06 | 计算输入的正确性：怎么选择正确时间的数据？

2021-01-04 任杰

分布式金融架构课 进入课程 >

讲述：任杰
时长 22:25 大小 20.53M

你好，我是任杰。今天起我们进入了第二个模块：系统正确性保障。

在前面第一个模块“金融业务及系统”里，我带你了解了常见的金融业务、盈利模式和对系统工具的要求。在第一个模块的最后，我们讲了**领域驱动设计**，它是一个在金融行业行之有效的方法论。

但是领域驱动设计只是从顶层设计来分析应该怎么做金融系统，并没有说在具体实践的时候怎样才能把系统做好。所以我们在这个模块会重点解决怎么做才能达到金融系统的最重要的要求：正确性。

所谓巧妇难为无米之炊，如果在处理金融业务的时候没有用到正确的金融数据，那计算出的结果是万万不能相信的。所以正确的数据是所有正确性的基础。那让我们来一起看看怎

么解决正确性的第一个问题：怎么选择正确时间的数据。

业务举例

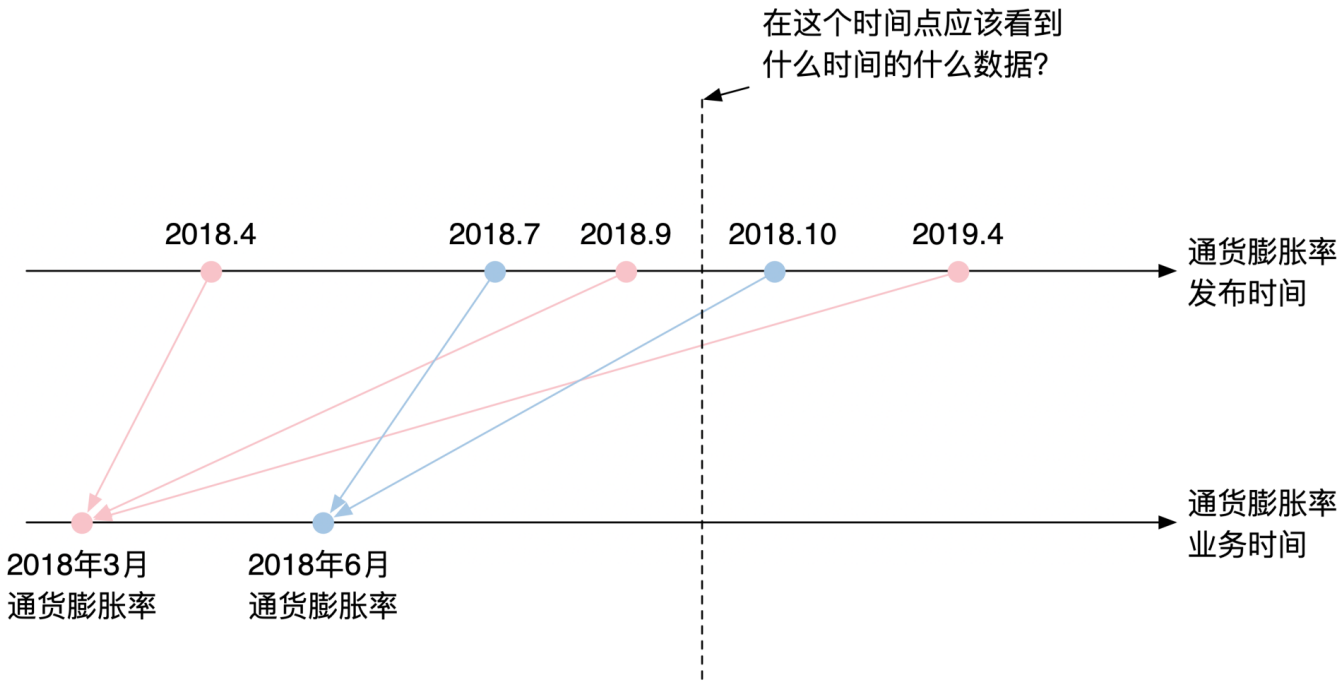
和前面一样，我们在分析技术之前先看一个金融业务的例子。

在国外有一种金融机构叫作养老基金，大家平时交的养老保险有时候就是养老基金在负责处理。由于养老基金的收益是在你退休之后才能获得，所以养老基金一个最重要的衡量指标就是，这个基金未来能不能给你足够的生活费用。

你应该能感觉到，现在生活费用越来越贵，同样的东西第二年就会涨价。那一个可能的衡量指标就是，养老基金每年的收益率能不能超过每年生活费用的涨幅。

生活费的涨幅一般用**通货膨胀率**（CPI，Consumer Price Index）来表示。通货膨胀率每个月都会公布，它就是一个数字而已，这是一个比较简单的金融市场数据。但是它的特点在于数据公布时间特别晚。当前只能公布一个现在通货膨胀率的预期值。真正的值可能要几个月之后才能公布，而且之后还有可能会修改。

比如下面这张图的例子里，我们公布了两份数据，分别是 2018 年 3 月和 6 月的通货膨胀率。2018 年 3 月的数据是一个月之后公布的，在 3 个月之后和 1 年之后又公布了两次修改。6 月份的数据也类似。



那问题来了。上面这个例子里公布了 5 次共 2 个数据，每次公布的数据都不一样。你怎么确保每次数据的更新都不会影响前面已经完成的金融业务呢？

这个问题看起来非常简单，但是如果我给你介绍一下大的背景，你就不会觉得简单了。金融公司会面临成千上万的金融数据，通货膨胀率只是其中的一种。而且，像保险这种金融业务可能一旦签署就需要执行好几十年，这几十年里公司的信息系统会发生翻天覆地的变化。

如果你成为了金融公司的 CTO。有一个人拿了 30 年前的养老保险，投诉你说通货膨胀率用错了。这时候监管人员来到你的办公室，要求你一步一步证明合同数据是正确的。合同的签订需要好几个月，也会涉及到很多部门。你怎么能保证这么长的时间内，所有部门都没有因为之后的数据更新而用错数据呢？

再假设你是这家金融公司的 CFO。你想分析一下 2018 年 3 月的通货膨胀率，看看每次调整对保险合同价格的影响是多少，这样你才能知道通货膨胀对整个公司的影响有多大，所以你想用更新后的数据来算老合同的价格。

