Versioning 101

Essential information about ArcSDE geodatabases

By Derek Law, ESRI Product Management

Editor's note: *Rather than a comprehensive discussion of versioning, this article presents the key concepts about ArcSDE geodatabase versioning including version creation; version workflows; reconciliation and post; states; and compress operations.*

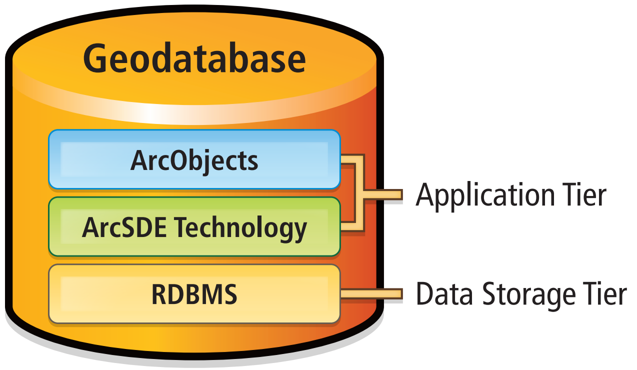


Figure 1: At a conceptual level, ArcSDE geodatabases have a multi-tier architecture that implements advanced logic and behavior in an application tier on top of a data storage tier. The application tier consists of ArcObjects and ArcSDE technology, while the data storage tier is comprised of database management system (DBMS) software. ArcSDE geodatabases utilize the simple, formal data model of a DBMS for storing and managing information in tables. They also leverage DBMS support for multiuser transaction processing.

What Is **Versioning?**

Versioning is the mechanism that enables concurrent multiuser geodatabase editing in ArcSDE geodatabases. It uses an optimistic concurrency data-locking model, which means no locks are applied to affected features and rows during long transactions. It is the default editing environment in enterprise ArcSDE geodatabases and supports complex editing workflows that are required by enterprise GIS systems.

Versioning records and manages states of individual features and rows as they are edited while preserving integrity in the database. It is the basis for multiple users accessing and editing data simultaneously in enterprise ArcSDE geodatabases. Conceptually, a version of the geodatabase represents an alternative, independent, persistent view of the geodatabase. It supports multiple concurrent editors and does not involve creating a copy of the data. A version references a specific state of the geodatabase. It contains all the datasets in the geodatabase and evolves over time. Users access data in an enterprise ArcSDE geodatabase through a version. Behind the scenes, simple queries in the underlying DBMS are used to view and work with the referenced state for a particular point in time or to see an individual user's current edits.

Note: Database transactions represent a package of work that makes changes to databases. Most database transactions occur within a very short time period, often within seconds. A state is a unit of change (i.e., an edit) that is performed on data in the geodatabase. It represents a discrete snapshot of the database whenever a change is made.

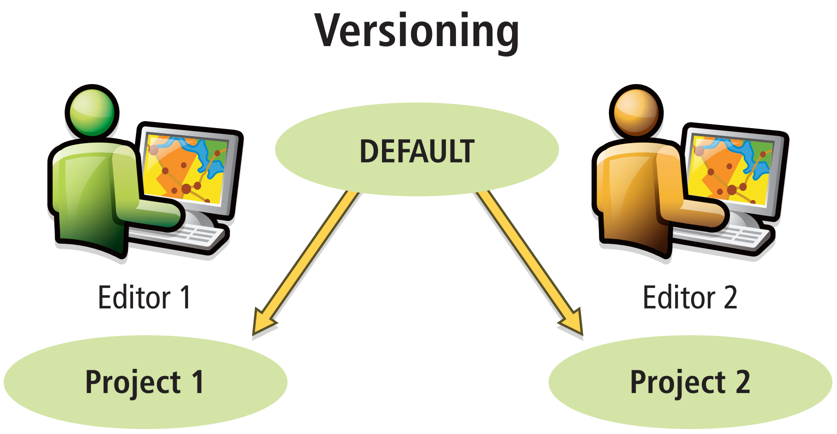


Figure 2: Versioning allows multiple users to work on the same geodatabase. DEFAULT is the parent version and Project 1 and Project 2 are child versions.

Enterprise ArcSDE geodatabases provide support for many users creating and maintaining large amounts of GIS data in a central location. In many cases, multiple users need to edit the same data at the same time. In other words, they require concurrent multiuser geodatabase editing. The nature of the spatial relationships and connectivity that define geographic data requires that edit sessions for geospatial data typically span long periods of time (e.g., hours, days, or weeks). These long edit sessions can be thought of as long transactions in the DBMS. Additional user requirements include the ability to undo or redo changes, the capability to develop alternative application design proposals without affecting the published geodatabase, and a mechanism to manage how the data and the geodatabase have changed over time.

