将所述内存型数据库针对排行场景需要同步的数据同步至所述内存型数据库中,得到所述排行场景对应的缓存数据;

构建模块,用于在获取针对所述排行场景的排行查询请求后,构建有序集合数据结构的排序命令;

执行模块,用于对所述排行场景对应的缓存数据执行所述排序命令,得到所述排行场景对应的排行数据。

11.如权利要求10所述的装置,其特征在于,

所述有序集合数据结构为ZSet数据结构,所述排行场景中的排行周期作为所述ZSet数据结构的键key,所述排行场景中的排行对象作为所述ZSet数据结构的成员member,所述排行场景中的排行属性值作为所述ZSet数据结构的得分score;

其中,所述排序命令用于以key为约束条件,对member按照score的大小进行排序。

12.如权利要求11所述的装置,其特征在于,

所述内存型数据库针对排行场景所需要向业务数据库同步的数据包括目标排行对象的排行属性值对应的变量值,所述目标排行对象属于所述排行场景中的排行对象;

所述同步模块具体用于:

若所述内存型数据库的key存在当前排行周期的字段值,则基于所述变量值,对所述目标排行对象在当前排行周期对应所述内存型数据库的score中的字段值进行修改;

若所述内存型数据库的key不存在当前排行周期的字段值,则将当前排行周期的字段配置至所述内存型数据库的key中,并基于所述变量值,对所述目标排行对象在上一排行周期对应所述内存型数据库的score中的字段值进行计算,并将计算结果写入至所述目标排

行对象在当前排行周期对应所述内存型数据库的score中。

13.如权利要求12所述的装置,其特征在于,

所述同步模块还具体用于:

若所述内存型数据库的member未存在所述目标排行对象的字段值,则将所述目标排行对象的字段值配置至所述内存型数据库的member中。

14.如权利要求13所述的装置,其特征在于,

所述排行场景中的排行对象均对应有业务系统配置的数值表示的编号;

所述同步模块具体用于:

构建排行对象的字段值的配置过程,所述配置过程用于使用一既定数值减去排行对象对应的编号,并将计算结果作为排行对象的字段值配置至所述内存型数据库的member中;其中,所述既定数值大于所述业务系统为排行对象配置的编号的数值上限;

基于所述配置过程,将所述目标排行对象的字段值配置至所述内存型数据库的member中。

15.如权利要求12所述的装置,其特征在于,

所述构建模块具体用于:

若所述内存型数据库的key存在当前排行周期的字段值,则构建以当前排行周期的字段值为约束条件,对member按照score的大小进行排序的排序命令;

若所述内存型数据库的key不存在当前排行周期的字段值,则构建以上一排行周期的字段值为约束条件,对member按照score的大小进行排序的排序命令。

一种数据排行的辅助方法及装置

技术领域

[0001] 本申请实施例涉及数据处理技术领域，尤其涉及一种数据排行的辅助方法及装置。

背景技术

[0002] 传统的业务系统是从关系型的业务数据库中确定出排行数据。这种方式严重依赖于业务数据库的性能。在高并发的排行查询情景下，会造成业务数据库没有资源提供连接，导致查询失败。

[0003] 针对这一问题，现有的解决方式是基于异步线程，将从业务数据库中确定出的排行数据缓存在内存型数据库中，由内存型数据库负责提供排行数据。但异步线程采用定期执行机制，无法保证排行数据的时效性；此外，在业务数据库下线或异常的场景中，比如对业务数据库进行性能扩展时，内存型数据库将无法得到更新，导致缓存的排行数据严重滞后，极大影响了用户的体验。

[0004] 有鉴于此，如何在满足高并发的排行查询要求的前提下，提供实时的排行数据，是本申请所要解决的技术问题。

发明内容

[0005] 本申请实施例目的是提供一种数据排行的辅助方法及装置，能够在满足高并发的排行查询要求的前提下，提供实时的排行数据。

[0006] 为了实现上述目的，本申请实施例是这样实现的：

[0007] 第一方面，提供一种数据排行的辅助方法，包括：

[0008] 在触发预设条件时，获取内存型数据针对排行场景需要同步的数据；

[0009] 按照有序集合数据结构，将所述内存型数据库针对排行场景需要同步的数据同步至所述内存型数据库中，得到所述排行场景对应的缓存数据；

[0010] 在获取针对所述排行场景的排行查询请求后，构建有序集合数据结构的排序命令；

[0011] 对所述排行场景对应的缓存数据执行所述排序命令，得到所述排行场景对应的排行数据。

[0012] 第二方面，提供了一种数据排行的辅助装置，包括：

[0013] 获取模块，用于在触发预设条件时，获取内存型数据针对排行场景需要同步的数据；

[0014] 同步模块，用于按照有序集合数据结构，将所述内存型数据库针对排行场景需要同步的数据同步至所述内存型数据库中，得到所述排行场景对应的缓存数据；

[0015] 构建模块，用于在获取针对所述排行场景的排行查询请求后，构建有序集合数据结构的排序命令；

[0016] 执行模块，用于对所述排行场景对应的缓存数据执行所述排序命令，得到所述排

行场景对应的排行数据。

[0017] 本申请实施例中，在排行场景相关的业务数据发生变化时，按照有序集合数据结构，将需要针对排行场景同步的数据同步至内存型数据库，从而利用有序集合数据结构的排序命令，对内存型数据库中针对排行场景的缓存数据进行排序，得到排行场景对应的排行数据。一方面，不需要依赖异步线程从业务数据库中获取排行数据，因此具有较高的实时性，且在业务数据库下线或发生异常时，也不会受到影响；另一方面，基于内存型数据库强大的吞吐量和有序集合数据结构超高的排序查询效率，还能够满足高并发的排行查询要求。

附图说明

[0018] 为了更清楚地说明本申请实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本申请实施例中记载的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动性的前提下，还可以根据这些附图获得其他的附图。

[0019] 图1为本申请实施例提供的数据排行的辅助方法的流程示意图；

[0020] 图2为本申请实施例提供的数据排行的辅助方法在实际应用中的第一个流程示意图；

[0021] 图3为本申请实施例提供的数据排行的辅助方法在实际应用中的第二个流程示意图；

[0022] 图4为本申请实施例提供的数据排行的辅助方法在实际应用中的第三个流程示意图；

[0023] 图5为本申请实施例提供的数据排行的辅助装置的结构示意图。

具体实施方式

[0024] 为了使本技术领域的人员更好地理解本申请中的技术方案，下面将结合本申请实施例中的附图，对本申请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申请一部分实施例，而不是全部的实施例。基于本申请中的实施例，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，都应当属于本申请保护的范围。