那么问题来了，你怎样才能保证你只是用了更新后的 2018 年 3 月的数据，而不是用了更新后的其他月份的通货膨胀率数据呢？如果你想对公司层面所有合同做类似的计算，怎样才能保证所有人把所有数据都更新正确了呢？

所以，**金融公司的数据正确性是一个正确性的管理问题。**你需要让业务、运营、财务、合规等所有部门的信息系统都用统一的数据访问方式。这种数据访问方式还需要可重现。不管是过了多少年，系统更新换代了多少次，开发人员换了多少批，你都能正确地知道过去发生了什么。这也是我们这节课叫作“正确时间”的原因。

对一家小金融公司来说这些都不是问题。但是如果是一家大的金融机构，立志于流芳百世，那么这些就会是非常困难的架构挑战和管理挑战。

既然我们要找到一个能满足金融公司所有部门的长期的数据使用方案，那么这个方案一定要和金融数据的核心原理相关。在金融系统里这个解决方案叫作双时序数据库。

我需要提醒你一下，**双时序数据库和领域驱动设计一样，只适合解决复杂的问题。**所以它一般用来解决机构金融业务，而很少用来解决普惠金融业务。

那话不多说，让我们来看看它是怎么从原理上解决数据时间的问题吧。

如何理解双时序数据库？

双时序坐标轴

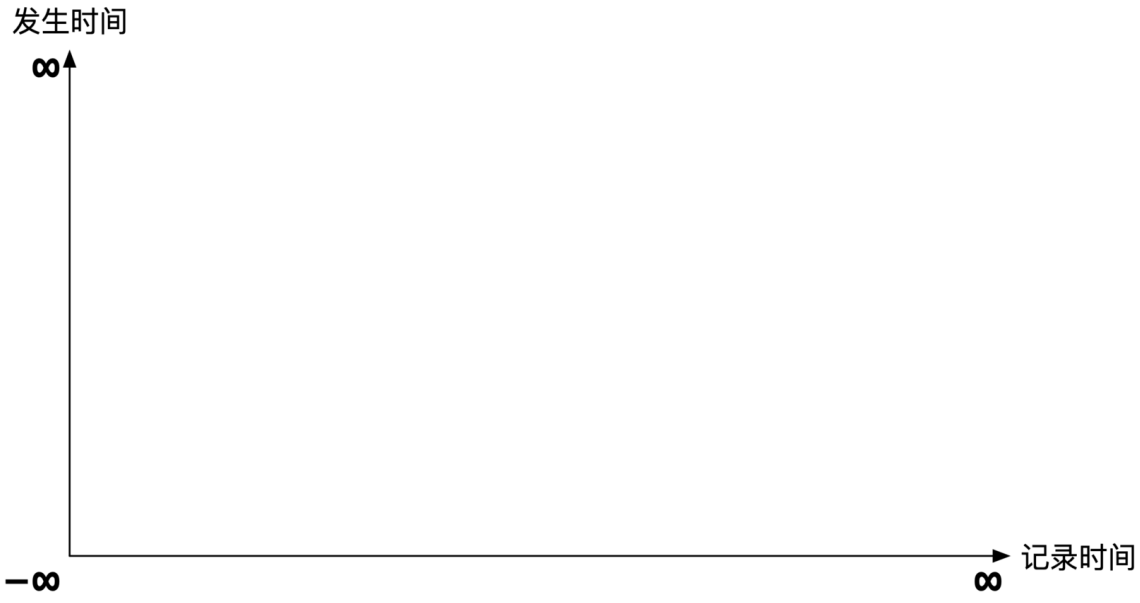
从前面的通货膨胀率的例子你可以看到，一个金融数据会有两个时间，一个是数据对应的业务发生时间，一个是数据的修改时间。在双时序数据库里这两个时间分别叫作**发生时间**和**记录时间**。发生时间也叫作 **valid time**，一般缩写成 **VT**。记录时间叫作 **transaction time**，一般缩写成 **TT**。

既然存在双时序数据库，那么一定还有单时序数据库。你平常见到的时序数据库其实就是单时序数据库。这两者的区别在于，**单时序数据库解决的是数据增加问题，双时序数据库解决的是数据修改问题。**

时序数据库作为双时序数据库的简化版本，在复杂度要求不高的场景有广泛的应用。时序数据库在金融行业和互联网行业都有很多相关介绍，你如果有兴趣可以查看相关文档。

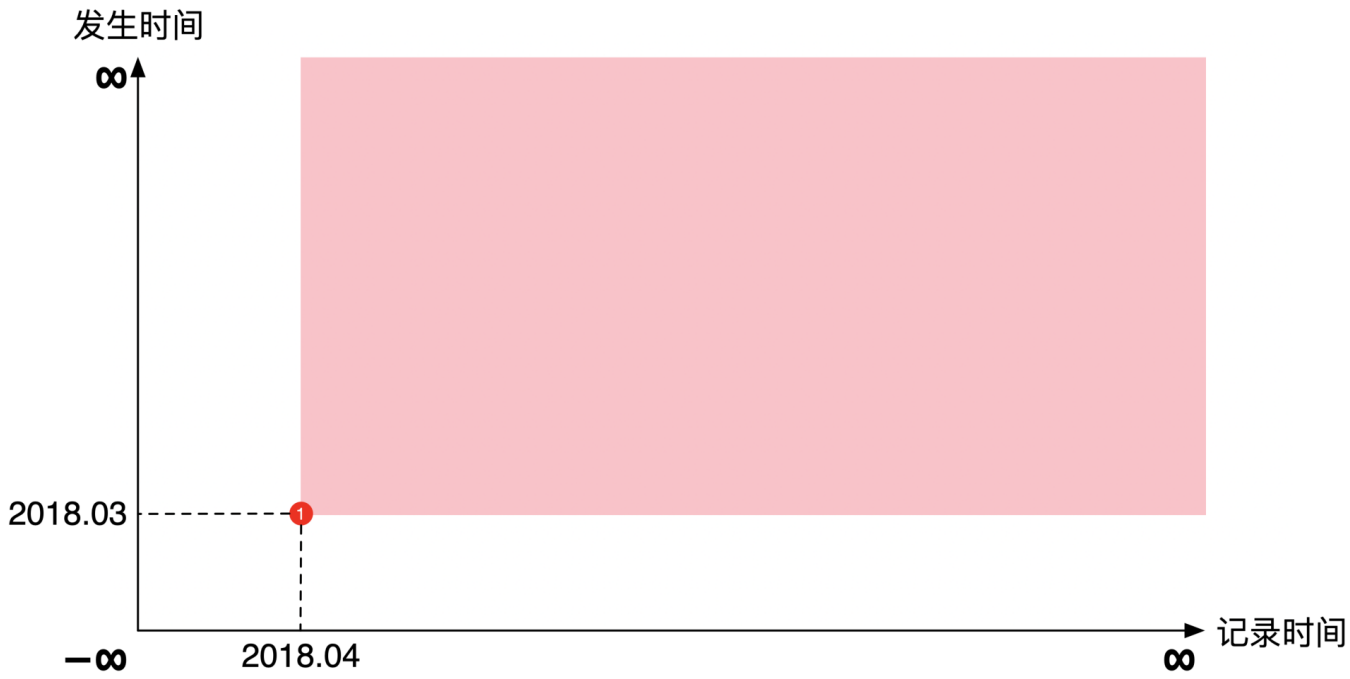
为了方便你理解核心概念，我把双时序数据库的时间逻辑画了出来。由于双时序数据库有两个时间，我们需要二维平面才能表示，所以我们需要画一个坐标系。

