

Using Compress on ArcSDE Geodatabases with Replicas

Introduction

[Geodatabase replication](#) is built on top of [versioning](#). With replication, versioning is used during synchronization to determine which changes to send between [replicas](#), and where to receive changes. When creating a replica, the replica version is explicitly defined and can be either the default version or a named versioned. When sending changes during synchronization, edits that have been made to the replica version are sent. When changes are received during synchronization, they are applied to the replica version.

When a replica is created system versions are also created by ArcGIS behind-the-scenes. These system versions are used by ArcGIS during synchronization. The synchronization process may remove existing system versions or create new ones. Directly interacting with these versions can result in errors or inconsistent results when using replication. System versions are hidden by design to protect against this. Hidden versions do not appear in ArcGIS and are not returned by ArcObjects (e.g. when using custom code).

Understanding the details of how these system versions are used by ArcGIS is not necessary to successfully implement geodatabase replication. However, these versions can have an effect when compressing an ArcSDE geodatabase. This article describes several best practices for handling system versions when compressing an ArcSDE geodatabase that contains replicas.

Compressing an ArcSDE Geodatabase

Data in a feature class or table that is not registered as versioned exists in a base table. There may actually be several tables depending on the [storage type](#), but for this article we will refer to them collectively as a “base table”. When a feature class or table is registered as versioned, [delta tables](#) are created in the geodatabase and associated with the feature class or table. When edits are made in ArcGIS to the feature class or table, the changes are stored in their associated delta tables. When you view the data, the contents of the base table are queried along with the contents of the delta tables for the version you are working with. If you have several versions, each with different edits, these queries return different results depending on the version.

Over time, changes accumulate in these delta tables that may no longer be needed. This increases the size of the geodatabase and can affect performance. [Compress](#) is a process a geodatabase administrator runs periodically that performs the following to remove unneeded changes from the delta tables:

- 1) Looks for and removes changes that are no longer accessed by a version.
- 2) Looks for changes that are accessed by all versions, applies them to the base table and then removes them from the delta tables.

The following examples show cases where unneeded changes accumulate and how compress works on versions in these cases. In each diagram, the content of the base table is shown at the top of the diagram. Changes in the delta tables are also shown and described with a statement starting with the word edit (e.g. Edit1: parcel 3 is moved). Changes that will be removed from the delta tables when compress is applied are drawn grayed out.

Example 1: Versioned editing and compress

In this example, the ArcSDE geodatabase has a parcels feature class that has 3 features. The feature class is registered as versioned which creates delta tables in the geodatabase. A new version named project1 is created from the default version. This version is created so that a group within the organization can independently make edits to the parcels feature class. Since there are no changes in the delta tables, the same 3 parcels from the base table are displayed regardless of which version you are connected to (figure 1).

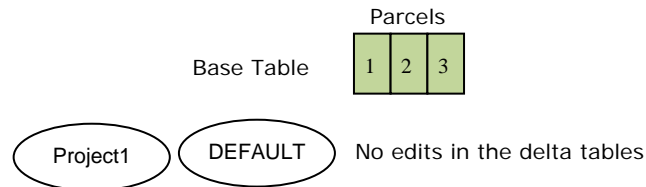


Figure 1: No edits have been applied to the parcels

An editor then makes an edit in the default version where parcel 3 is moved. At the same time, another editor working with the project1 version adds parcel 4. These edits are stored in the delta tables. When connecting to the default version and viewing parcels, ArcGIS queries the geodatabase such that parcels 1 and 2 from the base table plus the edit to parcel 3 from the delta tables are returned. When connecting to the project1 version, ArcGIS queries the geodatabase such that parcels 1,2 and 3 from the base table as well as parcel 4 from the delta tables are returned (figure 2).

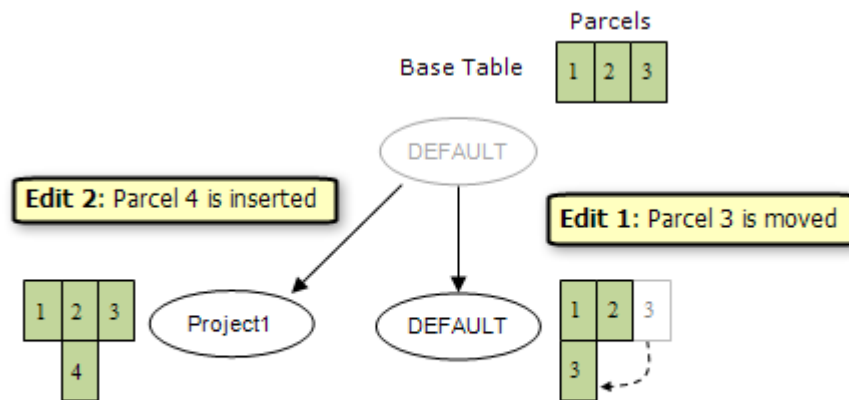


Figure 2: Parcel 3 is moved in the default version (Edit 1) and parcel 4 is added to the project1 version (Edit 2).

At this point, the work in the project1 version is complete and it is ready to be merged with the default version. This is done by performing a [reconcile](#) and [post](#).

The reconcile process moves changes from the default version into the project1 version. This is done by moving the project1 version under the default and copying the edits it references (i.e insert of parcel 4). If the same features have been edited in each version, a [conflict](#) occurs and a decision has to be made as to which change to accept. In this case there are no conflicts (figure 3).