[0025] 如前所述，现有技术是通过定期执行的异步线程将从业务数据库中确定出的排行数据缓存至内存行数据库中，并由内存型数据库负责提供排行数据。这种方式在时效性无法满足用户的要求，比如一些实际业务场景中，实时的排行数据能够刺激用户的参与感，对于用户来讲是影响体验的重要因素。此外，内存型数据的更新依赖业务数据库，当业务数据库下线或者处于异常时，就会导致内存型数据的缓存数据存在严重的滞后，也是用户所无法接受的。

[0026] 有鉴于此，本申请实施例提供一种可解决上述问题的技术方案。

[0027] 一方面，本申请实施例提供一种数据排行的辅助方法，如图1所示，包括：

[0028] 步骤102，在触发预设条件时，获取内存型数据针对排行场景需要同步的数据；

[0029] 针对步骤102而言：

[0030] 可以在业务系统针对排行场景对应的业务数据发生变动时，获取内存型数据针对排行场景需要同步的数据。

[0031] 比如，上述业务系统为金融系统，对应的排行场景可以是对用户按照余额的大小进行排行。每当用户产生增加或者减少的金额时，即可获取内存型数据库针对排行场景所需要同步的数据。

[0032] 步骤104，按照有序集合数据结构，将内存型数据库针对排行场景需要同步的数据同步至内存型数据库中，得到排行场景对应的缓存数据；

[0033] 针对步骤104而言：

[0034] 数据排序是有序集合数据结构的基础功能，本申请实施例将排行场景的缓存数据按照有序集合数据结构存储至内存数据库，即可利用有序集合数据结构的排序功能，对缓存数据进行排序，得到排行数据。

[0035] 步骤106，在获取针对排行场景的排行查询请求后，构建有序集合数据结构的排序命令；

[0036] 针对步骤106而言：

[0037] 本步骤可以基于有序集合数据结构的排序命令，实现多种方式的排序，比如实现分页排序、范围排序等。

[0038] 本申请实施例不对具体的排序方式进行限定。

[0039] 步骤108，对排行场景对应的缓存数据执行排序命令，得到排行场景对应的排行数据。

[0040] 针对步骤108而言：

[0041] 应理解，排行数据是对内存型数据库中的缓存数据执行排序命令后获得的，并不是像现有技术中那样，从业务数据库中获取。因此内存型数据库可以不依赖业务数据库提供排行数据。

[0042] 本申请实施例中，在排行场景相关的业务数据发生变化时，按照有序集合数据结构，将需要针对排行场景同步的数据同步至内存型数据库，从而利用有序集合数据结构的排序命令，对内存型数据库中针对排行场景的缓存数据进行排序，得到排行场景对应的排行数据。一方面，不需要依赖异步线程从业务数据库中获取排行数据，因此具有较高的实时性，且在业务数据库下线或发生异常时，也不会受到影响；另一方面，基于内存型数据库强大的吞吐量和有序集合数据结构超高的排序查询效率，还能够满足高并发的排行查询要求。

[0043] 下面对本申请实施例的数据排行的辅助方法进行详细介绍。

[0044] 本申请实施例具体按照ZSet数据结构(一种有序集合数据结构)，将需要针对排行场景同步的数据同步至内存型数据库中。

[0045] ZSet数据结构属于key-value其中一种表现方式。它包括：key、member和score三种元素，每个key下，member对应有一个score。ZSet数据结构的特点是可以基于score值的大小对member排序。

[0046] 基于ZSet数据结构的特点，本申请实施例对排行场景定义了排行周期、排行对象和排行属性值三种属性，分别作为ZSet数据结构的键key、成员member和得分score。

[0047] 后续对同步后的缓存数据执行ZSet数据结构的排序命令，能够以排行周期为约束

条件,对排行对象按照排行属性值的大小进行排序,得到排行场景相对应的排行数据。

[0048] 其中,内存型数据库针对排行场景所需要向业务数据库同步的数据包括:目标排行对象的排行属性值对应的变量值(目标排行对象属于排行场景中的排行对象);

[0049] 在执行上述步骤104时,具体包括以下步骤:

[0050] 步骤1041,若内存型数据库的key存在当前排行周期的字段值,则基于变量值,对目标排行对象在当前排行周期对应内存型数据库的score中的字段值进行修改;

[0051] 步骤1042,若内存型数据库的key不存在当前排行周期的字段值,则将当前排行周期的字段配置至内存型数据库的key中,并基于变量值,对目标排行对象在上一排行周期内存型数据库的score中的字段值进行计算,并将计算结果写入至目标排行对象在当前排行周期对应内存型数据库的score中。

[0052] 应理解,本申请实施例中的内存型数据库可以保存多个排行周期的排行数据,即内存型数据库的Key中含有多个排行周期的字段值。

[0053] 或者,本申请实施例中的内存型数据库也可以指保留最近一个排行周期的排行数据。即,在执行完上述步骤1042后,还进一步删除掉目标排行对象在上一排行周期对应key、member、score中的字段值。

[0054] 当然,若内存型数据库的member未存在目标排行对象的字段值,则本申请实施例还可以进一步将目标排行对象的字段值配置至内存型数据库的member中。

[0055] 需要说明的是,ZSet数据结构的排序指令是先根据score的字段值大小对member进行排序,在score字段值大小相同时,再进一步根据member字段值以从大到小的固定顺序对member进行排序。

[0056] 因此,本申请实施例在为内存型数据库的member配置排行对象时,可以通过设置排行对象的字段值的大小,以实现更具体的排序效果。

[0057] 比如,本申请排行场景中的排行对象分为高优先级排行对象和低优先级排行对象。

[0058] 在排行属性值相同时,高优先级排行对象的排名应位于低优先级排行对象的排名之前。

[0059] 对应地,本申请实施例可以使用数值表示的编号代表排行对象,高优先级的排行对象的编号数值大于低优先级的排行对象的编号数值。

[0060] 在为内存型数据库的member配置排行对象时,可以将排行对象的编号作为排行对象的字段值写入至member中。

[0061] 以业务系统中的用户作为排行对象为示例,在实际场景中,用户之间的优先级体现在业务系统侧,业务系统一般会给其用户配置作为用户标识的编号,以区分不同的优先级。