The DEFAULT Version

Every enterprise ArcSDE geodatabase has a default version named DEFAULT that is owned by the ArcSDE administrator. The DEFAULT version always exists and cannot be deleted or renamed. It is the root version and, therefore, the ancestor to all other versions in the geodatabase. In many workflow strategies, it is the published version of the geodatabase, representing the current "public" end-user view of the geodatabase. The DEFAULT version is typically maintained and updated over time by incorporating changes to it from other versions. Like any other version, it can also be directly edited.

Enterprise ArcSDE geodatabases can have many versions. A new version (child version) is created from an existing version (parent version). When a new child version is first created, it is identical to its parent. However, over time, parent and child versions may diverge as changes are made to each version. In the figures in this article, Project 1 is a child version of DEFAULT, its parent version (Figure 2).

When editing geodatabase datasets within the versioned editing environment, each version will seem to have its own copy of the data. As a dataset is edited in one version, it will appear differently when viewed in another version. Regardless of how many versions exist within the geodatabase, each dataset is only stored once in the DBMS. Behind the scenes, ArcGIS leaves each dataset in its original state during editing. All changes to a dataset are recorded in associated tables known as delta tables. Delta tables are also commonly called the A (adds) and D (deletes) tables. Each table or feature class will have an associated pair of these delta tables when they are registered as versioned within ArcCatalog.

Every version has an owner, description, parent version, associated database state, and level of user access. There are three levels of access to a version:

- Private: Only the owner can view and edit.
- Protected: All users can view, but only the owner can edit.
- Public: All users can view and edit.

The access level for the DEFAULT version is public by default. It is recommended that its access level be set to protected to ensure data in an enterprise ArcSDE geodatabase is not accidentally corrupted or lost. This means that only the ArcSDE administrator can edit or post changes to the DEFAULT version.

Versions are beneficial for workflow management in enterprise ArcSDE geodatabases, such as modeling different discrete stages in a GIS project (e.g., each stage is represented by a version) and modeling what-if scenarios without affecting the original datasets. They provide a framework for security management and quality assurance in data editing, and they also support historical archiving and geodatabase replication.

Versioning Workflows

Versioning supports many complex editing workflows and can be easily adapted and/or customized to meet the business requirements of any organization. Three example business workflows using versions are shown in Figure 3. The simplest workflow is to have concurrent editors directly editing the DEFAULT version (see Figure 3A). Another option is to create a separate version (e.g., multiple projects) for each editor in the geodatabase (see Figure 3B). To ensure that the publication view of the geodatabase is protected from accidental data corruption, many organizations create a quality assurance (QA) version from the DEFAULT version (see Figure 3C). The QA version would be maintained by a data quality manager and would regulate all edits that are applied back to DEFAULT. Note that each versioning workflow strategy has its own advantages and disadvantages. It is important to use a strategy that best meets the requirements of the business workflow.

Continued on page 44

Versioning 101

Continued from page 43

Database States and Versions

A version references a specific database state—a unit of change that occurs in the database. Every edit operation performed in the geodatabase creates a new database state. An edit operation is any task or set of tasks (e.g., additions, deletions, or modifications) undertaken on features and rows. State ID values apply to any and all changes made in the geodatabase.

Initially, the DEFAULT version points to state 0. As edits are made to datasets in the geodatabase, the state ID will increase incrementally. In general, the state ID increases by a value of one for each edit operation. However, there are some exceptions where state ID may increase by a value greater than one, such as during a reconcile operation.

Figure 4 illustrates the state ID increasing as edits are made to two feature classes in the geodatabase. An edit session is started on a polygon and a point feature class in the DEFAULT version (see Figure 4A). A new polygon feature is added (see Figure 4B). Next, an existing point feature is deleted (see Figure 4C). Lastly, an attribute property for two polygon features is modified in one operation, then the edit session ends and edits are saved (see Figure 4D). For each edit, the state ID incremented by a value of one. The DEFAULT version now points to state 3.

In the previous example, the geodatabase state ID increased because editing was performed through the DEFAULT version. If the scenario had included another edit session with another version, the state ID would have also grown by the number of edits performed in the second edit session.