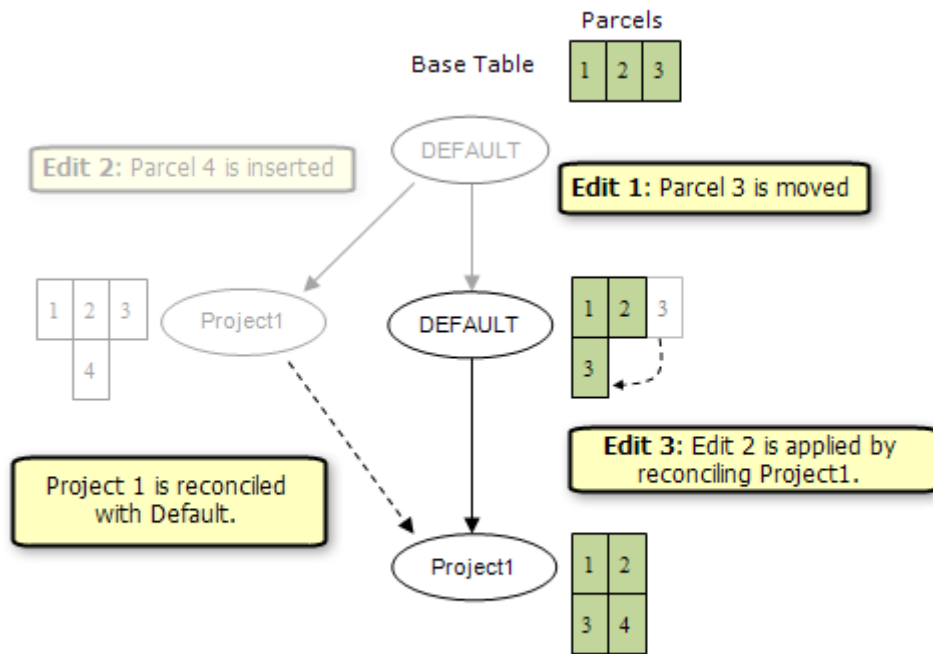


Figure 3: Project1 is reconciled with Default. This results in moving Project1 under Default and copying the edits it references (**Edit 3**). Edit 2 will be removed during compress.

In figure 3, edit2 is grayed out indicating that if compress is called at this point the change will be removed from the delta tables. In this case, the change is removed because it is no longer referenced by any version. The project1 version, which was the only version referencing edit2, was moved under default by reconcile and no longer references it.

Once reconcile is complete, post can be executed. The post process moves changes from the project1 version into the default version. Both versions now access the same edits in the delta tables. When connecting to either the default version or the project1 version and viewing parcels, ArcGIS queries the geodatabase such that parcels 1 and 2 from the base table and parcels 3 and 4 from the delta tables are returned (figure 4).

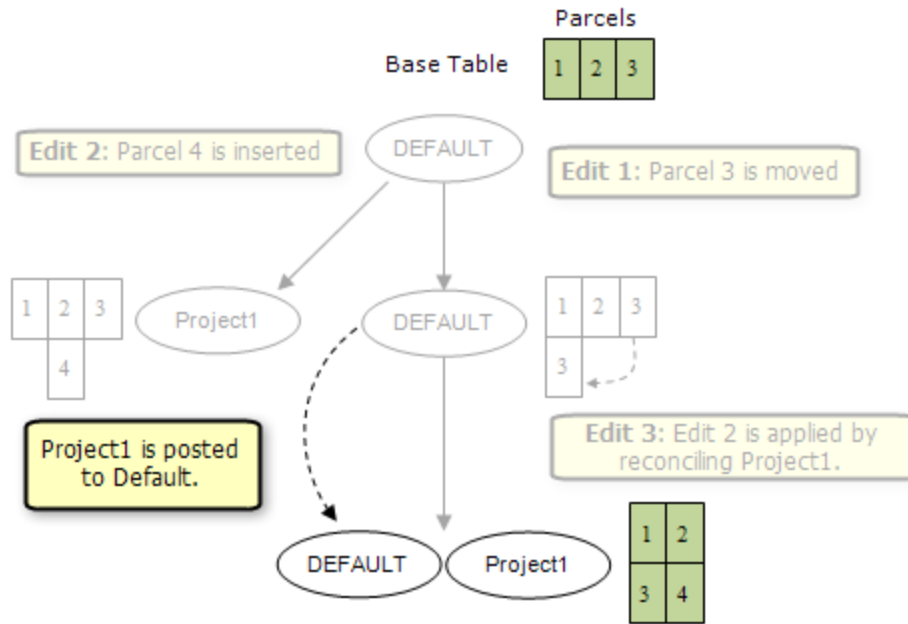


Figure 4: The default version is posted with the project1 version. Compress will now move Edit1 and Edit3 to the base table.

In figure 4, edit1 and edit3 are grayed out indicating that they will be removed from the delta tables by a compress. Since edit1 and edit3 are referenced by all versions in the geodatabase (i.e. project1 and default), compress will move them to the base table and then remove them from the delta tables.

A compress is then run on the geodatabase where edit1, edit2 and edit3 are removed from the delta tables for the reasons described above. The project1 version is also manually deleted since the work has been completed and the changes reconciled and posted into default. Figure 5 shows the results of the compress.

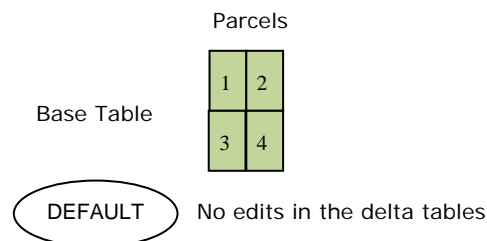


Figure 5: Edit1, Edit2 and Edit3 are removed from the delta tables by compress

Users accessing parcels in the default version before the compress (figure 4) and after the compress (figure 5) will see the same results. However, after to the compress process, there are no edits in the delta tables. Therefore, the geodatabase is smaller and less data is accessed to return the same results.

