基于 ArcSDE 的空间数据版本管理问题研究

夏 宇[1],朱欣焰[2],吴 维[2]
XIA Yu[1],ZHU Xin-yan[2],GUO Wei[2]

1.武汉大学 遥感信息工程学院,武汉 430079
2.武汉大学 测绘遥感信息工程国家重点实验室,武汉 430079
1.School of Remote Sensing and Information Engineering,Wuhan University,Wuhan 430079,China
2.State Key Laboratory of Information Engineering in Surveying,Mapping and Remote Sensing,Wuhan University,Wuhan 430079,China
E-mail:geoxy@126.com

XIA Yu,ZHU Xin-yan,GUO Wei.Research on spatial data version management based on ArcSDE.Computer Engineering and Applications,2007,43(14):14-16.

Abstract: This paper firstly discusses the advantages of version management based on ArcSDE under multi-user edit,and proposes three concepts of "conceptional layer","middle layer" and "physical layer",and then points out the version,state and table are the kernal concepts of the three layers respectively.Then the authors analyze the principle of version management from the three levels deeply.A new method of spatial data access for the spatial data after muti-user edit given,of which idea and implement are also discussed.The results of experiments demonstrate that it is feasible and practical.
Key words: ArcSDE;spatial data;version management;GIS

摘 要:首先指出多用户编辑下 ArcSDE 版本管理的优势,提出概念层、中间层和物理层的概念,指出版本、状态和表分别是这三个层次上的核心概念,并分别从各个层次上深入分析 ArcSDE 版本管理机制的实现原理。然后提出了一种新的 ArcSDE 版本管理中多用户编辑后空间数据访问方法,阐述了其思想和具体实现算法,最后以实验证实了其可行性和实用性。
关键词:ArcSDE;空间数据;版本管理;地理信息系统
文章编号:1002-8331(2007)14-0014-03 **文献标识码**:A **中图分类号**:P208

1 引言

通常 DBMS 采用"锁定-修改-释放"策略来实现对多用户并发操作数据库的控制,而地理空间数据的编辑工作常被称为"长事务处理"[1]。对地理空间数据的并发控制,如果仍然使用这种加锁策略,将会严重影响事务的并发程度,不能满足实际应用需要。对地理空间数据并发操作控制许多学者做过深入研究,朱欣焰[2]等较早就提出"通知-重读"法一定程度上提高了空间数据操作的并发程度,但实际上仍采用加锁机制,没有从根本上解决对空间数据的并发控制,而且客户端只被动接受更新的数据,缺乏灵活性,适合于多用户环境下要求在较短时间内完成的较为简单的编辑工作;程昌秀[3]等依然以多用户环境下较为简单的编辑为研究对象,进一步提出面向拓扑空间视图的扩展锁技术,提高多用户环境下空间数据库系统的效率;Gjermund Hanssen[4]总结分析了分布式地理空间数据库并发控制面临的挑战和目前的方法;Oracle WorkSpace Manager 在数据库的层次上采用版本管理对空间数据并发操作提供了支持,但不能满足大部分 GIS 用户对空间数据可视化操作的要求;ArcSDE 采用版本管理机制对长事务处理提供了底层的支持,结合 ArcMap 等可视化编辑工具,为多用户编辑并发控制提供了解决方案,其优势主要体现在:(1)用户各自在自己的版本里工作,无需对多个用户同时访问的数据对象进行锁定,提高了对空间数据操作的并发程度;(2)版本只是数据的"逻辑拷贝",没有空间数据的复制;(3)方便地在各个历史版本之间进行历史回溯。因此,研究 ArcSDE 版本管理机制以及探讨多用户编辑后空间数据访问方法具有一定的实用价值。

2 ArcSDE 版本管理机制

ArcSDE 版本管理机制可以从概念层、中间层和物理层三个层次上来认识,对应于版本、状态和表三个不同层次上版本管理机制的核心概念,层次结构如图 1 所示。

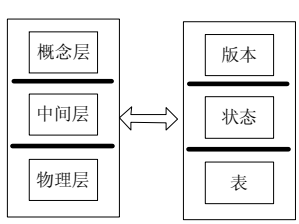


图 1 ArcSDE 版本管理机制层次结构

基金项目:国家重点基础研究发展规划(973)(the National Grand Fundamental Research 973 Program of China under Grant No.2006CB701305)。
作者简介:夏宇(1981-),男,博士生,主要从事空间数据库、遥感影像处理等方面研究;朱欣焰,男,教授,主要从事空间数据库、网络 GIS、空间信息服务等方面研究;吴维,男,博士生。