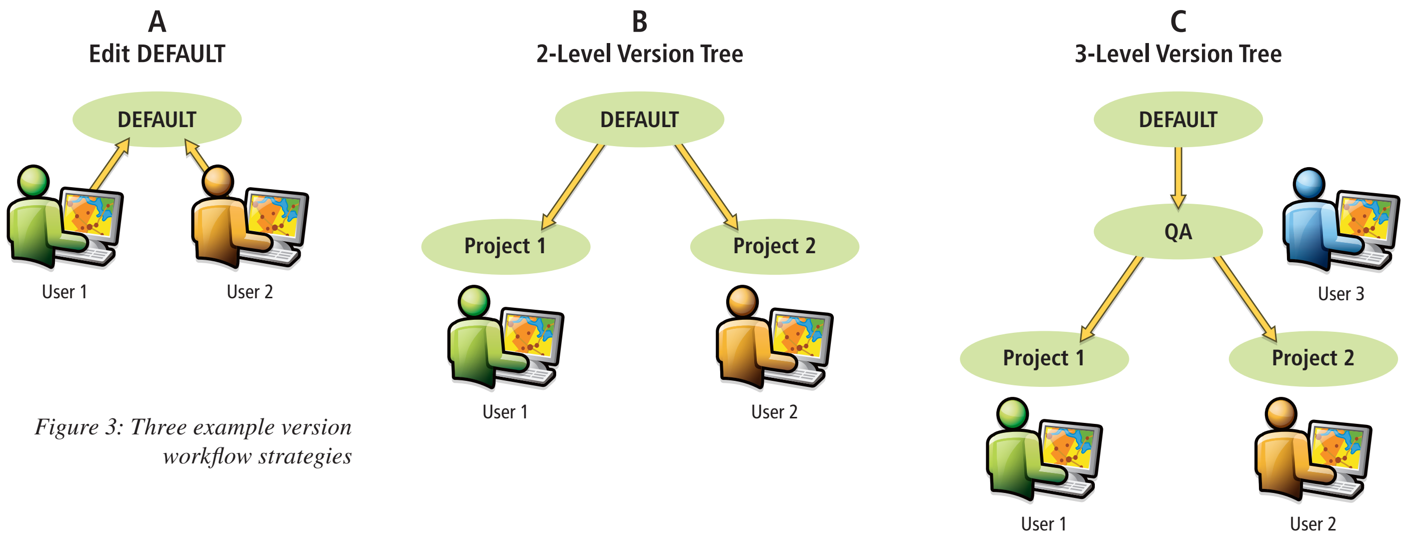


Figure 3: Three example version workflow strategies

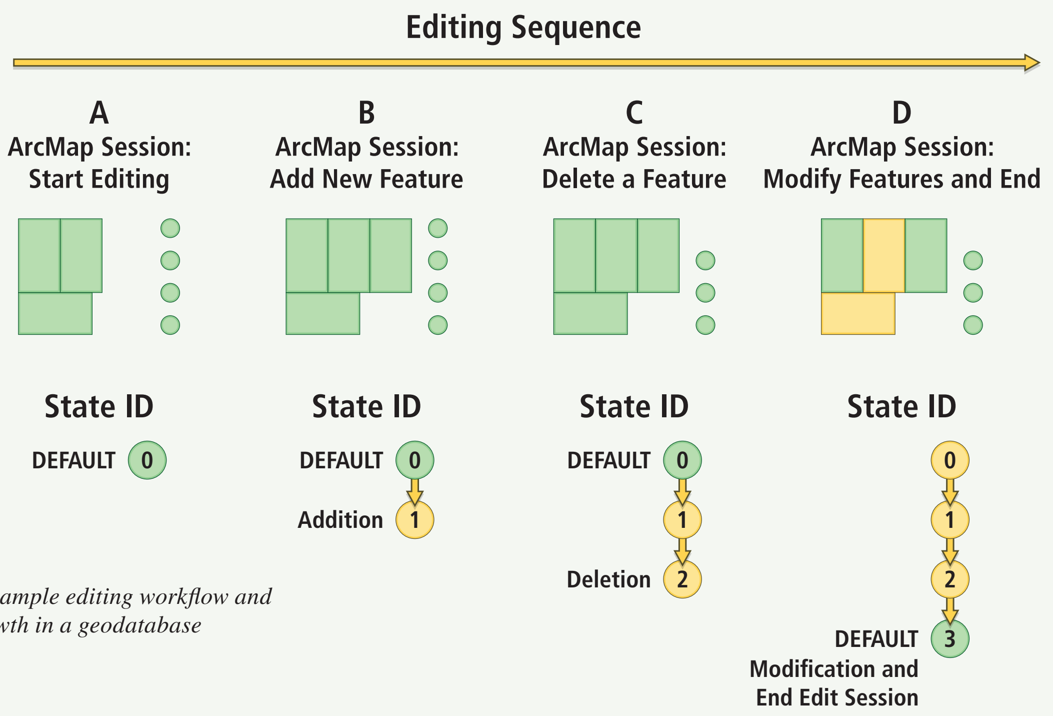

Editing Sequence
A
ArcMap Session: Start Editing
B
ArcMap Session: Add New Feature
C
ArcMap Session: Delete a Feature
D
ArcMap Session: Modify Features and End
State ID
DEFAULT 0
State ID
DEFAULT 0
Addition 1
State ID
DEFAULT 0
1
Deletion 2
State ID
0
1
2
DEFAULT 3
Modification and End Edit Session