按照行业惯例，**横轴是记录时间，纵轴是发生时间。**理论上每个轴都指向正负无穷远，但是在实际展示中通常将坐标原点表示为负无穷远，主要是为了画起来好看。示意图如下：



数据可见范围

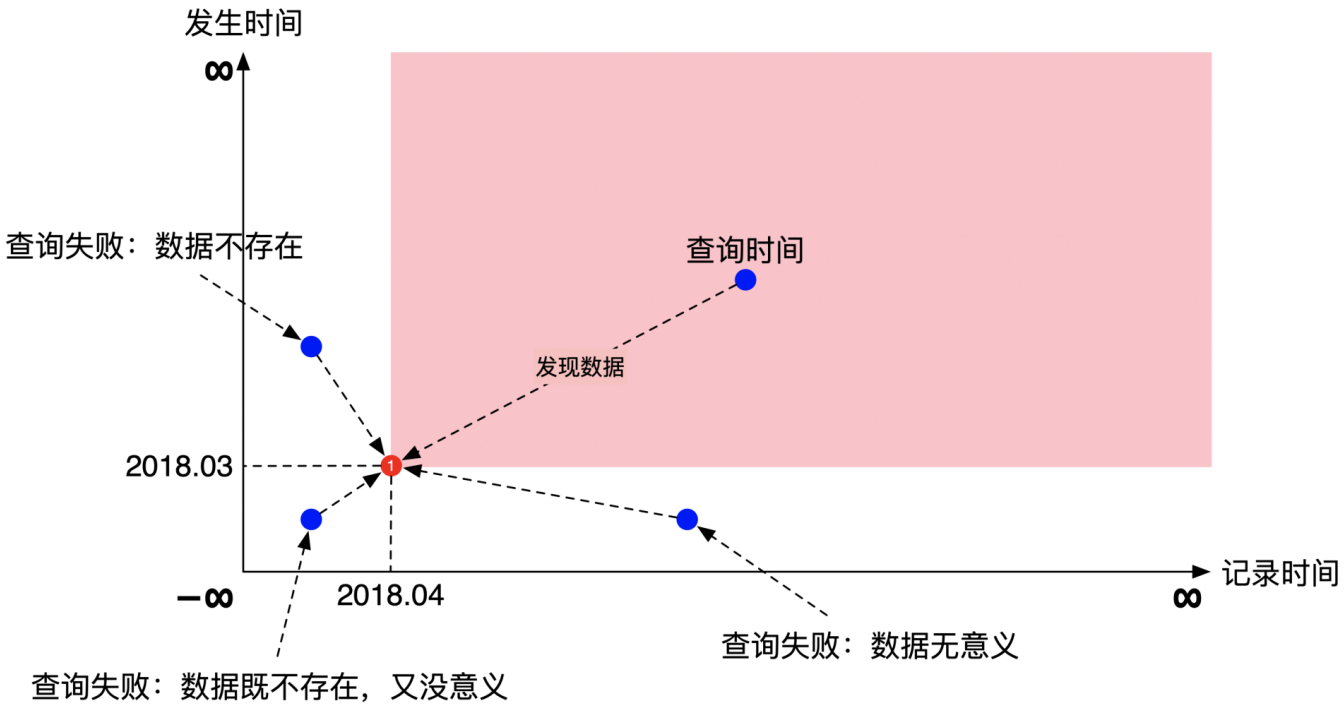
那根据前面的例子，我们在 2018 年 4 月收到了一个月前的通货膨胀率的数据。这个数据对应了坐标系的一个点，坐标为（2018 年 4 月，2018 年 3 月）。画出来就是下面这个样子：



你会发现，上面图中这个坐标系里有一个粉红色的方块。这个方块表示了数据的在系统内的**可见范围**。那什么叫可见范围呢？这就涉及到如何查询双时序数据库的数据了。

既然数据的存储有两个时间，那么**数据的查询也需要同时提供两个时间，也是发生时间和记录时间。先看记录时间的逻辑。通货膨胀率的发生时间是 2018 年 3 月，这意味着这个时间点数据没有任何意义。再看看记录时间。通货膨胀的记录时间是 2018 年 4 月，这意味着这个时间点前数据还不存在。所以可见范围指的是数据既存在而且有意义的时间范围。**