This example describes a geodatabase with a single version and a few edits. In many cases, however, a multiuser editing environment involves many edits and multiple versions. In these cases an [automated reconcile and post](#) process is normally used.

Example 2: Versioned editing and compress with a 1way replica

This example starts like example 1 except in this case a 1 way replica (Replica1) is created with the default version as the replica version. The replica creation process creates a system version from default for replica1 in the parent geodatabase. A system version works like other versions except that it is associated with a replica and is maintained by ArcGIS. Figure 6 shows the parent replica geodatabase with the system version (Replica1). System versions are represented in the diagrams with the replica name and a dashed outline.

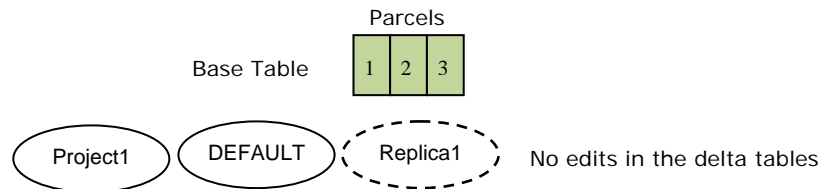


Figure 6: A system version is created when Replica 1 is created. No edits have been applied to the parcels at this point.

The same activity from example 1 up to figure 4 is then applied.

- An editor moves parcel 3 in DEFAULT
- Another editor adds parcel 4 in the Project1 version
- A reconcile and post is performed between Project1 and DEFAULT

Figure 7 shows the results.

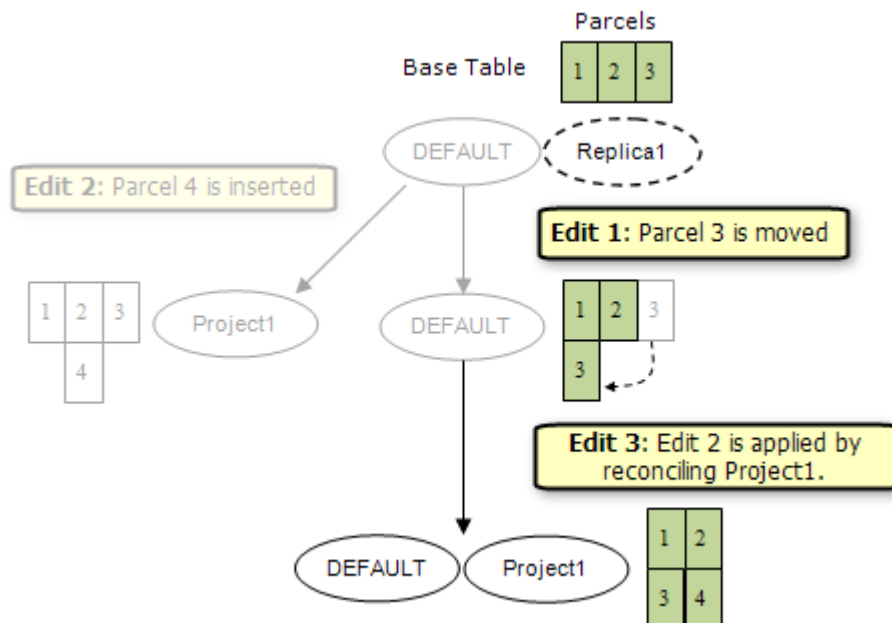


Figure 7: Changes up to figure 4 from example 1 are applied. Parcel 3 is edited in the default version, parcel 4 is added in the project1 version and the project1 version is reconciled and posted with default.

If you compare figure 4 and figure 7, you will see that if compress is called at this point it will be much less effective as than it was in example 1. In this example, only edit 2 will be removed from the delta tables instead of all edits at the same point in example 1. Edit 1 and edit 3 cannot be moved to the base table in figure 7 because they are not referenced by all versions (Edit1 and edit3 are not referenced by the Replica1 system version).

Next, replica1 is synchronized where changes are sent from this geodatabase to the [relative replica geodatabase](#). The synchronization process sends edit 1 and edit 3 to the relative replica since these changes were applied to the replica version (default) after the replica was created. In this case, we are performing a [connected synchronization](#) where an [acknowledgement](#) is automatically applied after the relative replica receives the changes. Upon receiving the acknowledgement, the existing system version is removed and a new one created from default. Figure 8 shows the results after the synchronization.