Figure 4: Example editing workflow and state ID growth in a geodatabase

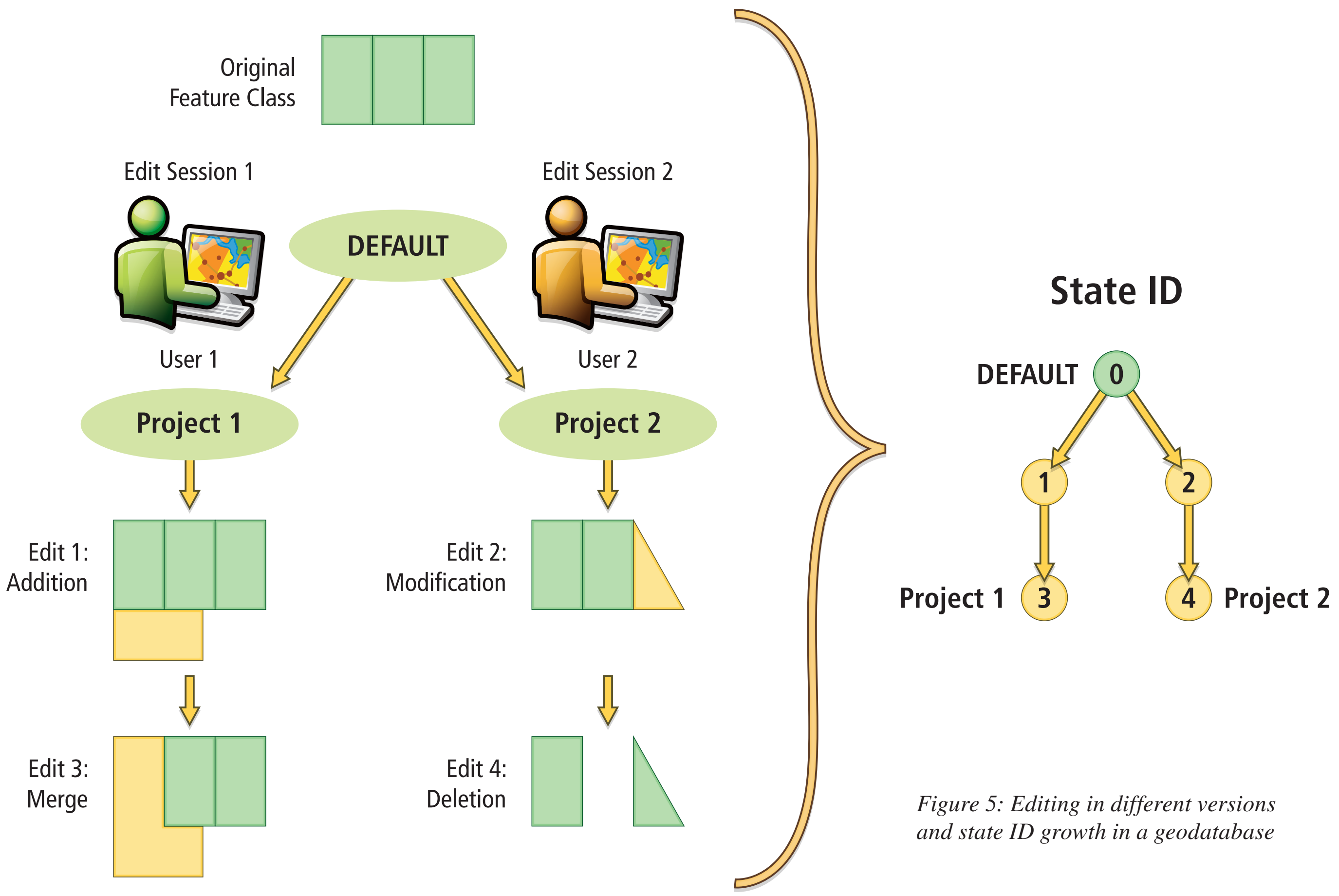


Figure 5: Editing in different versions and state ID growth in a geodatabase

Figure 5 illustrates a two-level version tree editing workflow. Two versions (Project 1 and Project 2) were created from DEFAULT. Initially, they were both exactly the same as DEFAULT and pointed to state 0. As User 1 starts an edit session and adds a new feature, the state ID increases by one. When User 2 begins an edit session, a new separate branch is created from DEFAULT to record edits. In this scenario, these editing operations occur as follows:

1. User 2 modifies an existing feature.
2. User 1 merges two features into a single feature.
3. User 2 deletes a feature.

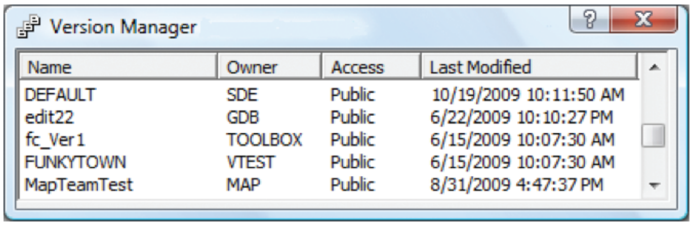
The order of these edit operations is recorded with corresponding state IDs that represent each change made in the geodatabase.

State IDs in the geodatabase can be conceptually thought of as being maintained in a treelike structure. This structure, called a state tree diagram, is a logical map between states in a geodatabase. As the geodatabase is edited over time, a lineage of states is maintained that identifies all the changes that have occurred in a version. To determine the lineage for a specific version, follow the most direct path up the state tree to state 0.

At the end of the example in Figure 5, DEFAULT points to state 0. Project 1 points to state 3 and has a lineage of 3, 1, 0; and Project 2 points to state 4 and has a lineage of 4, 2, 0. Version parent-child relationships can be derived from the state lineages. Both the Project 1 and Project 2 versions reference newer state IDs in contrast to DEFAULT, and their lineages contain the state ID that DEFAULT references: state 0. This indicates that DEFAULT is likely an ancestor version to them. In this case, DEFAULT is the parent version to both Project 1 and Project 2.