2.1　概念层

从概念层上看,ArcSDE 是如何实现不同版本之间并发控制的呢?如图 2 所示,由基版本 Default 派生子版本 EditGroup,再由 EditGroup 派生两个子版本 Edit1 和 Edit2。用户 User1 和 User2 各自在自己版本里工作,当 User1 编辑完成,提交前需要进行冲突协调,可以选择编辑版本 Edit1 的父版本 EditGroup 或祖先版本 Default 作为冲突协调的目标版本。当 User1 选择 EditGroup 作为目标版本进行冲突协调时,因为编辑前版本和目标版本 EditGroup 一致,不会发生冲突,提交到目标版本 EditGroup 后,目标版本 EditGroup 更新为和编辑版本 Edit1 一致。当 User2 编辑完成后选择 EditGroup 作为目标版本进行冲突协调时,因为目标版本 EditGroup 已经被 User1 更新,和编辑前版本不一致,可能存在冲突。如果目标版本 EditGroup 和编辑版本 Edit2 对同一要素进行了修改则会发生冲突。将没有冲突的要素合并到编辑版本 Edit2,将存在冲突的要素根据用户的选择合并到编辑版本 Edit2,冲突协调完成后再提交到目标版本,从而实现了多用户编辑下的版本的并发控制。那么,版本更新又是如何实现的呢?

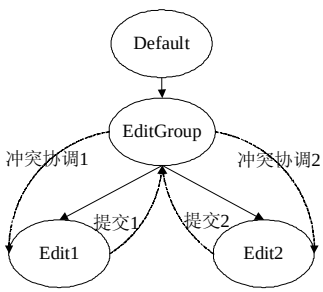


图 2　概念层上版本管理机制

2.2　中间层

ArcSDE 版本实质上是数据库某时刻的一种状态引用,每个版本必然引用一种状态。从中间层上看,ArcSDE 版本并发控制实质上是通过引用状态的更新来完成的。如图 3 所示,设版本 Default 当前引用状态是 0,EditGroup 引用状态是 1,Edit1 编辑完成保存后引用状态是 5,Edit2 编辑完成保存后引用状态是 7。当 Edit1 编辑完成后选择版本 EditGroup 进行冲突协调时,因为目标版本引用状态 1 和编辑前版本引用状态 1 一致,因此没有冲突,当编辑版本 Edit1 将当前版本的更新提交给目标版本 EditGroup 时,目标版本 EditGroup 的引用状态变为 5。当 Edit2 编辑完成选择目标版本 EditGroup 进行冲突协调时,因为目标版本引用状态 5 和编辑前版本引用状态 1 不同,可能存在冲突。如果状态谱系(1、2、4、5)和状态谱系(1、3、6、7)中对同一要素进行了修改,则存在冲突。若不存在冲突,则直接将目标版本 EditGroup 合并到编辑版本 Edit2 中;若存在冲突,则根据用户的选择合并到编辑版本 Edit2 中。不妨设冲突协调完成后当前状态为 9,再提交到目标版本 EditGroup,则此时目标版本 EditGroup 引用状态更新为 9,这样就通过引用状态的更新实现了多用户编辑下版本的并发控制。那么,ArcSDE 又是如何完成引用状态的更新的呢?

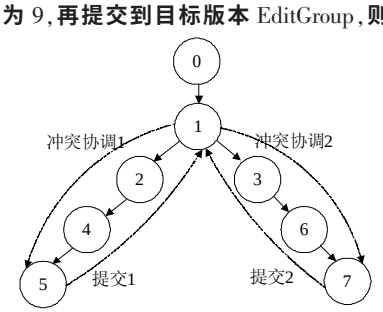


图 3　中间层上版本管理机制

2.3　物理层

ArcSDE 版本管理中引用状态实质上是数据库某一时刻的数据视图,ArcSDE 可以通过一些元数据表和版本化表来重建某一状态的数据视图。从物理层上看,ArcSDE 是通过不同版本引用状态的更新来实现不同的数据视图,进而完成不同版本的并发控制的。元数据表包括版本表(versions)、状态表(states)、状态谱系表(state_lineage)和更新记录表(mvtables_modified)。版本表记录每个版本的引用状态等信息,状态表记录每个状态的所有者、创建和关闭时间、父状态 ID 等信息,状态谱系表记录每个状态谱系的所有状态 ID,更新记录表记录每个基表对应的所有修改所在的状态 ID;版本化表包括基表(base table)、A 表(A<registration_ID>)和 D 表(D<registration_ID>)。基表保存原始空间数据,A 表保存某状态下增加或更新的要素,D 表记录删除或者修改的要素 ID。如图 4 所示,ArcSDE 根据基表用户名和表名通过注册 ID 关联所有更新的状态 ID,然后根据状态表、状态谱系表以及基表、A 表和 D 表生成每一个状态下的数据视图。编辑过程中,每增加一个要素,即往 A 表中写入一条记录;每删除一个要素,则往 D 表中写入一条记录;当修改一个要素,则先往 D 表写入一条记录,再向 A 表写入一条记录。