下面这幅图表示了 4 种查询范围。其中在粉红色方块的查询能查到数据，其余 3 个都查不到数据。你可以感受一下具体的查询过程：

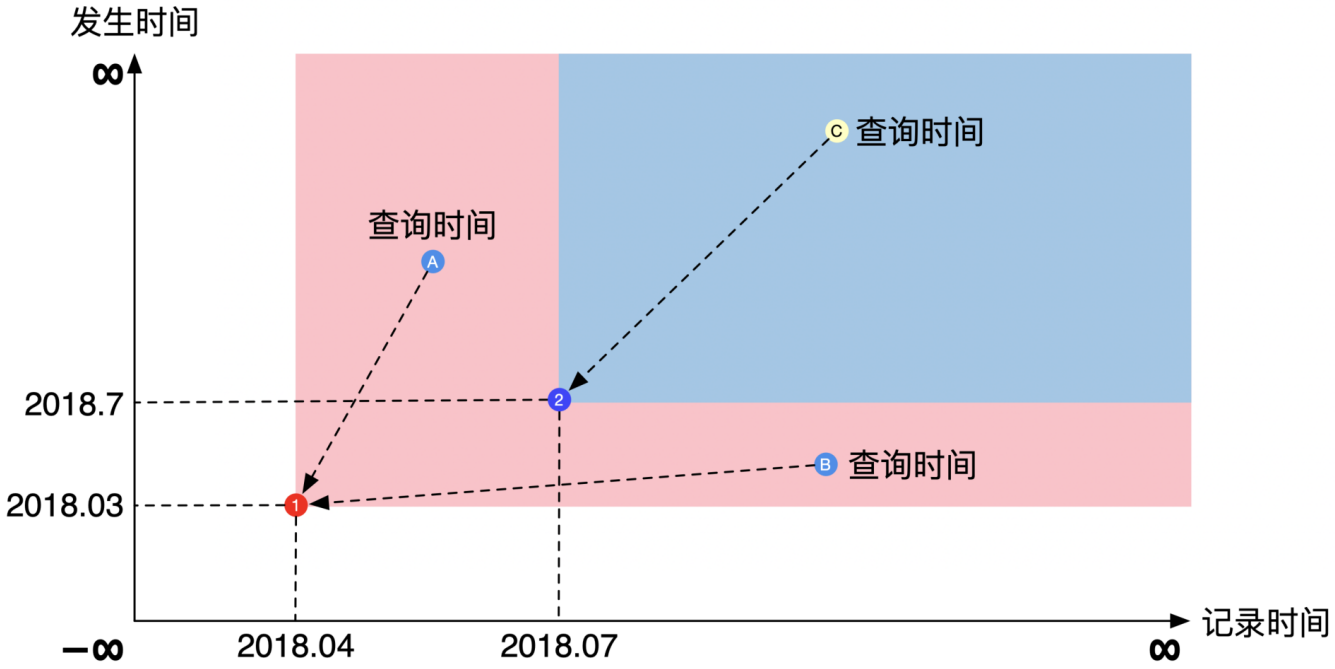


可见范围的覆盖

根据假设，在 3 个月之后的 2018 年 7 月，收到了一个月前的通货膨胀数据。这时候坐标系在（2018 年 7 月，2018 年 6 月）多了第二个点。我们将这两个点的可见范围画出来就是下面这幅图的样子：

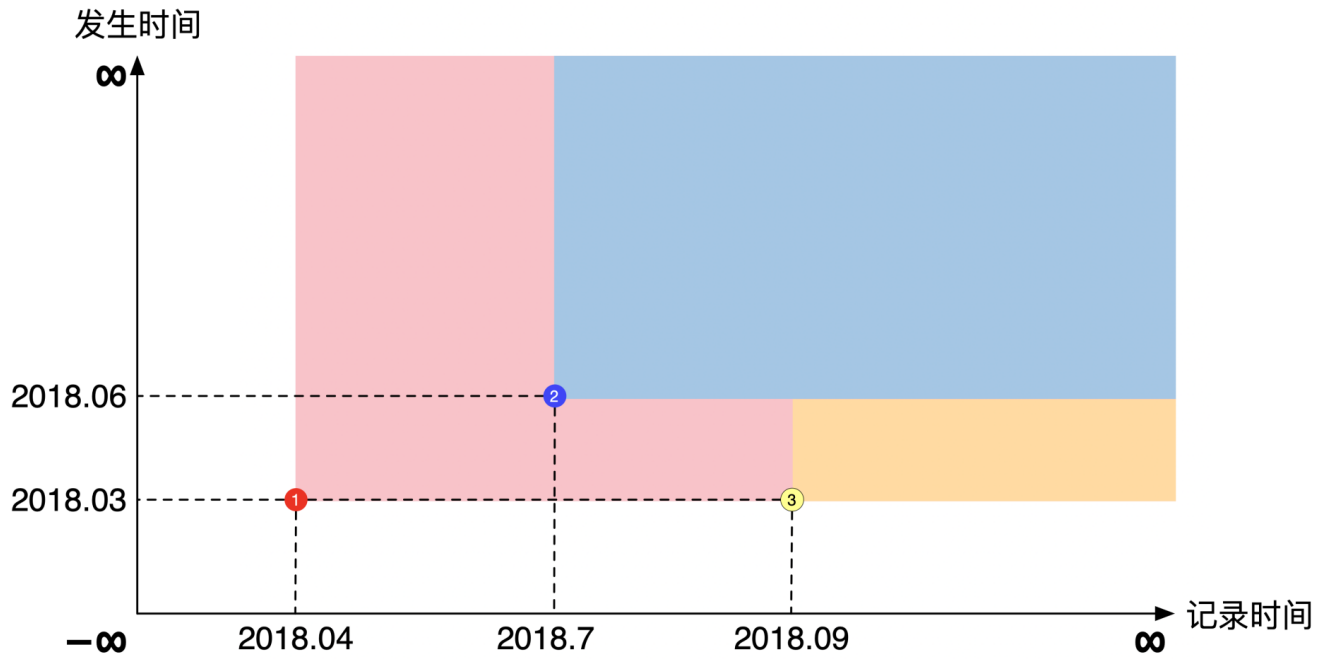
可以看出来，第二个点加入之后新增了一块蓝色的矩形区域，覆盖了原来矩形的右上角。还是按照之前对可见范围的定义，不同颜色的区域表示了你能看到的具体是哪个数据。

所以如果你查询用的两个时间点的坐标刚好在蓝色区域时，看到的就是 2018 年 7 月新增的通货膨胀率数据，如果你的点坐标在粉红色区域时，看到的就是 2018 年 4 月增加的数据，就像下面这幅图表示的一样：



可见范围的正确定义

假设又过去了 2 个月。在 2018 年 9 月的时候，机构更正了 2018 年 3 月的通货膨胀率，也就是更正了我们录入的第一个数据。这是我们坐标系的第三个点。你会发现这第三个数据和第一个数据的发生时间都是一样的，但是记录时间差了半年。下图展示了第三个数据加上去之后，各个数据的可见范围：



你会发现这第三个点的可见范围有些奇怪。第三个点的纵坐标碰到第二个点的时候就停下来了，而其它点的纵坐标都向上到无穷远。这个奇怪的现象要怎么理解呢？这就需要我们先理解一下可见范围的正确定义了。

可见范围是和数据查询息息相关的。我们是用查询的结果来定义这个结果的可见范围。当我们在做数据查询的时候，我们关心的是**离当前查询时间点最近的合理数据**。

这里需要一点解释。**定义里的“合理”指的是数据既存在，且有意义**，也就是说查询的记录时间和发生时间不能比数据的时间要早。**定义里的“最近”指的是当有多个数据都是合理的时候，选择发生时间最晚的数据。**

