文章
姚 鑫 · 六月 17, 2022 阅读大约需 7 分钟

第三章 锁定和并发控制（三）

第三章 锁定和并发控制（三）

升级锁

使用升级锁来管理大量锁。当锁定数组的节点时，它们是相关的，特别是当将多个节点锁定在同一下标级别时。

当给定进程在同一数组中的给定下标级别创建了超过特定数量（默认为 1000）的升级锁时， 将删除所有单独的锁名称并用新锁替换它们。新锁位于父级，这意味着数组的整个分支被隐式锁定。示例（如下所示）演示了这一点。

应用程序应在合适的情况下尽快释放特定子节点的锁（与非升级锁完全相同）。当释放锁时，
会减少相应的锁计数。当的应用程序移除足够多的锁时，会移除父节点上的锁。第二小节显示了一个示例。

锁升级示例

假设有 1000 个^MyGlobal("sales","EU",salesdate) 形式的锁，其中 salesdate 表示日期。锁表可能如下所示：

注意 Owner 19776 的条目（这是拥有锁的进程）。 ModeCount 列指示这些是共享的、升级的锁。

当同一进程试图创建另一个相同形式的锁时， 会升级它们。它会移除这些锁并用名称为 ^MyGlobal("sales","EU") 的单个锁替换它们。现在锁表可能如下所示：

ModeCount 列表明这是一个共享的升级锁，它的计数是 1001。

请注意以下关键点：

- ^MyGlobal("sales","EU") 的所有子节点现在都被隐式锁定，遵循数组锁定的基本规则。
- 锁定表不再包含有关 ^MyGlobal("sales","EU")
  的哪些子节点被特别锁定的信息。这在删除锁时具有重要意义。见下一小节。

当同一进程添加更多形式为 ^MyGlobal("sales","EU",salesdate) 的锁名称时，锁表会增加锁名称
^MyGlobal("sales","EU") 的锁计数。锁定表可能如下所示：

ModeCount 列指示此锁的锁计数现在为 1026。

移除升级锁

与非升级锁完全相同，应用程序应尽快释放特定子节点的锁。当这样做时，
会减少升级锁的锁计数。例如，假设代码删除了 ^MyGlobal("sales","EU",salesdate) 的锁定，其中 salesdate 对应于 2011 年的任何日期 — 因此删除了 365 个锁定。锁表现在看起来像这样：

请注意，即使现在锁的数量低于阈值 (1000)，锁表也不包含 ^MyGlobal("sales","EU",salesdate). 的锁的单独条目。

节点 ^MyGlobal("sales")保持显式锁定，直到该过程再删除 661 个 ^MyGlobal("sales","EU",salesdate) 形式的锁定。

重要提示：有一点需要考虑，与前面的讨论有关。应用程序可能会 “ 释放 ” 数组节点上的锁，这些节点一开始就从未锁定，从而导致升级锁的锁计数不准确 - 并且可能在需要这样做之前释放升级锁。

例如，假设进程锁定 ^MyGlobal("sales","EU",salesdate) 中从 2010 年到现在的节点。这将创建超过 1000 个锁，并且此锁将按计划升级。假设应用程序中的错误删除了 1970 年节点的锁。
将允许此操作，即使这些节点以前没有被锁定，并且 会将锁计数减少 365。生成的锁计数不会是所需锁的准确计数。如果应用程序随后移除了其他年份的锁，则升级的锁可能会意外地提前移除。

Locks, Globals, and Namespaces

锁通常用于控制对全局变量的访问。因为可以从多个命名空间访问全局， 为其锁定机制提供自动跨命名空间支持。该行为是自动的，不需要干预，但在此描述以供参考。有几种情况需要考虑：

- 任何命名空间都有一个默认数据库，其中包含持久类和任何其他全局变量的数据；这是此命名空间的全局数据库。访问数据时， IRIS 会从该数据库中检索数据，除非有其他考虑。一个给定的数据库可以是多个命名空间的全局数据库。请参见方案 1。
- 命名空间可以包括提供对存储在其他数据库中的全局变量的访问的映射。请参见方案 2。
- 命名空间可以包括下标级别的全局映射，这些映射提供对部分存储在其他数据库中的全局变量的访问。请参见方案 3。
- 在一个命名空间中运行的代码可以使用扩展引用来访问在此命名空间中不可用的全局变量。请参见方案 4。

尽管锁名称本质上是任意的，但是当使用以插入符号 (^) 开头的锁名称时，IRIS
提供了适合这些情况的特殊行为。以下小节给出了详细信息。为简单起见，只讨论排他锁；共享锁的逻辑类似。