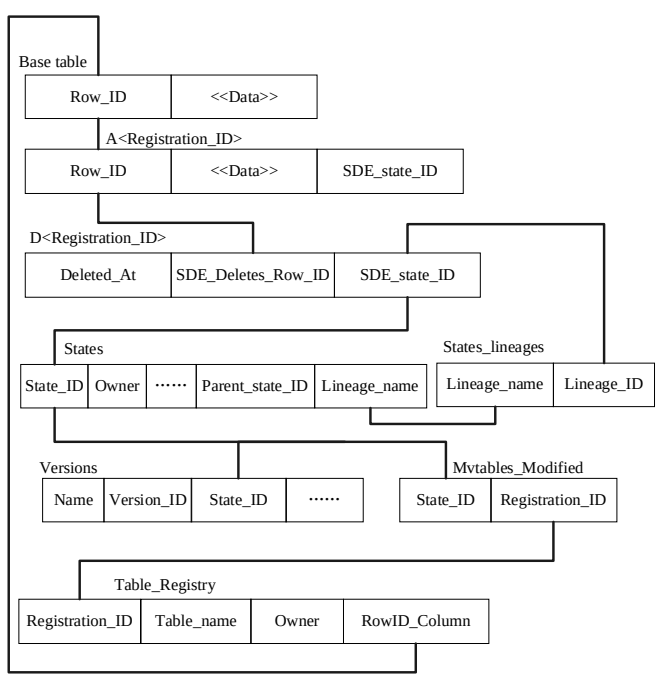


图 4　物理层上版本管理机制

因此,当在目标版本引用状态 5、编辑版本引用状态 7 和编辑前版本引用状态 1 之间进行冲突检查时,实际上是在 D 表和 A 表中检查状态谱系(1、3、6、7)和状态谱系(1、2、4、5)是否存在对同一行 ID 值的记录。例如,假设 D 表中存在 STATE_ID 为 2,Deleted_at 为 4,行 ID 为 85 的一条记录,A 表中存在 STATE_ID 为 4,行 ID 为 85 的一条记录(即在状态谱系 1、2、4、5 下修改了行 ID 为 85 的记录),同时 D 表存在 STATE_ID 为 6,行 ID 为 85 的一条记录(即状态谱系 1、3、6、7 中删除了行 ID 为 85 的记录),则会发生冲突。不妨设用户在冲

突协调过程则会在 D 表和A 表增加记录，同时状态也更新，设冲突协调完当前状态为 9，提交到目标版本 EditGroup，此时 Versions 表中版本 EditGroup 对应的 STATE_ID 值更新为 9，这样就通过不同的数据视图完成了引用状态的更新，实现了 ArcSDE 的版本管理机制。

从以上分析可以看出，ArcSDE 版本管理机制的本质就是通过建立数据库修改“痕迹”的增量记录来实现对数据库状态的更新，并通过引用状态的更新来完成版本的更新来协调冲突，从而实现多用户编辑下版本的并发控制。

3 ArcSDE 版本管理中空间数据访问方法

ArcSDE 版本管理很好地实现了空间数据的并发控制。但也正是由于其版本管理机制，在多用户编辑后访问 ArcSDE 空间数据的时候，不能采用传统的方法，如果直接用 PL/SQL 或用 ArcGIS 提供的二次开发组件对空间数据表进行空间查询和空间分析时，是得不到期望结果的。这是因为 ArcSDE 版本编辑后基表空间数据没有发生任何变化，更新记录在对应的 A 表和 D 表中，ArcSDE 通过元数据表和版本化表创建视图来访问各个版本编辑后的空间数据，而该视图并没有直接暴露给用户。那么，该如何访问 ArcSDE 多用户编辑下的空间数据呢？一种方法是利用 ArcSDE API 创建多版本视图，PL/SQL 查询和分析以及用 ArcGIS 二次开发组件进行空间查询和分析时都通过视图访问编辑后的空间数据。但该方法在对版本化的空间数据表进行反注册取消版本化后，编辑后的空间数据将会丢失，这是由于 ArcSDE 自身的机制造成的，其在进行反注册是没有将 A 表和 D 表数据更新到基表。

本文提出一种基表更新的方法，能够克服这些问题，很方便地实现对多用户编辑后空间数据的访问。基本思想是将 D 表和 A 表的数据按照一定算法更新到基表，然后清空 A 表和 D 表的数据，具体算法实现步骤如下：