Version Management

The number of versions that exist within an enterprise ArcSDE geodatabase can be seen in the Version Manager dialog box in ArcCatalog and ArcMap (as shown in Figure 6). Version Manager will show all versions in a geodatabase, except those marked as private—those versions will only be visible to their respective owners.



Name	Owner	Access	Last Modified
DEFAULT	SDE	Public	10/19/2009 10:11:50 AM
edit22	GDB	Public	6/22/2009 10:10:27 PM
fc_Ver1	TOOLBOX	Public	6/15/2009 10:07:30 AM
FUNKYTOWN	VTEST	Public	6/15/2009 10:07:30 AM
MapTeamTest	MAP	Public	8/31/2009 4:47:37 PM

Figure 6: Version Manager dialog box in ArcGIS

Versions can be created or deleted from the Version Manager dialog box. As stated earlier, it is important to implement a versioning workflow strategy that best fulfills the requirements of the business workflow. The complexity of managing versions in a geodatabase increases as more versions are used.

Edits that are made within a version are isolated to that version until its owner or the ArcSDE administrator decides to merge the changes into another version. The exception to this statement is schema changes. When the schema in a version is changed—for example, by adding a new field to a table—the change applies to all other versions. The operational task of properly merging edits between versions in an enterprise

Continued on page 46

Versioning 101

Continued from page 45

ArcSDE geodatabase is achieved in ArcGIS through two operations: reconciling and posting. These two operations are typically performed in tandem (i.e., reconciling followed by posting) to combine edits from one version with another version.