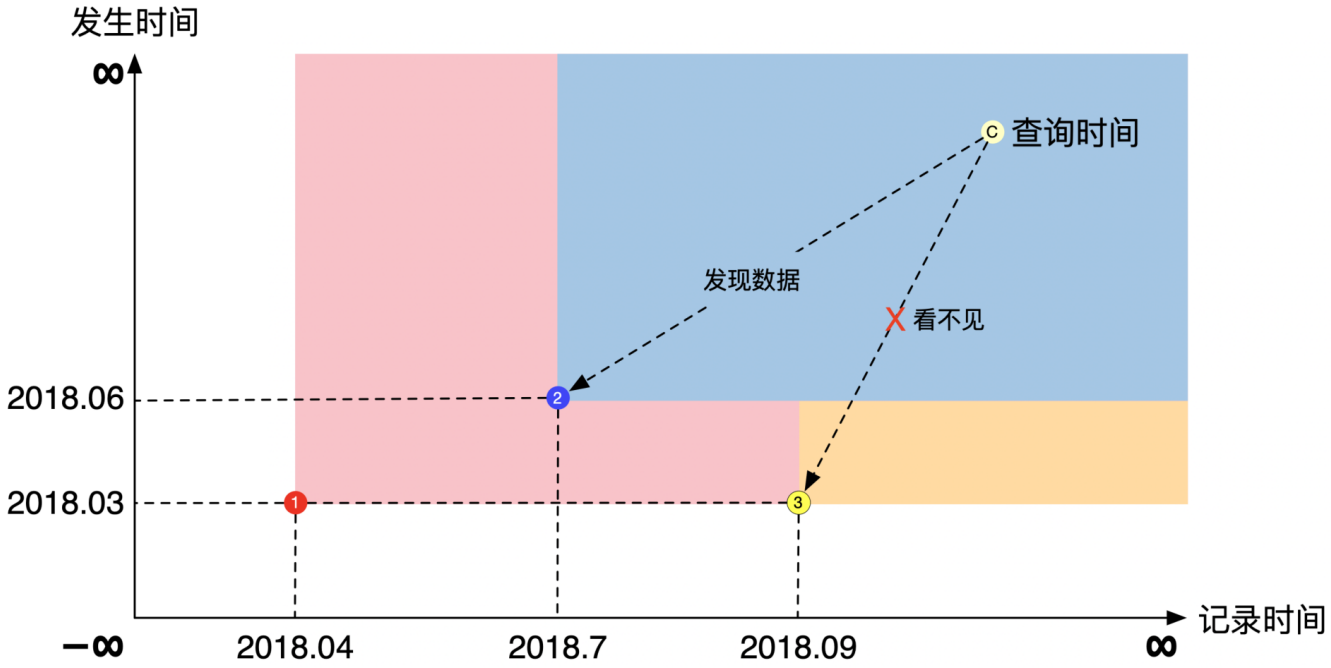
你要注意的是，可见范围定义里的“最近”和金融业务里的“最近”的定义是基本一致的。

比如在金融业务中我们经常会问到现在的股价是多少，或者现在的利率是多少。由于**金融数据的变化永远是离散的，而不是连续的，所以并不存在一个时间叫“现在”。**当你问现在是多少，其实从逻辑上来讲，你问的是离现在最近的数据是多少。

所以当你在双时序数据库查询的时候，你表达的意思是当你坐上“时间机器“返回到查询所对应的记录时间，然后查询在发生时间点以前就已经生效的所有数据之中，哪个数据离你最近。

所以，**正确的数据可见范围定义是能查询到这个数据的查询时间点的范围。数据的可见范围和查询是互相定义的，你需要仔细思考。**

当我们解释完最终版的可见范围之后，你就能理解为什么下面这幅图查询到的是第 2 个数据，而不是第 3 个数据，而且为什么第 3 个数据的可见范围只有一小部分。



优缺点分析

说到这里，你应该已经了解双时序数据库的基本原理和使用方法了。在我们实际应用之前，还需要知道它的优缺点，这样你才能设计之初就会有个合理判断。

优点

双时序数据库最大的优点是**数据的不变性**。没有特殊要求的情况下，金融行业要求数据不可被覆盖和篡改，这种业务需求决定了系统数据一定要具有不变性。

另一个优点是**数据的唯一性**。所有数据都有唯一标识符，也就是数据对应的记录时间和发生时间。所有数据的可见范围也可以由这个数据的唯一标识符来唯一决定。

如果数据有唯一标识符，而且数据永远不变，那么数据的使用就有了正确性保证。这里逻辑环环相扣，你一定要跟上。数据的使用由数据查询开始。数据查询对应的坐标点属于某一个可见范围之内，而这个可见范围有对应数据的唯一标识符。所以我们就可以从一个确定的查询时间定位到确定的数据时间。

那回到我们开头提的第一个问题，你怎么才能知道 30 年前养老保险涉及到的所有数据？当你用合同定制时间作为记录时间和发生时间，就能查询到 30 年前这个合同用到的所有数据。之后的修改一定不会影响你查询的结果。

最开始提到的第二个问题也有了答案。如果你想知道每次对 2018 年 3 月的通货膨胀率的修改究竟会带来什么影响，就需要**保持 2018 年 3 月这个发生时间不变，然后依次调整记录时间。**这样就能保证你只更新了 2018 年 3 月的数据，而没有意外地用到其他月份的数据。

需要指出的是，**通过调整记录时间来选择性地引入数据变化**的方法在金融行业有很广泛的应用。金融行业在进行风险分析的时候会采用**情景计算**（Scenario Analysis）的方式进行。监管机构会提出一些假设性的事件（What if），比如银行挤兑、地震、贸易战等等。为了完成计算，需要对金融合同数据进行修改。

在引入双时序数据库之前，我们需要花费很大的人力物力来保证情景计算的修改不会影响到真实数据的使用。在引入双时序数据库之后，由于每次修改只会影响到记录时间，我们只需要使用合同中记录的原始的业务时间，就能保证所有的业务数据不会受到情景计算的影响。

数据的时间正确性是所有金融计算正确性的开始。我们会在下一节课学习事件溯源的架构设计，这个架构能保证计算过程的正确性。一旦这个架构的数据输入是正确的，那么整个架构就能真正达到金融级别的正确性。