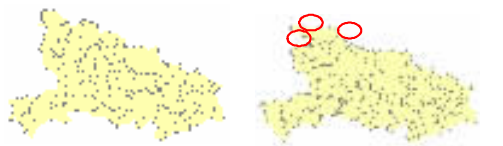
(1)基表对应的 D 表和 A 表的选择；通过查询注册表 Table_Registry 的记录集可以得到该用户名和表名的唯一注册 ID 号，基表对应的 D 表和 A 表的名称即由字符“D”和“A”尾随该注册 ID 号组成。

(2)基表中记录的更新；基表中记录的更新可以从编辑的三个方面考虑：增加对象、删除对象和修改对象。具体算法如下：从 D 表记录集中，按照字段 SDE_DELETES_ID 排序，依次获取记录的 SDE_DELETES_ID 字段值，该值即为对象的 ObjectID，对于 SDE_DELETES_ID 值相同的多条记录，取状态 ID 号 DELETED_AT 值最大的记录。然后依次检查 A 表是否存在该对象 ObjectID 值，如果不存在，则直接从基表中删除具有该 ObjectID 的对象；如果存在，则检查具有该 ObjectID 值的记录数，如果大于 1，则取 SDE_STATE_ID 值最大的记录，再检查该记录的状态 ID 值 SDE_STATE_ID，如果小于 D 表中该对象的状态 ID 号 DELETED_AT 值，则说明该对象已经被删除，此时不用更新基表；如果大于等于 D 表中该对象的状态 ID 号 DELETED_AT 值，则说明对基表中的该对象进行了编辑操作，需查找基表中的该对象，将 A 表中的数据更新到基表。然后依次检查 A 表中的其它记录，这些记录为编辑过程中新增加的对象，将 A 表中的对象插入到基表中。

该方法通过更新 A 表和 D 表的数据到基表，不仅应用简单方便，实现采用传统方法就能访问 ArcSDE 版本编辑后的空间数据，而且在 ArcGIS 中进行空间数据表反注册取消版本化时也不会造成数据丢失，同时也不影响 ArcGIS 对空间数据的访问，缺点是将最新的状态更新到基表，不能回溯历史版本的空间数据。

4 实验分析

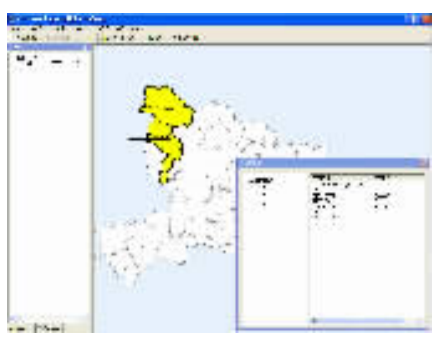
为了验证该方法的可行性和实用性，做了如下实验，实验数据为 1:25 万湖北省各县市矢量数据，用 ArcMap 编辑前后数据如图 5，图 5(a)为编辑前数据，图 5(b)为编辑后数据，编辑后删除了错误纳入湖北矢量数据里的河南的山阳县和浙川县以及陕西的白河县，图中用椭圆标识其位置。



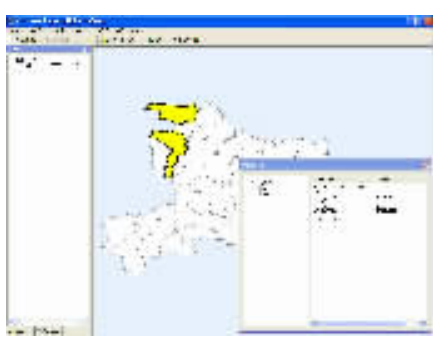
(a)编辑前 (b)编辑后

图 5 1:25 万湖北省各县市矢量数据

利用 Visual C++开发了一个实验平台，并按照上章算法，编写了一个更新基表程序，利用 ArcGIS 提供的二次开发组件 AO 对编辑后空间数据表进行空间查询，在执行该更新程序前后查询结果如图 6，图 6(a)显示在运行该更新程序前查询结果为 4 条记录，结果错误；图 6(b)显示运行该更新程序后查询结果为 2 条记录，结果正确。可见该方法对于 ArcSDE 多用户编辑下的空间数据访问非常方便实用。



(a)运行更新程序前



(b)运行更新程序后

图 6 运行更新程序前后对编辑后数据查询结果

5 结论

ArcSDE 通过建立数据库修改“痕迹”的增量记录来实现对数据库状态的更新，并通过引用数据库状态建立版本，用户在其建立的数据库版本里进行编辑修改时，并不用关心其他用户是否也在对同一数据进行操作。当用户完成了他的(长)事务处

(下转 24 页)