场景 1：具有相同Global数据库的多个命名空间

如前所述，虽然进程 A 拥有一个具有给定锁名的独占锁，但没有其他进程可以获取任何具有相同锁名的锁。

如果锁名称以插入符号开头，则此规则适用于使用相同全局数据库的所有命名空间。

例如，假设命名空间 ALPHA 和 BETA 都配置为使用数据库 GAMMA 作为其全局数据库。下面显示一个草图：

然后考虑以下场景：

1. 在命名空间 ALPHA 中，进程 A 获得一个名为 ^MyGlobal(15) 的独占锁。
2. 在命名空间 BETA 中，进程 B 尝试获取名称为 ^MyGlobal(15) 的锁。此 LOCK 命令不返回；进程被阻塞，直到进程 A 释放锁。

在这种情况下，锁表只包含进程 A
拥有的锁的条目。如果检查锁表，会注意到它指示了该锁应用到的数据库；请参阅目录列。例如：

场景 2：命名空间使用映射的Global

如果一个或多个命名空间包含全局映射，系统会自动跨适用的命名空间强制实施锁定机制。当在非默认命名空间中获得锁时， IRIS 会自动创建额外的锁表条目。

例如，假设命名空间 ALPHA 配置为使用数据库 ALPHADB 作为其全局数据库。假设命名空间 BETA 配置为使用不同的数据库 (BETADB) 作为其全局数据库。命名空间 BETA 还包括一个全局映射，它指定 ^MyGlobal 存储在 ALPHADB 数据库中。下面显示一个草图：

然后考虑以下场景：

1. 在命名空间 ALPHA 中，进程 A 获得一个名为 ^MyGlobal(15) 的独占锁。

与前面的场景一样，锁表仅包含进程 A 拥有的锁的条目。此锁适用于 ALPHADB 数据库：

2. 在命名空间 BETA 中，进程 B 尝试获取名称为 ^MyGlobal(15) 的锁。此 LOCK 命令不返回；进程被阻塞，直到进程 A 释放锁。

场景 3：命名空间使用映射的Global下标

如果一个或多个命名空间包含使用下标级别映射的全局映射，系统会自动跨适用的命名空间强制实施锁定机制。在这种情况下，当在非默认命名空间中获取锁时，IRIS 还会自动创建额外的锁表条目。

例如，假设命名空间 ALPHA 配置为使用数据库 ALPHADB 作为其全局数据库。命名空间 BETA 使用 BETADB 数据库作为其全局数据库。

还假设命名空间 BETA 还包括一个下标级别的全局映射，因此 ^MyGlobal(15) 存储在 ALPHADB 数据库中（而这个全局的其余部分存储在命名空间的默认位置）。下面显示一个草图：

然后考虑以下场景：

1. 在命名空间 ALPHA 中，进程 A 获得一个名为 ^MyGlobal(15) 的独占锁。
2. 与前面的场景一样，锁表仅包含进程 A 拥有的锁的条目。此锁适用于 ALPHADB 数据库（例如，c:\InterSystems\IRIS\mgr\alphadb）。

当非默认命名空间获得锁时，整体行为是相同的，但 IRIS 处理细节略有不同。假设在命名空间 BETA 中，一个进程获得了一个名为 ^MyGlobal(15) 的锁。在这种情况下，锁表包含两个条目，一个用于 ALPHADB 数据库，一个用于 BETADB 数据库。这两个锁都归命名空间 BETA 中的进程所有。

当此进程释放锁名称 ^MyGlobal(15) 时，系统会自动删除两个锁。

场景 4：扩展的Global引用

在一个命名空间中运行的代码可以使用扩展引用来访问在此命名空间中不可用的全局变量。在这种情况下，IRIS 将一个条目添加到影响相关数据库的锁表中。锁归创建它的进程所有。例如，考虑以下场景。为简单起见，此方案中没有全局映射。

1. 进程 A 在 ALPHA 命名空间中运行，该进程使用以下命令获取 BETA 命名空间中可用的全局锁：

```
lock ^["beta"]MyGlobal(15)
```

2. 现在锁定表包括以下条目：

请注意，这仅显示全局名称（而不是用于访问它的引用）。此外，在这种情况下，BETADB 是 BETA 命名空间的默认数据库。

3. 在命名空间 BETA 中，进程 B 尝试获取名称为 ^MyGlobal(15) 的锁。此 LOCK 命令不返回；进程被阻塞，直到进程 A 释放锁。

进程私有Global在技术上是一种扩展引用，但 IRIS 不支持使用进程私有全局名称作为锁名称；无论如何，都不需要这样的锁，因为根据定义，只有一个进程可以访问这样的全局。

#SQL #Caché

源URL: https://cn.community.intersystems.com/post/%E7%AC%AC%E4%B8%89%E7%AB%A0-%E9%94%81%E5%AE%9A%E5%92%8C%E5%B9%B6%E5%8F%91%E6%8E%A7%E5%88%B6%EF%BC%88%E4%B8%89%EF%BC%89