[0062] 因此,为简化方案,本申请实施例可以复用业务系统为用户配置的编号,作为用户在内存型数据库的member中的字段值。

[0063] 当然,若业务系统给其用户配置的用户标识不是以数值表示的编号,则本申请实施例再进一步基于预设的编译算法,将给户配置的用户标识编译成以数值表示的编号。

[0064] 应理解,用户的优先级越高,则编译算法编译的编号的数值越大;用户的优先级越低,则编译算法编译的编号的数值越小。

[0065] 这里需要说明的是,ZSet数据结构的排序指令只能以member的字段值从大到小的固定顺序进行排序。

[0066] 用户对应的编号大小取决于业务系统,实际中可能确定到的高优先级用户的编号会小于低先级用户的编号,这样会导致在基于ZSet数据结构的排序指令进行排序时,低优先级用户的排名位于高优先级的排名之前。为避免这一问题发生,可以使用一既定的较大值减去用户对应的编号来作为用户在member中的字段值。

[0067] 比如业务系统为高优先级用户配置的编号值为1,低优先级用户配置的编号值为2,则既定数值可以取10,高优先级用户在member中的字段值为10-1＝9,低优先级用户在member中的字段值为10-2＝8。由于高优先级在member中的字段值大于低优先级用户在member中的字段值(9＞8),因此在两者排行属性值相同的情况下,高优先级用户会排在低优先级用户之前。

[0068] 在具体实现时,本申请实施例可以构建排行对象的字段值的配置过程,该配置过程用于使用既定数值减去排行对象对应的编号,并将计算结果作为排行对象的字段值配置至内存型数据库的member中;其中,既定数值应大于业务系统为排行对象配置的编号的数值上限,以避免出现负值的情况。

[0069] 在获得配置过程后,即可基于上述配置过程,将任意目标排行对象的字段值配置至内存型数据库的member中。

[0070] 对应地,从内存型数据库中读取目标排行对象,则需要将目标排行对象在member中字段值还原回对应的编号,从而基于编号正确识别出目标排行对象。即,本申请实施例的辅助方法还包括:

[0071] 构建排行对象的字段值的读取过程,该读取过程用于使用上既定数值减去排行对象在内存型数据库的member中的字段值,得到排行对象对应的编号;其中,排行对象对应的编号作为排行对象的字段值的读取结果。

[0072] 比如,目标排行对象的编码为10,既定值为100,基于上述配置过程,目标排行对象在内存型数据库的member中的字段值为100-10＝90。

[0073] 通过上述读取过程在读取标排行对象在内存型数据库的member中的字段值时,可以使用既定值100减去目标排行对象在member中的字段值90,准确还原回目标排行对象的编码。

[0074] 应理解,本申请实施例的方案可以根据业务系统对用户标识的配置特点,决定是否构建排行对象针对member的配置过程和读取过程。

[0075] 在按照ZSet数据结构,将内存型数据库针对排行场景需要同步的数据同步至内存型数据库中后,即可通过ZSet数据结构的排序命令对缓存数据进行排序,以得到排行数据。

[0076] 具体地,在获取排行查询请求后,先确定当前排行周期的字段值,并通过当前排行周期的字段值来判断内存数据库是否存在对应当前排行周期的缓存数据。

[0077] 若内存型数据库的key存在当前排行周期的字段值,则构建以当前排行周期的字段值为约束条件,对member按照score的大小进行排序的排序命令;

[0078] 若内存型数据库的key不存在当前排行周期的字段值,则构建以上一排行周期的字段值为约束条件,对member按照score的大小进行排序的排序命令。

[0079] 在实际应用中,本申请实施例可以构建ZRank命令或ZRevarange命令,对缓存数据

实现排序。

[0080] 以ZRank命令为例,本申请可以基于ZRank指令根据排行对象的字段值,查询对应的排行位置。ZSet内部的skiplist跳跃表以时间复杂度为(平均O(logN)最坏O(N))跳跃表节点进行数据跨度,查询获得排行数据。

[0081] 以ZRevarange命令为例,可以直接使用ZrevRange命令实现范围查询,例如1-100名按照排行属性值从大到小的顺序对排行对象进行排序,获得前100名对排行对象的排行数据。

[0082] 应理解,本申请实施例基于ZSet数据结构能够以key为约束条件,按照score对member进行排序的特点,将排行场景中的排行周期、排行对象和排行属性值分别作为key、member和score,从而实现在某一排行周期内,按照排行属性值对排行对象进行排序的功能。key、member和score之间映射关系简单,即便当数据量级庞大时也能够实现高效快速的排序,性能卓越,是其他key-value的数据结构所无法企及的。

[0083] 下面结合实际应用,对申请实施例的辅助方法的流程进行介绍。

[0084] 示例性地,本实际应用中的业务系统为金属系统,配置有Mysql数据库和Redis数据库。其中,Mysql数据库作为业务数据库;Redis数据库作为内存型数据库,针对排行场景,采用ZSet数据结构。

[0085] 假设排行场景是针对用户的余额进行排行,排行周期为周,作为ZSet数据结构的key;排行对象为用户,作为ZSet数据结构的member;排行属性值为余额值,作为ZSet数据结构的score。

[0086] 本实际应用可以采用两种方式来对Redis数据库进行更新。

[0087] 一种方式是采用定期执行的异步线程,从Mysql数据库中获取针对排行场景需要同步的数据,该方式如图2所示,主要流程包括:

[0088] 在满足定期执行的触发条件时,从Mysql数据库中获取全量的流水数据。

[0089] 确定当前排行周期的key值(字段值),即本周对应的Key值。

[0090] 基于前排行周期的key值,通过全量的流水数据对Redis数据库进行数据同步。

[0091] 清空Redis数据库中上一排行周期的缓存数据。

[0092] 由于本方式中,Redis数据库的同步数据来源于业务数据库,因此可以从业务数据库获取全量的流水数据对Redis数据库中所有用户进行余额更新,在更新完成后,还可以对上一排行周期的缓存数据进行删除。

[0093] 另一种方式是,业务系统中一有用户的余额发生变化,就对Redis数据库进行同步,该方式如图3所示,主要流程包括:

[0094] 在业务系统中的目标用户的余额发生变化时,从业务系统中获取目标用户余额的变量值。

[0095] 确定当前排行周期的key值(即本周对应的key值)。

[0096] 判断redis数据库是否存在当前排行周期的key值,以及redis数据库是否存目标用户的member值。

[0097] 若redis数据库不存在当前排行周期的key值,则调用Zadd指令在redis数据库的key中配置当前排行周期的key值。

[0098] 若redis数据库不存在目标用户的member值,则调用ZSet数据结构的Zadd指令在

redis数据库的member中配置目标用户的member值,以及基于目标用户的余额的变量值,配置目标用户在当前周期的score中的字段值。

[0099] 若redis数据库存在当前排行周期的key值,以及目标用户的member值,则调用ZSet数据结构的incrementScorem命令,按照该用户余额的变量值,对该用户在当前周期对应的score值进行调整。

[0100] 在通过图2和图3所示的方法对Redis数据库进行更新后,即可基于排序命令获取Redis数据库的排行数据,对应的流程如图4所示,包括:

[0101] 获取以余额大小对用户进行排行的排行查询请求。

[0102] 确定当前排行周期的key值(即本周对应的key值)。

[0103] 判断redis数据库是否存在当前排行周期的key值。

[0104] 若redis数据库存在当前排行周期的key值,则基于当前排行周期的key值,对redis数据库的缓存数据执行ZSet数据结构的分页排序指令,得到分页显示的排行数据;并将分页显示的排行数据进行反序列化处理,以及解析、组装,以在业务系统中显示。

[0105] 若redis数据库是不存在当前排行周期的key值(当前时刻为止,本周的排行数据与上一周排行数据未发生变化),则确定上一排行周期的kye值,并判断redis数据库是否存在上一排行周期的key值;

[0106] 若redis数据库存在上一排行周期的key值,则基于上一排行周期的key值,对redis数据库的缓存数据执行ZSet数据结构的分页排序指令,到分页显示的排行数据;并将分页显示的排行数据进行反序列化处理,以及解析、组装,以在业务系统中显示。若redis数据库不存在上一排行周期的key值,则执行异常处理(如报警、记录异常日志等)。

[0107] 基于上述设计,本申请实施例提供的数据分页排序方法通过图2所示的步骤,对Redis数据库进行定期的全量更新,并基于图3所示的步骤,对Redis数据库进行实时的增量更新,从而保证Redis数据库提供的排行数据与实际情景一致,以最小的性能消耗来向外提供排行查询服务。同时只需要增加Redis数据库的key即可实现更多功能的扩展,比如同时实现周排行、日排行、月排行等功能。

[0108] 以上是对本申请实施例的数据排行的辅助方法的介绍。应理解,本申请实施例针对排行场景并不限于是仅按照ZSet数据结构对内存型数据库进行数据同步,还可以是在该基础之上,进一步按照String数据结构和Hash数据结构记录排行场景的缓存数据,以实现更多的拓展功能。此外,本申请实施例中的内存型数据库也不限于是仅记录一种排行场景对应的缓存数据,可以针对不同的排行场景分别记录相对应的缓存数据。

[0109] 与之对应地,本申请实施例还提供一种数据排行的辅助装置,如图5所示,包括:

[0110] 获取模块51,用于在触发预设条件时,获取内存型数据针对排行场景需要同步的数据;

[0111] 具体地,获取模块51可以在业务系统针对排行场景对应的业务数据发生变动时,获取内存型数据针对排行场景需要同步的数据。

[0112] 比如,上述业务系统为金融系统,对应的排行场景可以是对用户按照余额的大小进行排行。获取模块51在用户产生增加或者减少的金额时,即可获取内存型数据库针对排行场景所需要同步的数据。

[0113] 同步模块52,用于按照有序集合数据结构,将所述内存型数据库针对排行场景需

要同步的数据同步至所述内存型数据库中，得到所述排行场景对应的缓存数据；

[0114] 其中，数据排序是有序集合数据结构的基础功能，将排行场景的缓存数据按照有序集合数据结构存储至内存数据库，即可利用有序集合数据结构的排序功能，对缓存数据进行排序，得到排行数据。

[0115] 构建模块53，用于在获取针对所述排行场景的排行查询请求后，构建有序集合数据结构的排序命令；

[0116] 其中，构建模块53可以基于有序集合数据结构的排序命令，实现多种方式的排序，比如实现分页排序、范围排序等。本申请实施例并不对具体的排序方式进行限定。

[0117] 执行模块54，用于对所述排行场景对应的缓存数据执行所述排序命令，得到所述排行场景对应的排行数据。

[0118] 其中，排行数据是对内存型数据库中的缓存数据执行排序命令后获得的，并不是像现有技术中那样，从业务数据库中获取。因此内存型数据库可以不依赖业务数据库提供排行数据。

[0119] 本申请实施例中，在排行场景相关的业务数据发生变化时，按照有序集合数据结构，将需要针对排行场景同步的数据同步至内存型数据库，从而利用有序集合数据结构的排序命令，对内存型数据库中针对排行场景的缓存数据进行排序，得到排行场景对应的排行数据。一方面，不需要依赖异步线程从业务数据库中获取排行数据，因此具有较高的实时性，且在业务数据库下线或发生异常时，也不会受到影响；另一方面，基于内存型数据库强大的吞吐量和有序集合数据结构超高的排序查询效率，还能够满足高并发的排行查询要求。

[0120] 下面对本申请实施例的数据排行的辅助装置进行详细介绍。

[0121] 本申请实施例具体按照ZSet数据结构(一种有序集合数据结构)，将需要针对排行场景同步的数据同步至内存型数据库中。

[0122] ZSet数据结构属于key-value其中一种表现方式。它包括：key、member和score三种元素，每个key下，member对应有一个score。ZSet数据结构的特点是可以基于score值的大小对member排序。

[0123] 基于ZSet数据结构的特点，本申请实施例对排行场景定义了排行周期、排行对象和排行属性值三种属性，分别作为ZSet数据结构的键key、成员member和得分score。

[0124] 后续对同步后的缓存数据执行ZSet数据结构的排序命令，能够以排行周期为约束条件，对排行对象按照排行属性值的大小进行排序，得到排行场景相对应的排行数据。

[0125] 其中，内存型数据库针对排行场景所需要向业务数据库同步的数据包括：目标排行对象的排行属性值对应的变量值(目标排行对象属于排行场景中的排行对象)；同步模块52具体用于：

[0126] 若所述内存型数据库的key存在当前排行周期的字段值，则基于所述变量值，对所述目标排行对象在当前排行周期对应所述内存型数据库的score中的字段值进行修改；