缺点

优点说完了，我们再看看缺点，缺点有两个。

双时序数据库的第一个缺点是**学习成本高。**以往处理数据的时候都只有一个时间，现在变成了两个时间。所有开发人员都需要了解二维情况下的数据可见范围。有时候我们跟业务方和产品经理沟通，也会发现他们也需要用双时序数据来定义自己的数据使用规则。这些都是很高的教育成本。

双时序数据库的另一个缺点是**执行速度慢。**和时序数据库相比，双时序数据库多了一个维度的时间，所以需要多加一个索引。这个额外的索引在数据插入和查询时候都会消耗额外的时间，因此不太适合于延时要求非常高的使用场景。

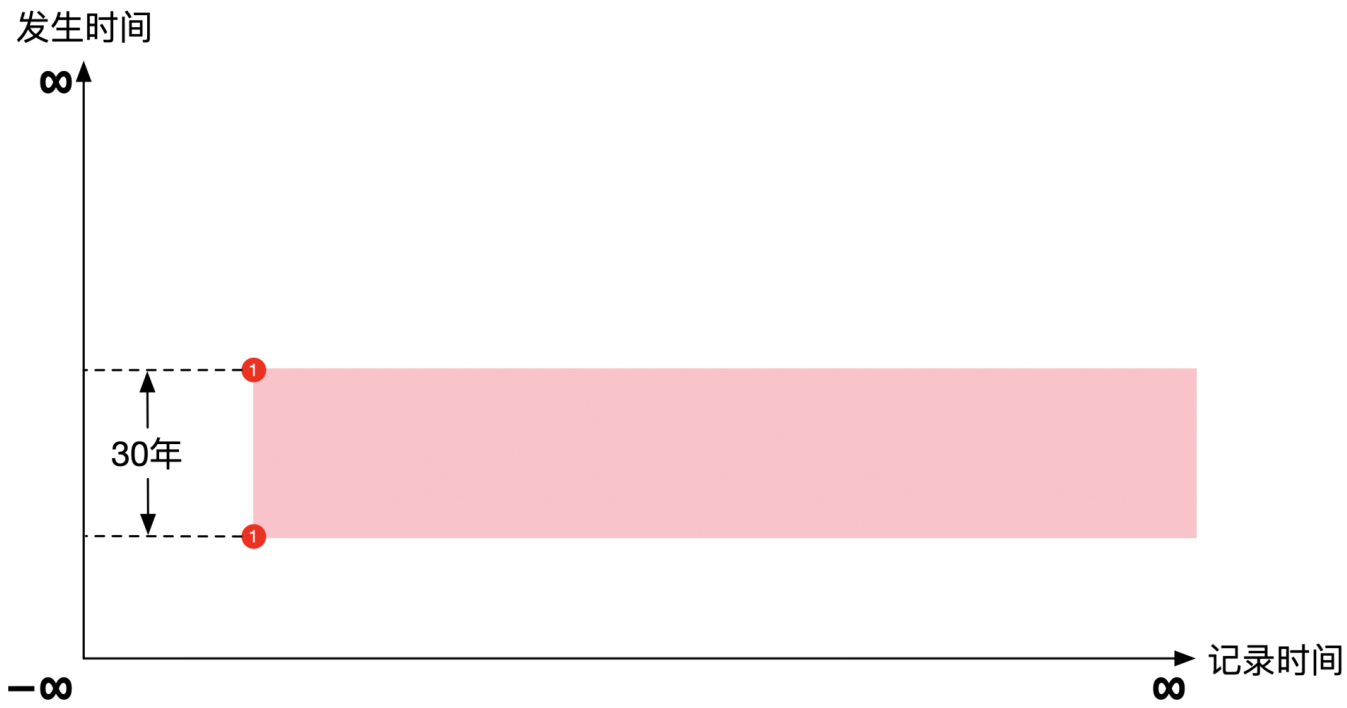
你还记得第 4 节课里，我们说过**金融讲的是投资回报比，而不是只单纯考虑成本。**虽然双时序数据库的学习成本和使用成本都不低，但是作为整个公司层面的数据正确性框架来

说，它能让所有人深入理解数据的时间本质，从框架层面排除了不正确的使用方式，从而降低出错的可能性。从长期来看，有十年磨一剑的功效。

理论与实际的区别

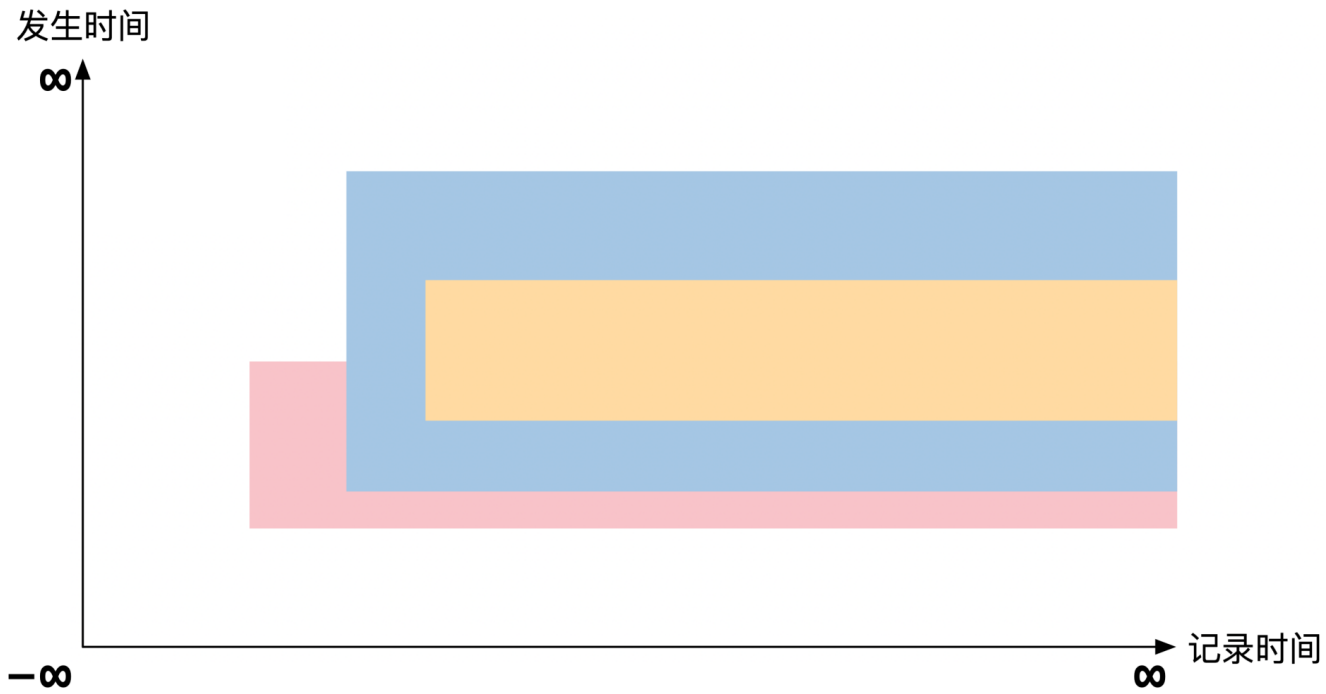
我们在最开始介绍双时序数据库的可见范围时，没有说过发生时间的可见范围有多大。所以可见范围默认是一直可见的。但是理论上并没有这个假设。**理论上数据的可见范围可以是有限的。**