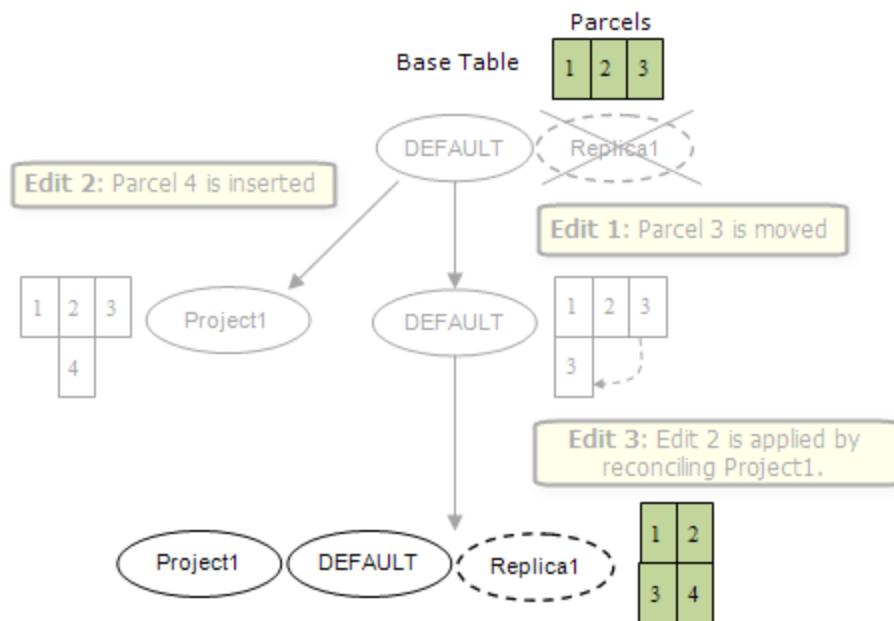


Figure 8: Replica1 is synchronized. This removes the old Replica1 system version and creates a new one from default.

Figure 8 shows that a compress executed at this point will be much more effective. Here, edit1 and edit3 are now referenced by all versions and thus they will be applied to the base table and removed from the delta tables.

Best Practice 1: Plan to synchronize replicas where you send changes on a regular basis. As the example shows, this process removes system versions that may be limiting the effectiveness of compress. Also note that when using [disconnected synchronization](#), the system versions are only removed after changes are sent and an acknowledgement message is received. One approach is to schedule synchronizations to run on a regular basis, for example, once a week or once a night. Another approach is to define the synchronization schedule around times when you anticipate having an appropriate volume of changes to send.

Best Practice 2: It is also important to remove replicas that you no longer plan to synchronize from the geodatabase. If the replica in this example is never synchronized, the system version added at replica creation is never removed. You can see how over time, as more edits are

applied to default, this will prevent many edits from being removed during compress. Removing unused replicas removes unused system versions which can lead to a more effective compress.

The final step in the example is to remove the project1 version, as we had done in example 1, and then run the compress. Figure 9 shows the results where all changes are removed from the delta tables.

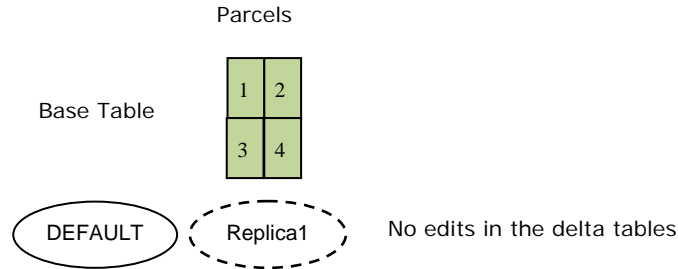


Figure 9: Edit1, Edit2 and Edit3 are removed from the delta tables by compress

Since this example describes a 1 way replica, we only showed the parent replica geodatabase. This is because no system versions are created in a child replica geodatabase for a 1 way replica. As a result, 1 way replicas have no effect on compress in the child replica geodatabase.

Example 2 is a simple example involving a single named version and a single 1 way replica. However, this can easily be expanded to include many named versions with many edits as well as several 1 way replicas. In this expanded scenario, a similar result can be achieved by doing the following:

- Reconcile and post each named version with DEFAULT and then manually delete each version
- Synchronize each replica with changes being sent and an acknowledgement received
- Compress the geodatabase

The end result is similar to figure 9, only instead of a single system version there would be one system version per replica.

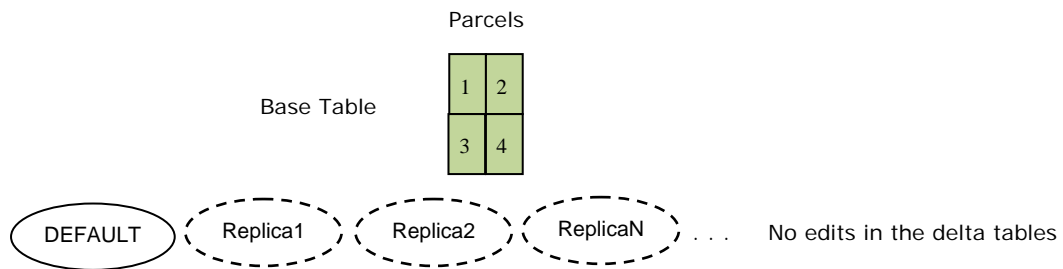


Figure 10: End result of Example 2 with multiple named versions / replicas

Another variation on this example would be where the project1 version is used as the replica version instead of DEFAULT. Here we would still first reconcile and post the project1 version with default and then synchronize the replica. This end result would again be similar except in this case the project1 version cannot be deleted since it is associated with a replica.

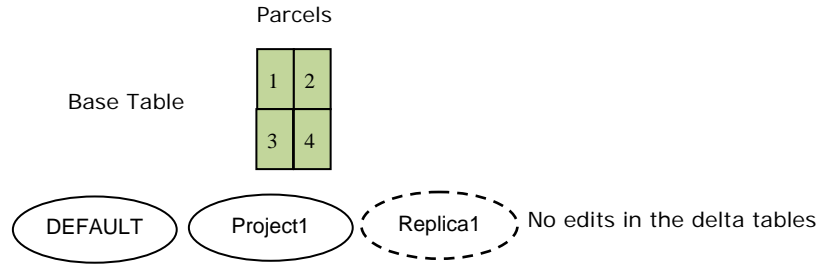


Figure 11: End result of Example 2 where project1 is the replica version

Example 3: Versioned editing and compress with a 2 way replica

This example starts off at the same point as the other examples except in this case a 2 way replica (Replica2) is created with the default version as the replica version. The replica creation process creates a system version from default for Replica2 in the parent geodatabase as well as the child geodatabase (Figure 12).