Reconcile

Reconciling is the first step in merging edits between two versions. In this process, edits from an ancestor version (called the target version) are brought into the version being edited in an edit session in ArcMap (called the edit version). A target version can be any version in the direct ancestry (i.e., in the lineage) of the version being edited. For example, referring back to the state tree diagram in Figure 5, DEFAULT is an ancestor version to Project 1, because DEFAULT points to state zero (0), which is part of Project 1's lineage: 3, 1, 0. The reconcile process involves merging edits from the target version into the edit version, as shown in Figure 7.

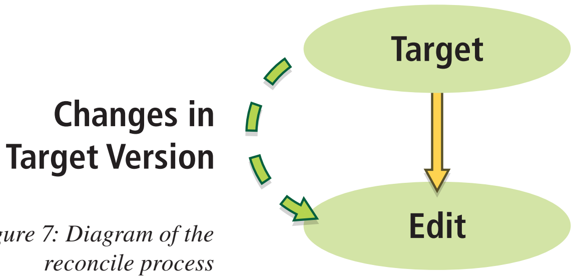


Figure 7: Diagram of the reconcile process

To perform a reconcile operation, there can only be one user editing the edit version. Since a version spans all the versioned objects in the geodatabase, any features or rows that were modified in the target version will be merged into the edit version. As the majority of these features and rows are not likely to be in conflict, they will merge seamlessly into the edit version. For example, if a new polygon feature was added in the target version, after the reconcile process, the polygon feature would appear in the edit version. The editor would then decide whether to save the changes in the edit version.

At a conceptual level, a reconcile process involves merging edits from one branch of the state tree with a different branch of the state tree. Figure 8 shows an example reconcile operation between two versions: QA and a child version of QA, Project 1. When Project 1 is initially created, it is the same as QA. An edit session is started on QA, a new feature is added, and that edit is saved. Next, a separate edit session is started on Project 1 and a new feature is added, but those changes are not saved yet. The editor for Project 1 then performs a reconcile process with the QA version, with QA as the target version and Project 1 as the edit version. All features that have been added, deleted, or modified in the QA version will be brought into the Project 1 version.

The reconcile process can occur either implicitly or explicitly.

- Implicit—A reconcile operation will occur implicitly when there are multiple editors editing the same version (see Figure 3A). Each editor maintains his or her own branch in the state tree for the duration of an edit session. When an editor attempts to save the edits in his or her edit session, a reconcile operation occurs to push the edits in the editor's branch to the branch currently referenced by the version. With multiple editors in one version, each time edits are saved, the reconcile process is executed. There is no choice of when the reconcile operation happens; it always occurs when edits are saved.

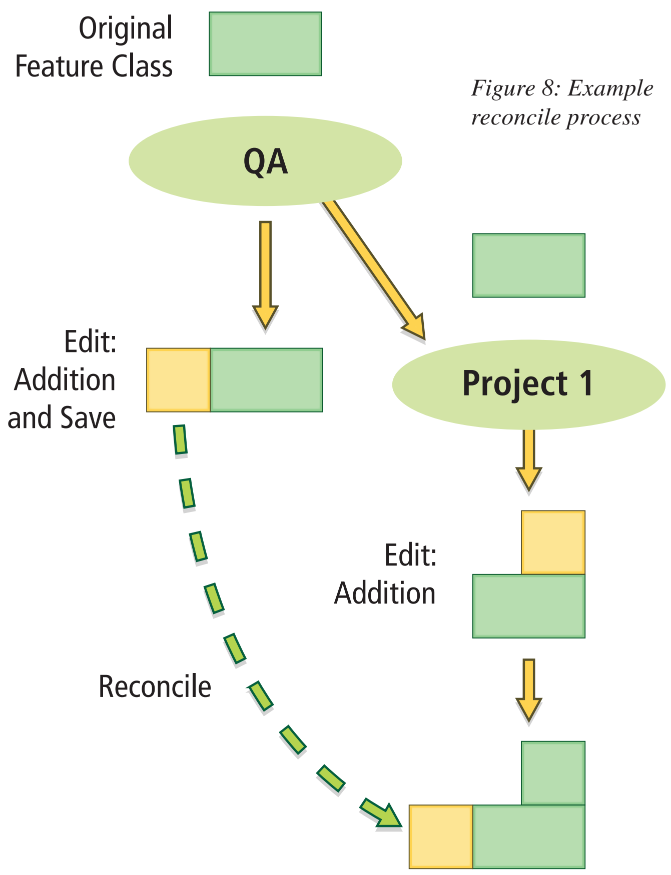


Figure 8: Example reconcile process

- Explicit—When performing a reconcile operation between different versions (as shown in Figure 8), an editor chooses when he or she wants the reconcile process to be executed. This differs from an implicit reconcile process, which occurs when edits are saved.