[0127] 若所述内存型数据库的key不存在当前排行周期的字段值，则将当前排行周期的字段配置至所述内存型数据库的key中，并基于所述变量值，对所述目标排行对象在上一排行周期对应所述内存型数据库的score中的字段值进行计算，并将计算结果写入至所述目标排行对象在当前排行周期对应所述内存型数据库的score中。

[0128] 应理解，本申请实施例中的内存型数据库可以保存多个排行周期的排行数据，即内存型数据库的Key中含有多个排行周期的字段值。

[0129] 或者，本申请实施例中的内存型数据库也可以指保留最近一个排行周期的排行数据。即，同步模块52在写入目标排行对象在当前排行周期对应内存型数据库的score中后，还进一步删除掉目标排行对象在上一排行周期对应key、member、score中的字段值。

[0130] 当然，若内存型数据库的member未存在目标排行对象的字段值，则本申请实施例同步模块52还可以进一步将目标排行对象的字段值配置至内存型数据库的member中。

[0131] 需要说明的是，ZSet数据结构的排序指令是先根据score的字段值大小对member进行排序，在score字段值大小相同时，再进一步根据member字段值以从大到小的固定顺序对member进行排序。

[0132] 因此，本申请实施例的同步模块52在为内存型数据库的member配置排行对象时，可以通过设置排行对象的字段值的大小，以实现更具体的排序效果。

[0133] 比如，本申请排行场景中的排行对象分为高优先级排行对象和低优先级排行对象。

[0134] 在排行属性值相同时，高优先级排行对象的排名应位于低优先级排行对象的排名之前。

[0135] 对应地，本申请实施例可以使用数值表示的编号代表排行对象，高优先级的排行对象的编号数值大于低优先级的排行对象的编号数值。

[0136] 同步模块52在为内存型数据库的member配置排行对象时，可以将排行对象的编号作为排行对象的字段值写入至member中。

[0137] 以业务系统中的用户作为排行对象为示例，在实际场景中，用户之间的优先级体现在业务系统侧，业务系统一般会给其用户配置作为用户标识的编号，以区分不同的优先级。

[0138] 同步模块52可以复用业务系统为用户配置的编号，作为用户在内存型数据库的member中的字段值。

[0139] 当然，若业务系统给其用户配置的用户标识不是以数值表示的编号，则同步模块52还可以进一步基于预设的编译算法，将给户配置的用户标识编译成以数值表示的编号。

[0140] 这里需要说明的是，ZSet数据结构的排序指令只能以member的字段值从大到小的固定顺序进行排序。

[0141] 用户对应的编号大小取决于业务系统，因此可能从业务系统侧确定到的高优先级用户的编号会小于低先级用户的编号，这样会导致在基于ZSet数据结构的排序指令进行排序时，低优先级用户的排名位于高优先级的排名之前。为避免这一问题发生，同步模块52可以使用一既定的较大值减去用户对应的编号来作为用户在member中的字段值。

[0142] 比如业务系统为高优先级用户配置的编号值为1，低优先级用户配置的编号值为2，则既定数值可以取10，高优先级用户在member中的字段值为10-1＝9，低优先级用户在member中的字段值为10-2＝8。由于高优先级在member中的字段值大于低优先级用户在member中的字段值(9＞8)，因此在两者排行属性值相同的情况下，高优先级用户会排在低优先级用户之前。

[0143] 在具体实现时，同步模块52可以构建排行对象的字段值的配置过程，该配置过程

用于使用一既定数值减去排行对象对应的编号，并将计算结果作为排行对象的字段值配置至内存型数据库的member中；其中，既定数值大于业务系统为排行对象配置的编号的数值上限。

[0144] 在获得配置过程后，同步模块52基于该配置过程，即可将任意目标排行对象的字段值配置至内存型数据库的member中。

[0145] 对应地，从内存型数据库中读取目标排行对象则需要将目标排行对象在member中字段值还原回对应的编号，从而正确识别出目标排行对象。即本申请实施例的同步模块52还具体用于：

[0146] 构建排行对象的字段值的读取过程，所述读取过程用于使用所述既定数值减去排行对象在所述内存型数据库的member中的字段值，得到排行对象对应的编号；其中排行对象对应的编号作为排行对象的字段值的读取结果。

[0147] 比如，目标排行对象的编码为10，既定值为100，基于上述配置过程，目标排行对象在内存型数据库的member中的字段值为100-10＝90。通过上述读取过程在读取标排行对象在内存型数据库的member中的字段值时，可以使用既定值100减去目标排行对象在member中的字段值90，准确还原回目标排行对象的编码。

[0148] 在实际应用中，可以根据业务系统对用户标识的配置特点，决定同步模块52是否构建排行对象针对member的配置过程和读取过程。

[0149] 在按照ZSet数据结构，将内存型数据库针对排行场景需要同步的数据同步至内存型数据库中后，构建模块34即可构建ZSet数据结构的排序命令。

[0150] 作为示例性介绍，构建模块34具体用于：

[0151] 在获取排行查询请求后，先确定当前排行周期的字段值，并通过当前排行周期的字段值来判断内存数据库是否存在对应的缓存数据。

[0152] 若内存型数据库的key存在当前排行周期的字段值，则构建以当前排行周期的字段值为约束条件，对member按照score的大小进行排序的排序命令；

[0153] 若内存型数据库的key不存在当前排行周期的字段值，则构建以上一排行周期的字段值为约束条件，对member按照score的大小进行排序的排序命令。

[0154] 在实际应用中，本申请实施例可以通过ZRank命令和ZRevarange命令对缓存数据实现分页排序。

[0155] 以ZRank命令为例，执行模块54可以基于ZRank指令根据排行对象的字段值，查询对应的排行位置。ZSet内部的skiplist跳跃表以时间复杂度为跳跃表节点进行数据跨度，查询获得排行数据。

[0156] 以ZRevarange命令为例，执行模块54可以直接使用ZrevRange命令实现范围查询，例如1-100名按照排行属性值从大到小的顺序对排行对象进行排序，获得前100名对排行对象的排行数据。

[0157] 此外，在实际应用中，本申请的获取模块51可以采用两种方式获取内存型数据库针对排行场景需要同步的数据。

[0158] 一种方式是获取模块51定期执行的异步线程，从业务数据库中获取针对排行场景需要同步的数据，该方式中，获取模块51需要触发的预设条件包括：预设置的异步线程的执行周期；