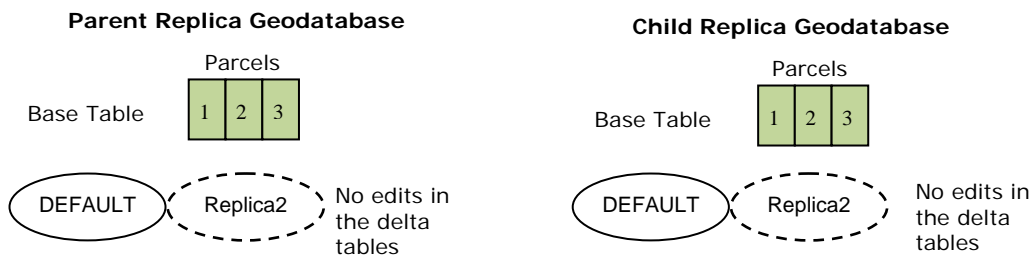


Figure 12: A system version is created in the parent and in the child geodatabase when a 2 way replica named Replica2 is created

Next, parcel 3 is moved in the default version of the parent replica geodatabase. Also, parcel 4 is added in the default version of the child replica geodatabase (figure 13).

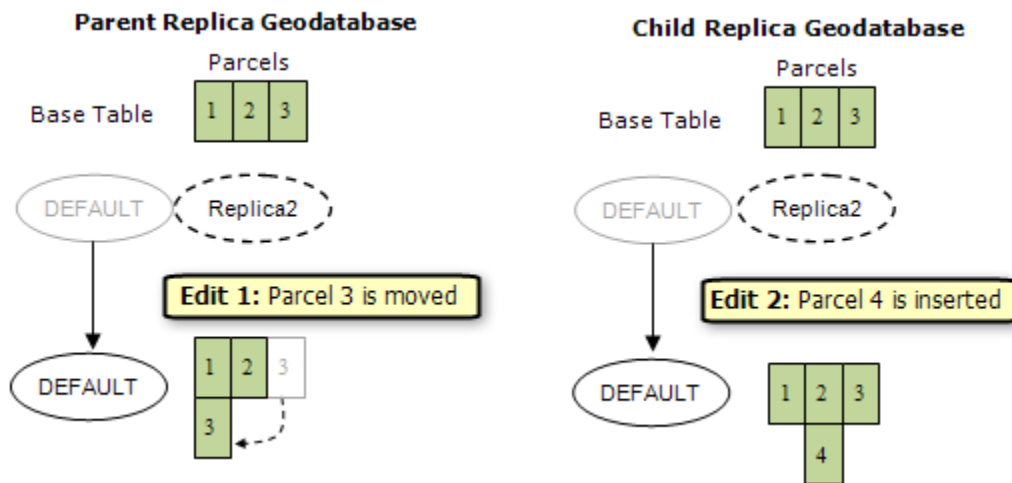


Figure 13: Parcel 3 is moved in the default version of the parent replica geodatabase (Edit 1) and parcel 4 is added in the default version of the child replica geodatabase (Edit 2).

Replica2 is then synchronized where changes are sent from the parent replica to the child replica (Figure 14). In this case, connected synchronization is used so the changes are therefore acknowledged implicitly. In this example there are no conflicts, however, had the same feature been redefined on both the parent and the child, there would have been a conflict. See the [synchronization and versioning](#) help topic for more information.

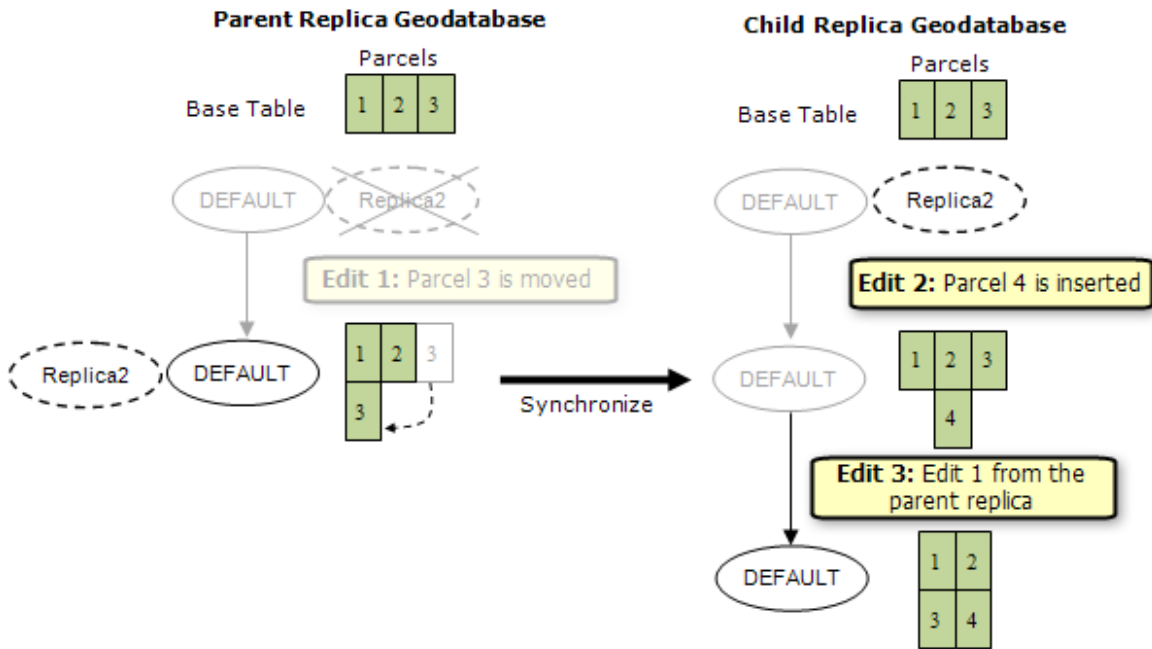


Figure 14: Changes are synchronized from parent to child. This applies Edit1 from the parent to the child as Edit 3.