拿房贷举个例子。房贷最长时间是 30 年，所以 30 年以后房贷合同就无效了，也就是房贷合同的可见范围只有 30 年。这意味着在双时序数据库里，你的房贷合同的可见范围是一个高度为 30 年的矩形，看起来应该是下图这个样子：



虽然有发生时间限制的房贷合同看起来非常合理，但是在实际处理过程中却碰到了操作复杂度上的问题。

比如你几年后和贷款公司商量要延长房贷合同的期限，将期限延长到 100 年。但是又过了几年，贷款公司觉得 100 年太长，又调整成 50 年。从理论上来讲，这时候会有 3 个高度有限的可见范围互相覆盖。如果用带有可见范围约束的双时序数据库来表示，结果就是下面这张图：



虽然逻辑上是正确的，但是在实际使用时人们发现可见范围的定义会变得过于复杂，同时在数据库实现上也会碰到很多查询优化问题，所以实际一般不推荐对发生时间做可见范围的约束。如果数据真的失效了，你可以通过保存一个新的无效版本来覆盖原来的可见范围。

小结

我们这节课学习了如何用双时序数据库来正确存储和查询金融数据。

因为金融数据大多都与时间相关，用时序数据库可以很好地解决一些金融数据的使用场景，但是无法很好地处理数据的修改问题。这样一来，我们就需要新的解决方案，也就是双时序数据库。

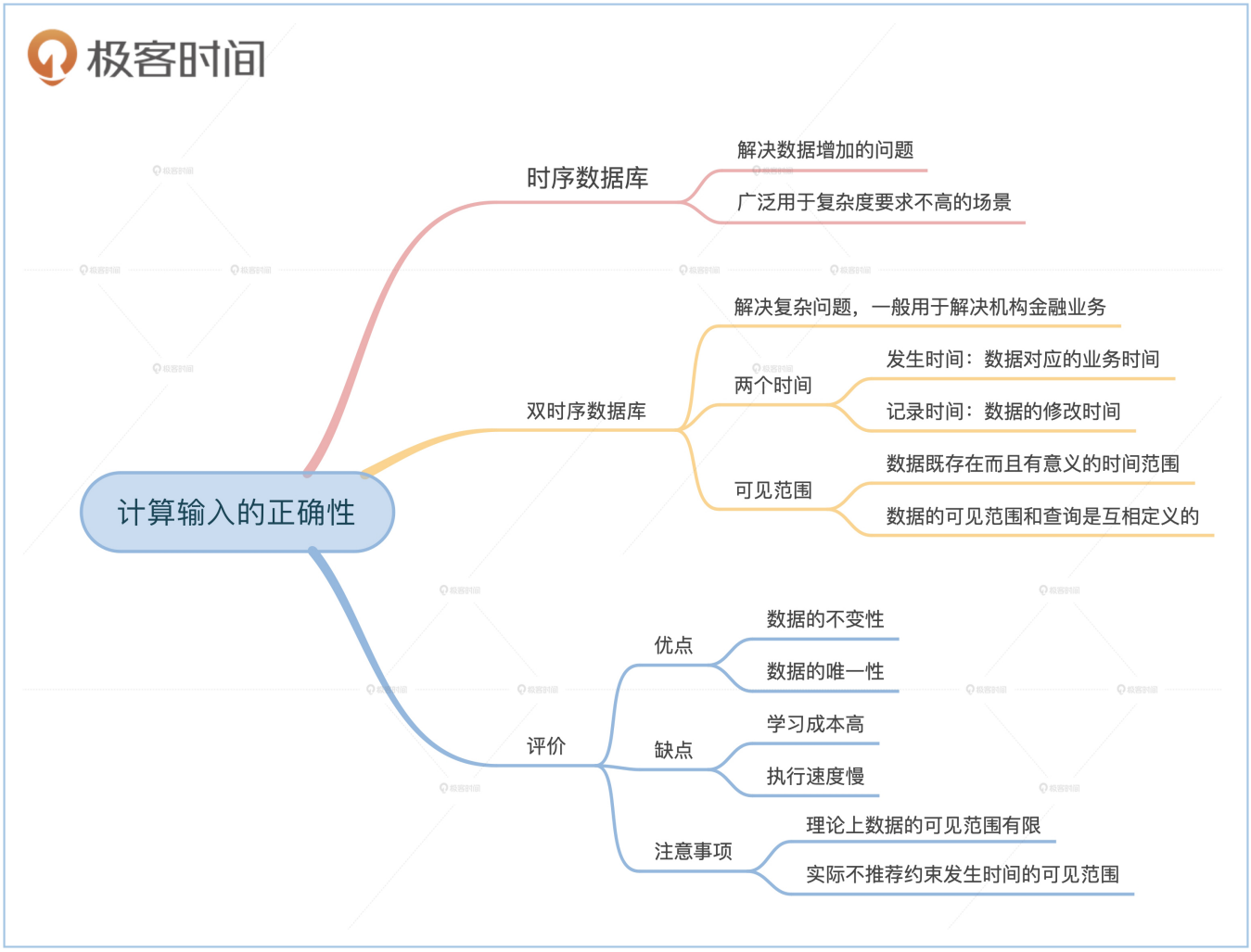
双时序数据库除了存储数据的发生时间外，还保存了系统的记录时间，所以对于每个数据都有两个相关时间，组成了一个坐标系。**双时序数据库的数据插入和查询操作都可以理解为坐标系节点之间的可见范围的处理。**