[0159] 另一种方式，业务系统中一有用户的余额发生变化，获取模块51就对Redis数据库进行同步。获取模块51需要触发的预设条件包括：业务系统针对所述排行场景对应的业务数据发生变动。获取模块51基于从业务系统中获取到的业务系统针对所述排行场景对应的业务数据发生变动的数据，确定出内存型数据库针对排行场景需要同步的数据。

[0160] 显然，本申请实施例提供的数据排行的辅助装置是本申请实施例上文提供的数据排行的辅助方法的执行主体，因此该辅助方法所能实现的技术效果，本申请实施例提供的辅助装置同样能够实现，比如图1、图2、图3和图4中所描述的步骤和功能。

[0161] 本申请实施例所提供的基于数据排行的辅助方法及装置的计算机程序产品，包括存储了程序代码的计算机可读存储介质，所述程序代码包括的指令可用于执行前面方法实施例中所述的步骤，具体实现可参见方法实施例，在此不再赘述。

[0162] 此外，本申请实施例还提供一种业务系统，该排行系统包括：

[0163] 内存型数据库以及上述图5所示的数据排行的辅助装置。

[0164] 基于与该辅助装置，本申请实施例的业务系统将排行场景对应的数据按照ZSet数据结构缓存至内存型数据库中。当接收到针对排行场景的排名查询指请求时，对内存型数据库的缓存数据执行排序命令，从而高效、快速地得到对应的排名数据，并响应。由于不依赖业务数据库，因此在业务数据库发生下线或者异常时，也能够保证提供实时的排名数据。在一些特殊的应用场景中，本申请实施例的业务系统可满足用户迫切获知当前排行数据的需求，从而调动用户的积极性和参与感。

[0165] 在本发明所提供的实施例中，应该理解到，所揭露的装置和方法，也可以通过其它的方式实现。以上所描述的装置和方法实施例仅仅是示意性的，例如，附图中的流程图和框图显示了根据本发明的多个实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或代码的一部分，所述模块、程序段或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意，在有些作为替换的实现方式中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个连续的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图和/或流程图中的每个方框、以及框图和/或流程图中的方框的组合，可以用执行规定的功能或动作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。另外，在本发明各个实施例中的各功能模块可以集成在一起形成一个独立的部分，也可以是各个模块单独存在，也可以两个或两个以上模块集成形成一个独立的部分。

[0166] 需要说明的是，在本文中，术语"包括"、"包含"或者其任何其它变体意在涵盖非排它性的包含，从而使得包括一系列要素的过程、方法、物品或者设备不仅包括那些要素，而且还包括没有明确列出的其它要素，或者是还包括为这种过程、方法、物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、物品或者设备中还存在另外的相同要素。

[0167] 以上所述仅为本申请的实施例而已，并不用于限制本申请。对于本领域技术人员来说，本申请可以有各种更改和变化。凡在本申请的精神和原理之内所作的任何修改、等同替换、改进等，均应包含在本申请的权利要求范围之内。

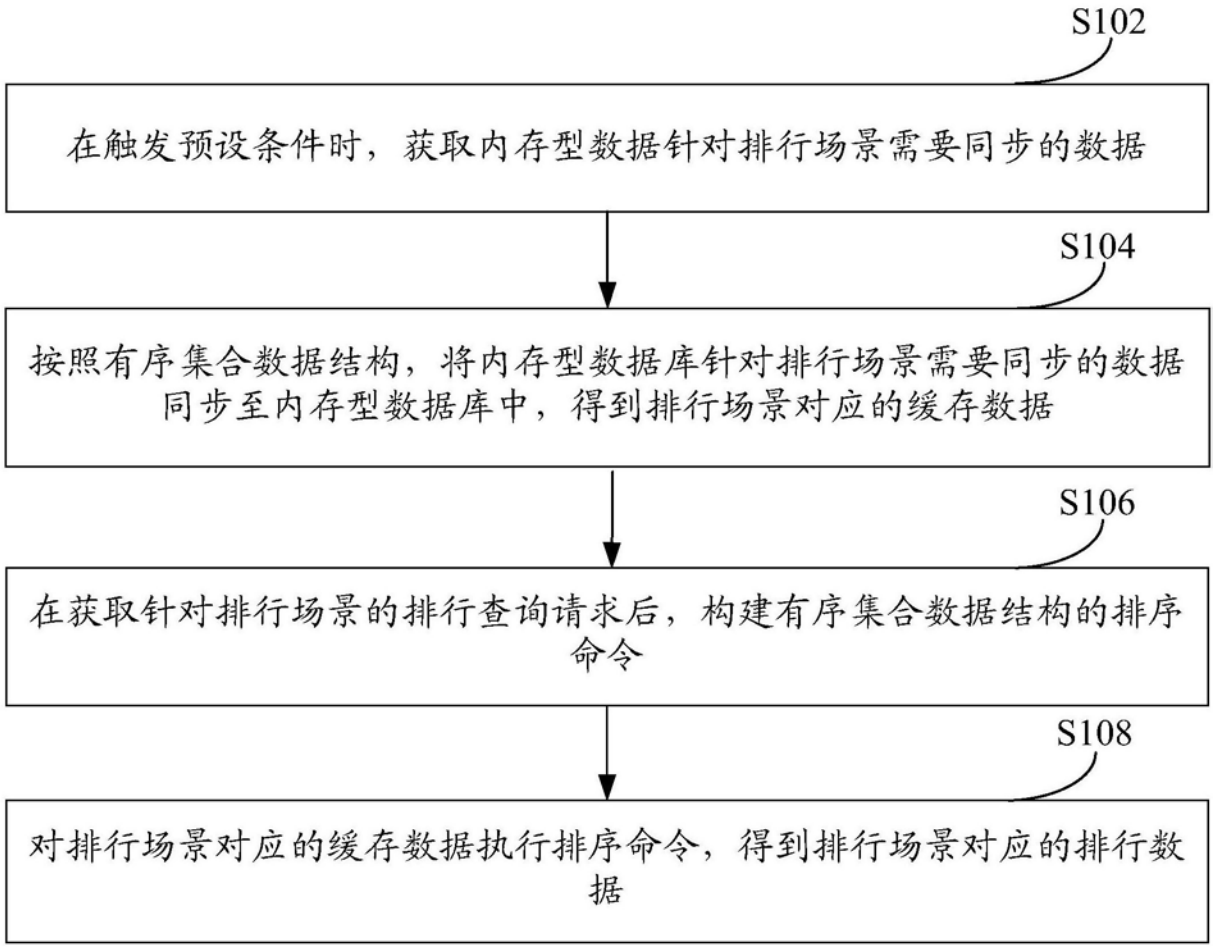

S102
在触发预设条件时，获取内存型数据针对排行场景需要同步的数据
S104
按照有序集合数据结构，将内存型数据库针对排行场景需要同步的数据同步至内存型数据库中，得到排行场景对应的缓存数据
S106
在获取针对排行场景的排行查询请求后，构建有序集合数据结构的排序命令
S108
对排行场景对应的缓存数据执行排序命令，得到排行场景对应的排行数据