In figure 14, since an acknowledgement was received by the parent, the initial system version is deleted and a new one created. With the old system version removed, a compress called at this point on the parent replica geodatabase would remove edit 1 from the delta tables and applied it to the base table. In the child replica geodatabase, edit1 from the parent is applied to the replica version (default) as a result of the synchronization and is labeled edit 3. Since the initial system version is left unchanged, a compress called on the child at this point will result in no changes being removed from the delta tables.

Next, Replica2 is synchronized again, this time changes are sent from the child replica to the parent replica (Figure 15). Once again, connected synchronization is used and the changes are acknowledged implicitly.

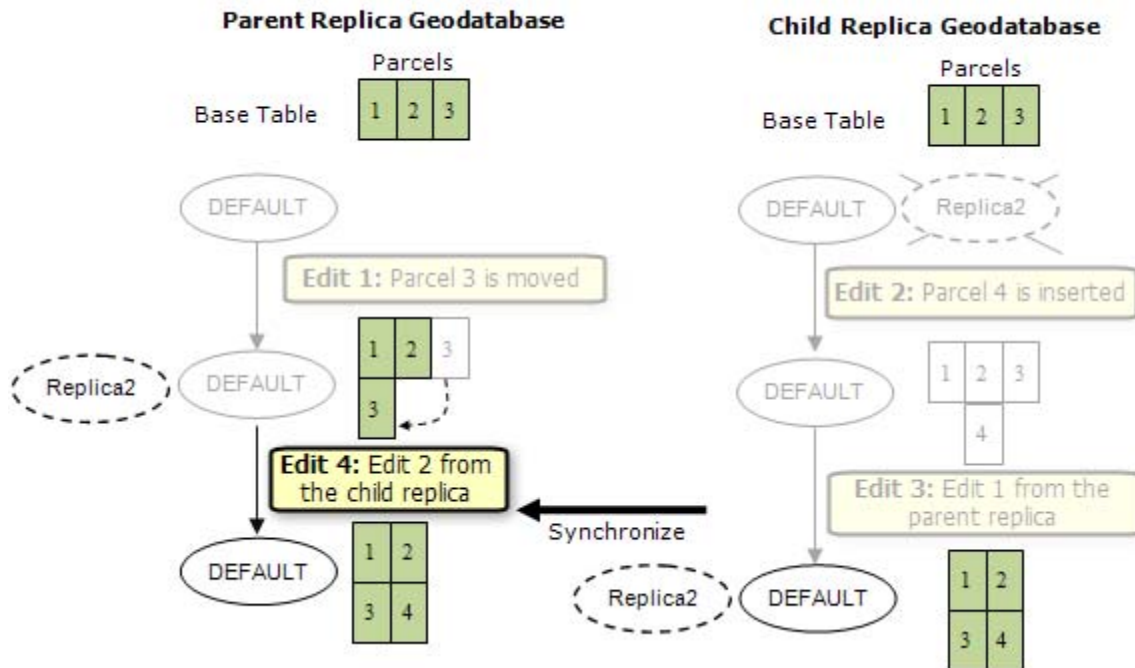


Figure 15: Changes are synchronized from child to parent. This applies Edit2 from the child to the parent as Edit 4.

In figure 15, edit 2 from the child replica is applied as edit 4 to the replica version (default) in the parent replica geodatabase. Edit 3 is not sent because the system identified it as a change that already exists in the child replica. In the child replica geodatabase, the existing system version is deleted and a new one created from default. This will allow a compress on the child geodatabase to apply edit2 and edit3 to the base table and remove them from the delta tables. A compress in the parent geodatabase, however, would still leave edit 4 in the delta tables.

Best Practice 3: In some cases, changes predominantly need to be sent in one direction but a 2 way replica is still used. For example, you may choose to do this if you see a potential future need to send changes back. In these situations, make sure to occasionally synchronize changes in the other direction even if there are little or no changes to send. This will remove system versions from the geodatabase that normally receives changes which allows compress to be more effective.

A third connected synchronization of replica2 is then run to send changes from the parent to the child (figure 16).