Regardless of how reconciliation occurs the mechanics are the same. The difference between implicit and explicit reconcile processes is when the reconcile process occurs and how the conflict detection options are specified.

Possible Conflicts during Reconciliation

In some cases, a small percentage of features and objects may be in conflict when comparing the target version and the edit version. Conflicts can occur in two editing scenarios: when the same feature is updated in both the target and edit versions or when the same feature is updated in one version and deleted in the other.

In practice, conflicts will not be encountered frequently for most reconciliation operations, because in many business workflows, versions typically represent different projects with distinct geographic areas (e.g., editing different areas of a map). Therefore, the likelihood of conflicts occurring is rare. Conflicts usually arise when editors are editing features that are in close proximity.

When performing a reconcile operation, ArcGIS finds conflicts in one of two ways: by object ID or by attribute. Conflict by object ID means that a feature is identified to be in conflict when any part of it (e.g., geometry or attributes) has been edited in both the target and edit versions. Conflict by attribute means that a feature is identified to be in conflict only when the same attribute (e.g., the same attribute field) has been edited in both the target and edit versions.

Default conflict resolution policies can be set to automatically resolve conflicts in favor of either the target version or the edit version. There is also the option to have the editor of the edit version resolve detected conflicts manually, by reviewing each conflict using the interactive Conflicts Resolution dialog box in ArcMap. Each conflict can be closely examined, and the editor decides whether to apply the target version edit, keep the edit version edit, or revert the feature to its state at the beginning of the edit session (i.e., the common ancestor state). After all conflicts (if any exist) are resolved, the reconcile process is considered completed and the editor can save edits and continue editing or proceed with a post operation.

Post

This is the second step when merging edits between two versions. This process must always follow a reconcile operation. A post process synchronizes the current edit version with the target version. All edits made in the edit version are saved into the target version, making both versions identical (see Figure 9).

Unlike a reconcile process, posting cannot be undone once it is performed because the changes are applied to a version outside of an edit session. Figure 10 conceptually illustrates a post operation between the QA and Project 1 versions. The post operation is performed immediately following the reconcile process previously discussed and shown in Figure 8. After posting, the new feature added in Project 1 (edit version) is merged into the QA version (target version). At the end of the reconcile and post workflow, both the QA and Project 1 versions will have the same view of the geodatabase. In other words, they are considered to be identical. At this point, the editor of Project 1 has the option to continue to make more edits in the edit session, then perform another reconcile and post process to synchronize the two versions or simply save edits and stop the edit session on the Project 1 version.

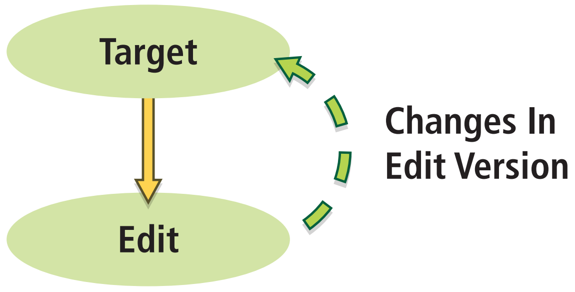


Figure 9: Diagram of the post process

Continued on page 48

CartêGraph^mobile users have lowered their sign inspection operating costs by up to 47%.

Isn't it about time that you did, too?

Simplicity. Capability. Affordability.

With the power of CartêGraph^mobile solutions, it's all within your reach.

CartêGraph mobile

800.688.2656
www.cartegraph.com

Versioning 101

Continued from page 47

Compress

Over time, an actively edited enterprise ArcSDE geodatabase typically accumulates hundreds of thousands of state IDs (representing edits stored in delta tables) and a deep and complex state tree. This can negatively impact performance. Periodically, the ArcSDE administrator must compress the ArcSDE geodatabase to remove any states not referenced by a version. A compress operation can reduce the depth of the state tree and helps maintain performance.