图1

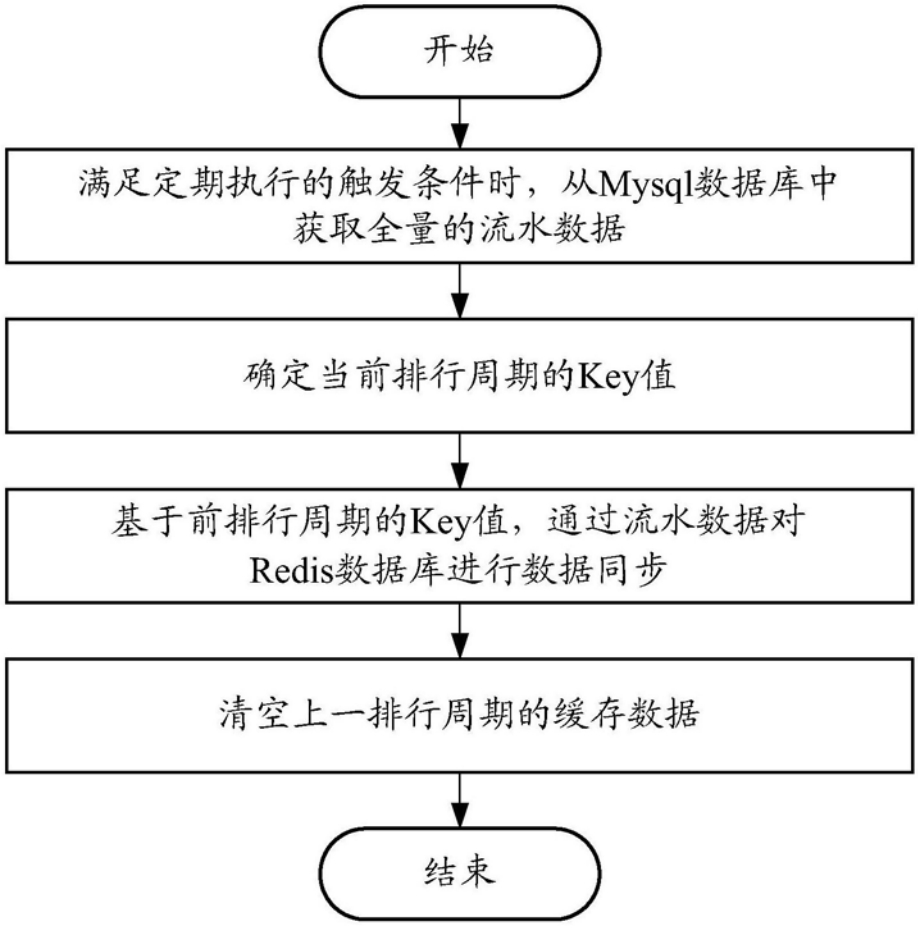

开始
满足定期执行的触发条件时，从Mysql数据库中获取全量的流水数据
确定当前排行周期的Key值
基于前排行周期的Key值，通过流水数据对Redis数据库进行数据同步
清空上一排行周期的缓存数据
结束

图2

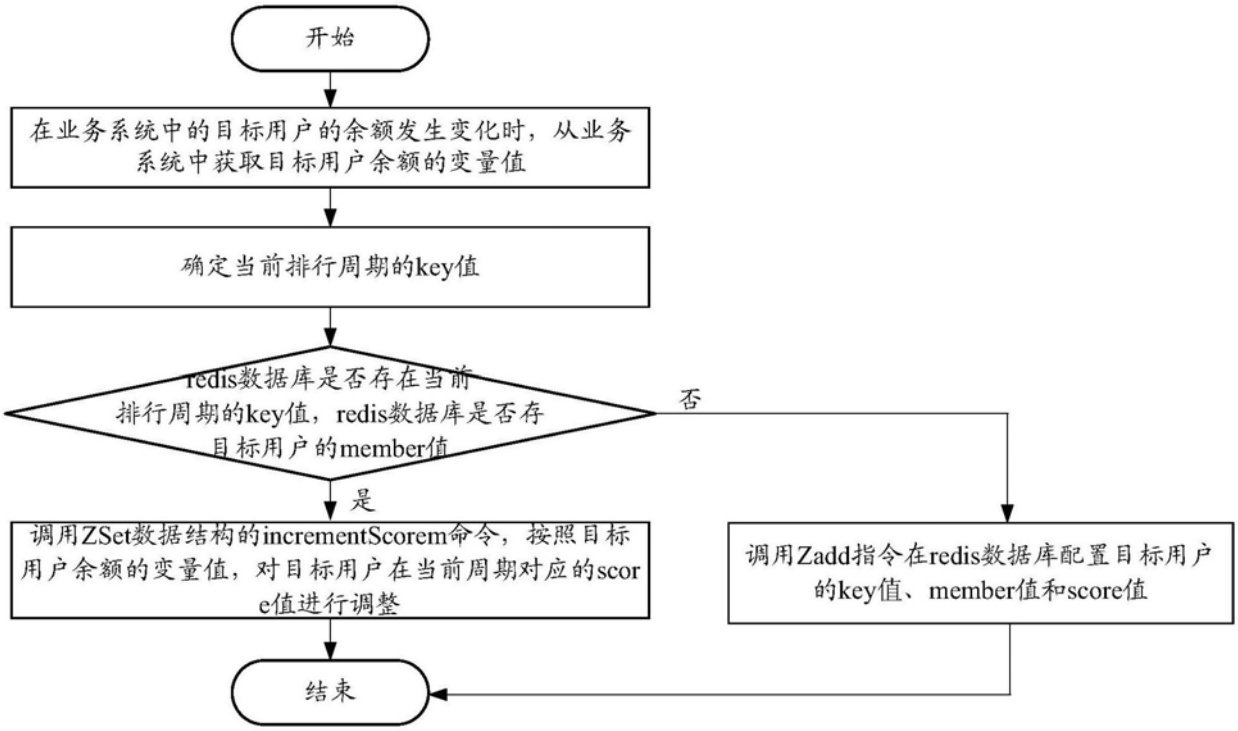

开始
在业务系统中的目标用户的余额发生变化时，从业务系统中获取目标用户余额的变量值
确定当前排行周期的key值
redis数据库是否存在当前排行周期的key值，redis数据库是否存目标用户的member值
否
是
调用ZSet数据结构的incrementScorem命令，按照目标用户余额的变量值，对目标用户在当前周期对应的score值进行调整
调用Zadd指令在redis数据库配置目标用户的key值、member值和score值
结束