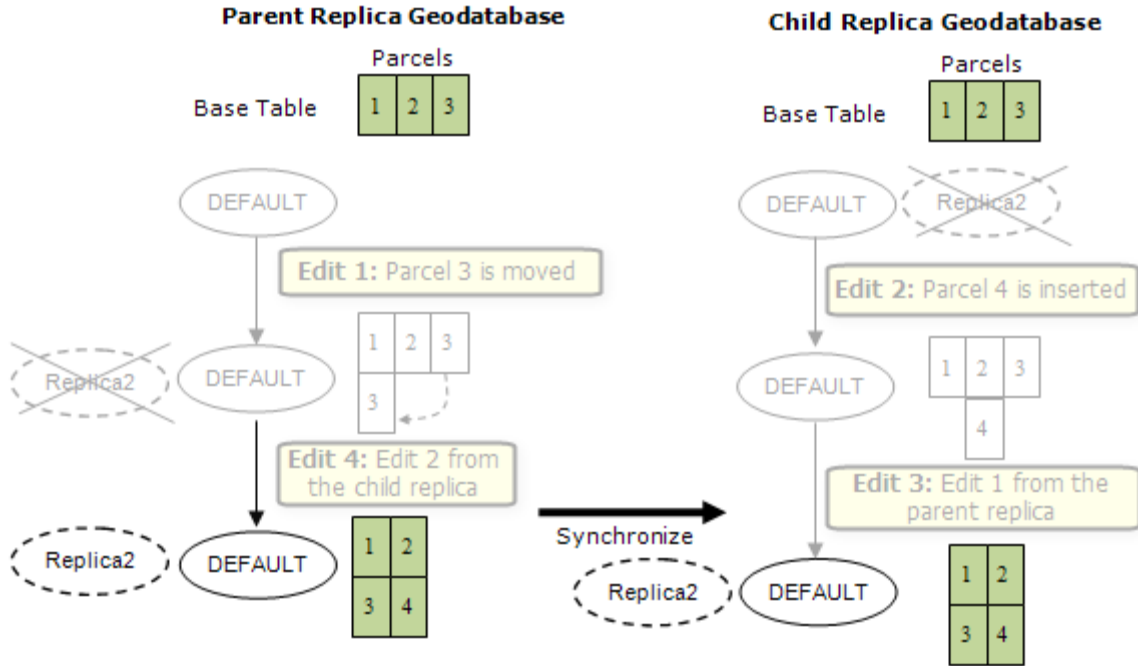


Figure 16: Changes synchronized from parent to child. No changes are sent during the synchronization.

In figure 16, edit 4 is identified as already existing in the child geodatabase and is therefore not sent. Since no other changes have been applied on the parent, no changes are sent to the child geodatabase. The acknowledgement received from the child allows the old system version to be dropped on the parent and a new one created. A compress run on the parent replica geodatabase at this point would move edit1 and edit4 to the base table and remove them from the delta tables.

A compress is then run on both replica geodatabases. This results in all edits being moved into the base tables and removed from the delta tables in both replica geodatabases (figure 17).

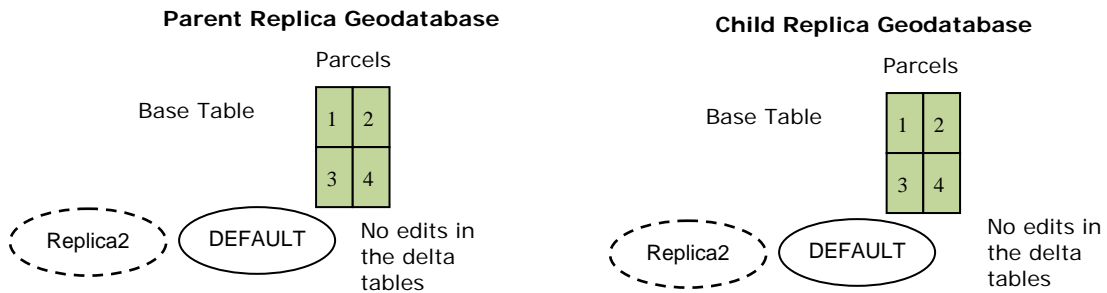


Figure 17: Edits are removed from the delta tables during compress in both replica geodatabases

For simplicity sake, this example shows edits applied only to the default version. In the earlier examples, changes were shown to be reconciled and posted into default from a named version and then the named version deleted. Had we followed the same pattern for this example, where named versions are reconciled and posted and the deleted before the synchronizations, the end result would be the same.

Note that as the synchronizations are performed, more system versions than what are shown in the example diagrams may be created. These are not shown in the diagrams because they are not needed to explain the results in the examples shown. You can think of the system version in the diagrams as one or more system versions. For more information on system versions, see appendix A.

Versioned editing and compress with check-out replicas

When a check-out replica is created, versions are created in the parent replica geodatabase as well as the child replica geodatabase if the child is an ArcSDE geodatabase. These versions remain until the check-out replica is synchronized (i.e. checked in). Synchronization removes the checkout replica as well as the system versions associated with it. A practice where you first check in these replicas and then perform the reconcile and post plus delete version pattern will yield the result shown in figure 5 (end result of Example 1).

Best Practice 4: These examples describe a pattern where changes from named versions are on occasion reconciled and posted with default and then optionally deleted. The geodatabase is then compressed. The following describes this type of workflow with ArcSDE geodatabases that have replicas:

- 1) First, synchronize (check in) your check-out replicas.
- 2) Reconcile and post and optionally delete the non-replica versions.
- 3) Reconcile and post the replica versions. As part of the synchronization process you may want to receive changes as well as send changes. In this case, you would ideally receive changes before reconciling and posting the replica versions.
- 4) Synchronize the replicas such that changes are sent and an acknowledgement received.
- 5) Run compress.

Note that when compress is complete, it is an ideal time to [update statistics and to rebuild indexes](#).

Full Compress

In the examples, the end result involves all edits being removed from the delta tables. This is known as a full compress. Although this is ideal, achieving a full compress is often not necessary and may not even be practical. For instance, in example 2, if many replicas are involved you will need to send changes for all replicas in order to achieve a full compress. However, you may never have a time in your business workflow where you are ready to send changes for all replicas. In example 3, we added a third synchronize that sent no changes in order to achieve a full compress. If you have many 2 way replicas, you may need to perform many more of these synchronizations with no changes to enable the geodatabase to fully compress. This again may not be practical.