由于多了一个维度的时间，双时序数据库有了额外的优点和缺点。优点是数据的唯一性和不变性得到了保证。缺点是系统的学习成本和使用成本偏高。

理论上双时序数据库里的数据发生时间范围并不一定是无限的，而是可以有一定区间范围。但是在实践过程中会导致额外的使用复杂度，所以并不建议采用。

下一节课我们会学习事件溯源这个保证计算正确性的框架。双时序数据作为框架的数据输入，是整个流程正确性的保证。

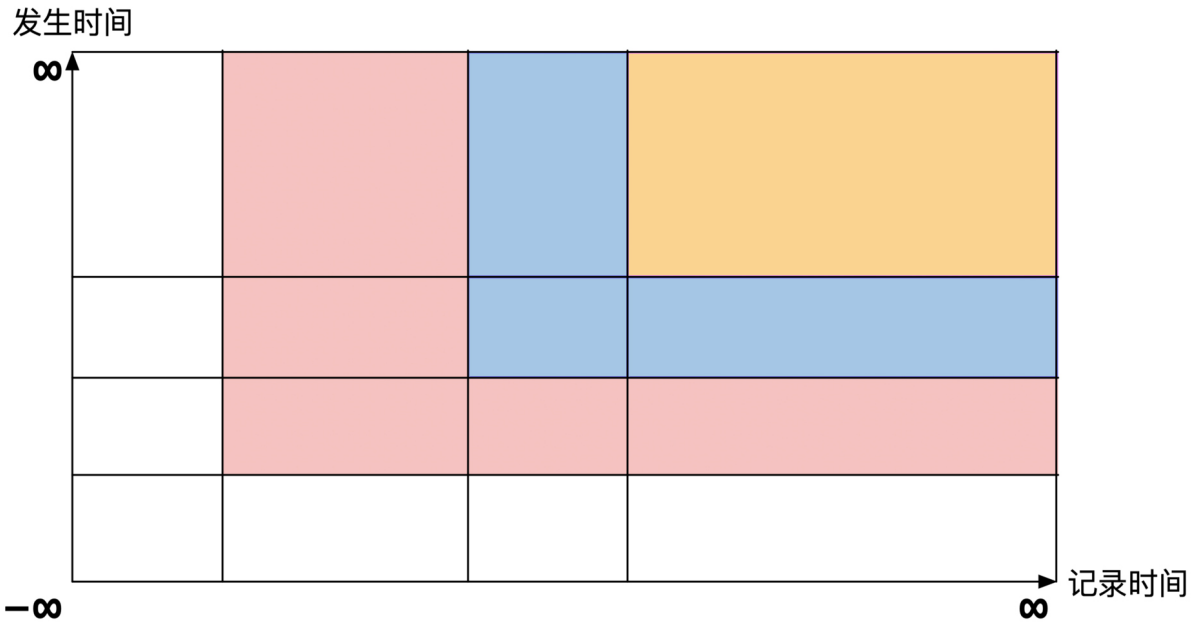
双时序数据库尽管看起来有些复杂，但是它是一个金融级的数据正确性解决方案，金融公司的规模越大，历史越悠久，就越能显示出这种方法的威力。这就是高盛和摩根士丹利这些华尔街的大型投资银行的核心竞争力。



思考题

双时序数据库里的一种存储方式是将坐标空间切割成尽量多的矩形，然后将这些矩形存储在数据系统内。数据库的索引建立在矩形的左下角和右上角这两个坐标点。

具体的切割做法是当坐标系内新增一个数据节点时，以这个点为中心，将整个坐标系进行水平和垂直切分。下图展示了系统中有 3 个数据点时的一个切割方式，3 个数据点将坐标系切割成了 16 个矩形：



每个插入操作都会对已有的矩形进行切割。每次查询都会遍历相关的矩形。那么你能算一算这个方案的**存储空间复杂度**和**查询时间复杂度吗**？

欢迎你在留言区分享你的感悟和疑问。如果有所收获，也欢迎你把这篇文章转发给自己的朋友、同事，一起交流学习。

提建议

上一篇 答疑集锦（一）| 思考题解析与外汇架构知识拓展

下一篇 07 | 计算过程的正确性：如何设计正确的数据处理架构？

精选留言 (4)

写留言

Geek_9c3134

2021-01-04

老师 支付密码用什么方式加密保存比较好

展开

作者回复: 这位同学你好。支付密码的存储方式和一般密码的方式一样，在这里给你简单说一下原则。

最重要的是密码一定不能明文存储，也不能明文传输。所以在一般会存储密码的哈希值。

有一个叫做“彩虹表”的工具可以很快速的破解强度不高的密码。为了对抗彩虹表，你需要在哈希值里加一些其它的数据，一般叫做“加盐”。“加盐”有很多种选择，比如电话、随机数等等，要点是每个用户的“盐”要不一样，这样就能增加破解的难度。另外，“盐”和密码哈希值不能放在一起，要放在不同的表，甚至不同的地方。

另外，哈希算法的选择也会有一些考虑。密码的攻防其实是你和对方在比拼资源的消耗。所以你可以选择一些计算特别耗时的哈希算法，这样就能消耗破解方的计算资源。

总之，密码的处理是不同行业的通用问题，你可以看一下互联网行业关于这方面的详细介绍，基本大同小异。

3

webmin

2021-01-16

查询时间复杂度：时间是线性增长的，本身就是有序的，二分查找左下角坐标，即第一个小于等于VT和TT的坐标，再查找右上角坐标最后一个大于等于VT和TT的坐标，再在这个矩形中查询相关指标数据，算下来(logN * logN) + logN = N + logN
存储空间复杂度：与时间点数有关系，每加一个时间点之前的数据都要指数倍增涨，所以存储空间复杂度是N^2

展开

Geek_c51819

2021-01-10

双时序数据库到底在数据库里面怎么设计呢？发生时间一对多修改时间么？

tt

2021-01-04

三个点把空间分成的16份，那空间复杂度应该是O(N^2)，索引建立在左下角和右上角两

个坐标点上，那么每次查询的时候根据查询时间点可以直接定位到一个矩形内，所以时间复杂度是O(1)。

也是和所有索引一样，用空间换时间。...

展开

编辑回复: 感谢tt同学踊跃留言，给你的学习热情点赞！

双时序数据库目前还没有对外开源的例子，我们知道的是它比较适合交易量稍小的场外市场业务，一般是金融公司自研。