图3

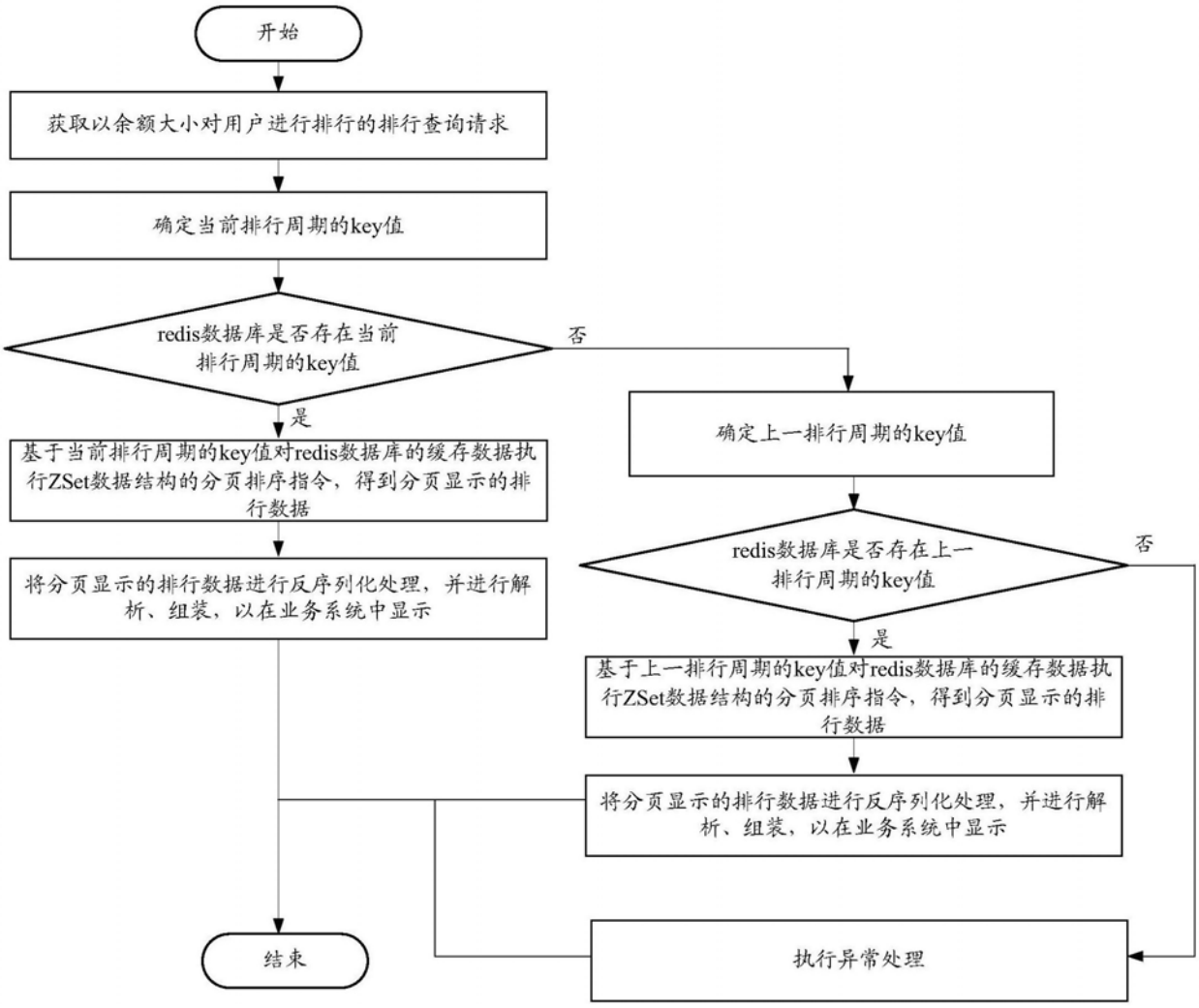

开始
获取以余额大小对用户进行排行的排行查询请求
确定当前排行周期的key值
redis数据库是否存在当前排行周期的key值
否
是
基于当前排行周期的key值对redis数据库的缓存数据执行ZSet数据结构的分页排序指令，得到分页显示的排行数据
将分页显示的排行数据进行反序列化处理，并进行解析、组装，以在业务系统中显示
结束
确定上一排行周期的key值
redis数据库是否存在上一排行周期的key值
否
是
基于上一排行周期的key值对redis数据库的缓存数据执行ZSet数据结构的分页排序指令，得到分页显示的排行数据
将分页显示的排行数据进行反序列化处理，并进行解析、组装，以在业务系统中显示
执行异常处理

图4

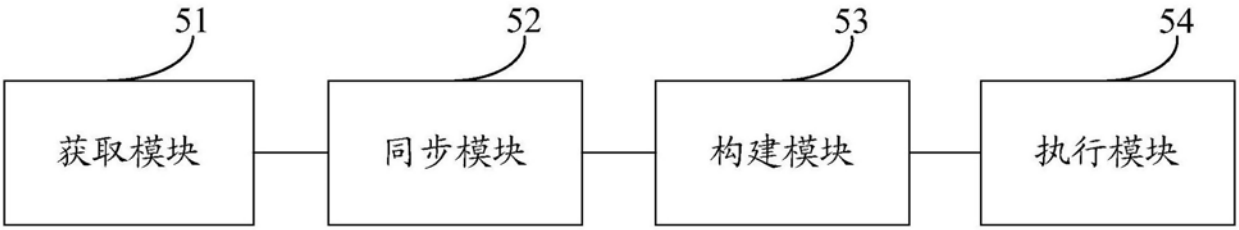

51
52
53
54
获取模块
同步模块
构建模块
执行模块

图5