Using the information in this article, you can develop a practice where you schedule compress around reconcile, post and synchronization. Another approach is to simply schedule a compress process to occur on a regular basis (i.e. nightly). As long as the the practices described in this article are applied, you should not see a case where changes that you expect to be removed during compress are not removed because of replication.

It is sometimes assumed that a full compress is needed because external, non-ESRI applications are used which can only access data in the base tables. In this case, look at creating [multi-version views](#) and accessing them from the non-ESRI applications instead of trying to compress to base.

Best Practice 5: In general, look to achieve some level of compress rather than a full compress. Attempting to achieve a full compress may lead to many, additional synchronizations which may not be practical in your particular business workflow.

Summary

This article describes how versioning is used by geodatabase replication. Aside from the compress process, it has little impact on existing versioning workflows. The information in this article can be used to achieve an effective compress with replicas in the geodatabase. The article also includes what the replica system versions are and how to find them (See Appendix A).

Throughout the article, some best practices are outlined. These are summarized below:

Best Practice 1: It is important to remove replicas that you no longer plan to synchronize from the geodatabase.

Best Practice 2: Plan to synchronize replicas where you send changes on a regular basis.

Best Practice 3: If you predominantly send changes in one direction with a 2 way replica, occasionally synchronize changes in the other direction even if there are little or no changes to send.

Best Practice 4: Do the following to incorporate replication into an existing pattern where changes from named versions are reconciled and posted with default and then optionally deleted:

- 1) First, synchronize (check in) your check-out replicas.
- 2) Reconcile and post plus delete the non-replica versions
- 3) Reconcile and post the replica versions. As part of the synchronization process you may want to receive changes as well as send changes. In this case, you would ideally receive changes before reconciling and posting the replica versions.
- 4) Synchronize the replicas such that changes are sent and an acknowledgement received.
- 5) Run compress.

Note that when compress is complete, it is an ideal time to [update statistics and to rebuild indexes](#).

Best Practice 5: In general, look to achieve an effective compress rather than a full compress. An effective compress can be achieved by following the best practices in this document and scheduling compress to happen a regular basis (i.e. nightly).

Appendix A: Replica System Versions

The examples in this article refer to replica system versions. These are versions that are used by ArcGIS to determine the changes to synchronize for a replica. A set of system versions exists for each replica in the geodatabase. Various system versions are created and deleted by ArcGIS as the replicas are synchronized.

These versions are hidden, which means that they are not displayed in ArcGIS and not returned by ArcObjects. If you query the SDE.VERSIONS table in the underlying DBMS, however, you will see these versions listed. You can identify them by the way that they are named. The naming conventions for the replica system versions are as follows:

2 way and 1 way replicas

SYNC_RECEIVE_<replica id>_<generation number>
SYNC_RECEIVE_REC_<replica id>_<generation number>
SYNC_SEND_<replica id>_<generation number>

Check-out replicas

REP_CO_SYNC_<replica id>

For example, an Oracle ArcSDE geodatabase has been built with a 2 way replica named rep_2way, a 1 way replica called rep_1way and a parent check-out replica named rep_co. After connecting to Oracle using SQL plus, the following query is executed to list these replicas along with their replica ids:

```
SQL> select id, name from sde.gdb_replicas;
```

```
      ID NAME
-----
      57 rep_1way
      56 rep_2way
      58 rep_co
```

Next the sde.versions table is queried as follows:

```
SQL> select name from sde.versions order by name;
```

```
NAME
-----
DEFAULT
SYNC_RECEIVE_56_0
SYNC_RECEIVE_56_1
SYNC_RECEIVE_56_2
SYNC_RECEIVE_REC_56_2
SYNC_SEND_56_0
SYNC_SEND_57_0
rep_co
```

In this geodatabase you can see that the only non-system versions are default and rep_co. The rep_co version was created when the check-out replica was created and is the replica version. Applying the naming pattern information above, you can see that there are several system versions for rep_2way (replica id 56) and one system version for rep_1way (replica id 57). The following query returns only the system versions for rep_2way.

```
SQL> select name from sde.versions where name like 'SYNC_SEND_56%' or
name like
'SYNC_RECEIVE_%56%';
```

NAME

```
-----
SYNC_RECEIVE_56_0
SYNC_RECEIVE_56_1
SYNC_RECEIVE_56_2
SYNC_RECEIVE_REC_56_2
SYNC_SEND_56_0
```

In this example, the child check-out replica is hosted in another Oracle ArcSDE geodatabase. When SDE.GDB_REPLICAS table in that geodatabase is queried, the following is returned:

```
SQL> select id, name from sde.gdb_replicas;
```

ID NAME

```
-----
42 rep_co
```

Next the sde.versions table is queried in the same geodatabase as follows:

```
SQL> select name from sde.versions order by name;
```

NAME

```
-----
DEFAULT
REP_CO_SYNC_42
rep_co
```

The results of the query show the default version and the rep_co version as the only non-system versions. The rep_co version is the replica version and was created as part of the check-out process. Also listed is a version named REP_CO_SYNC_42 which we know from the naming pattern above is a system version for the rep_co check-out replica.

Using this information, you can determine which system versions exist in your geodatabase and which replicas they are associated with. Note that the system versions listed will change over time as the replicas are synchronized. It is very important, however, to never interact with these system versions manually. If you need to remove these versions, you should either follow the examples in this article or delete the replica in ArcGIS.