Compressing an ArcSDE geodatabase never removes data that is accessed through a version's lineage. It cleans up only unused data. A compress operation is implemented as a series of potentially large DBMS transactions that remove and renumber states inside one database transaction to ensure that the DBMS can restore the geodatabase to a consistent state. Figure 11 shows what happens to the state tree when a geodatabase is compressed. Before the compress operation, there are three versions: DEFAULT, Project 1, and Project 2, which reference states 0, 3, and 4, respectively. A compress operation removes states that are not directly referenced by a version (states 1 and 2 in this example) and deletes them from the state tree. The edits they reference in the delta tables are also removed. Unused data in the geodatabase is deleted. A compress also moves entries in delta tables common to all versions into the base tables, reducing the amount of data the DBMS will need to search through for version queries. Both actions help maintain performance in the geodatabase.

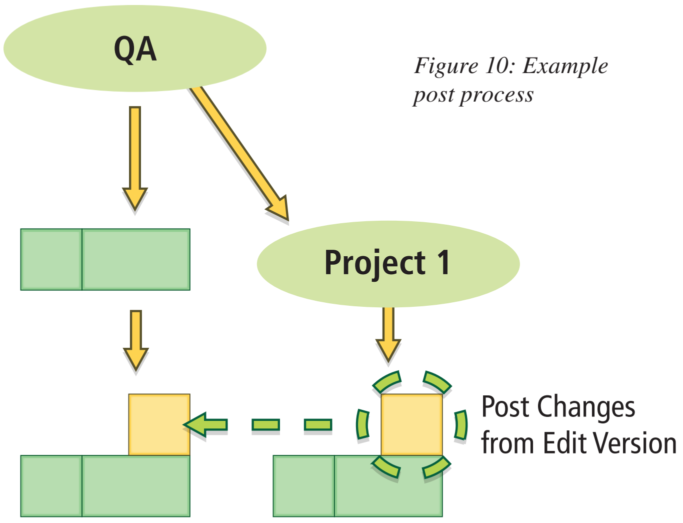


Figure 10: Example post process

Other Editing Models

At the 9.2 release, two additional editing models were added to ArcSDE geodatabases: nonversioned editing and versioned editing with the option to move edits to base. Nonversioned editing enables users to perform edits directly on the base tables of geodatabase datasets. It is analogous to a standard database transaction model for editing geospatial data.

Versioned editing with the option to move edits to base enables users to perform versioned editing as discussed in this article but also supports some nonversioned editing functionality. This editing model functions like regular versioned editing except when editing the DEFAULT version. When the DEFAULT version is edited through this editing model, changes are made directly to the base tables of geodatabase datasets. Also, when versions are reconciled and posted back to the DEFAULT version, edits in the delta tables are moved to the base tables.

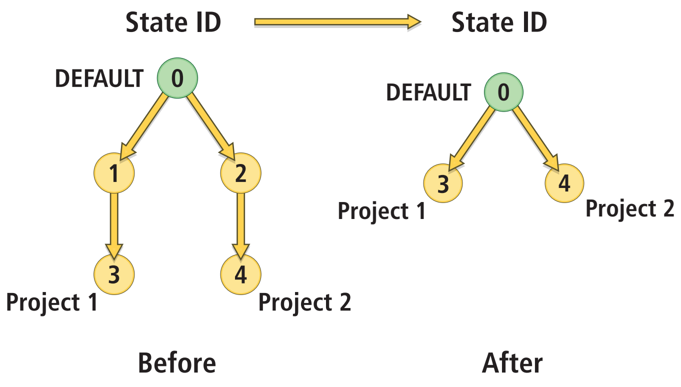


Figure 11: State tree diagram before and after a compress operation

Both editing models have some limitations on the types of datasets that can be edited and do not support geodatabase replication. More information on both editing models can be found in the ArcGIS Desktop Help Online documentation.

Conclusion and Resources

This article discussed the main concepts related to versioned editing in an ArcSDE geodatabase. It presented many of the common terms and concepts regarding concurrent multiuser editing in ArcSDE geodatabases. Versioning is the framework that supports historical archiving and geodatabase replication functionality. When implementing versioned editing, it is important to select the versioning workflow that best meets the organization's business requirements.

For more detailed information on the mechanics of how versioning works in an ArcSDE geodatabase, read the ESRI white paper *Versioning*, available from support.esri.com.

For more background on how state trees behave in versioned editing, read ESRI Knowledge Base article #32352—FAQ: How does the state tree change during an edit session? also available from support.esri.com.

For more examples on versioning workflow strategies, read the ESRI white paper titled *Versioning Workflows,* also available from support.esri.com.

For some tips and tricks on versioning, listen to the ESRI Instructional podcast *ArcSDE: Top Five Versioning Myths* available on esri.com.