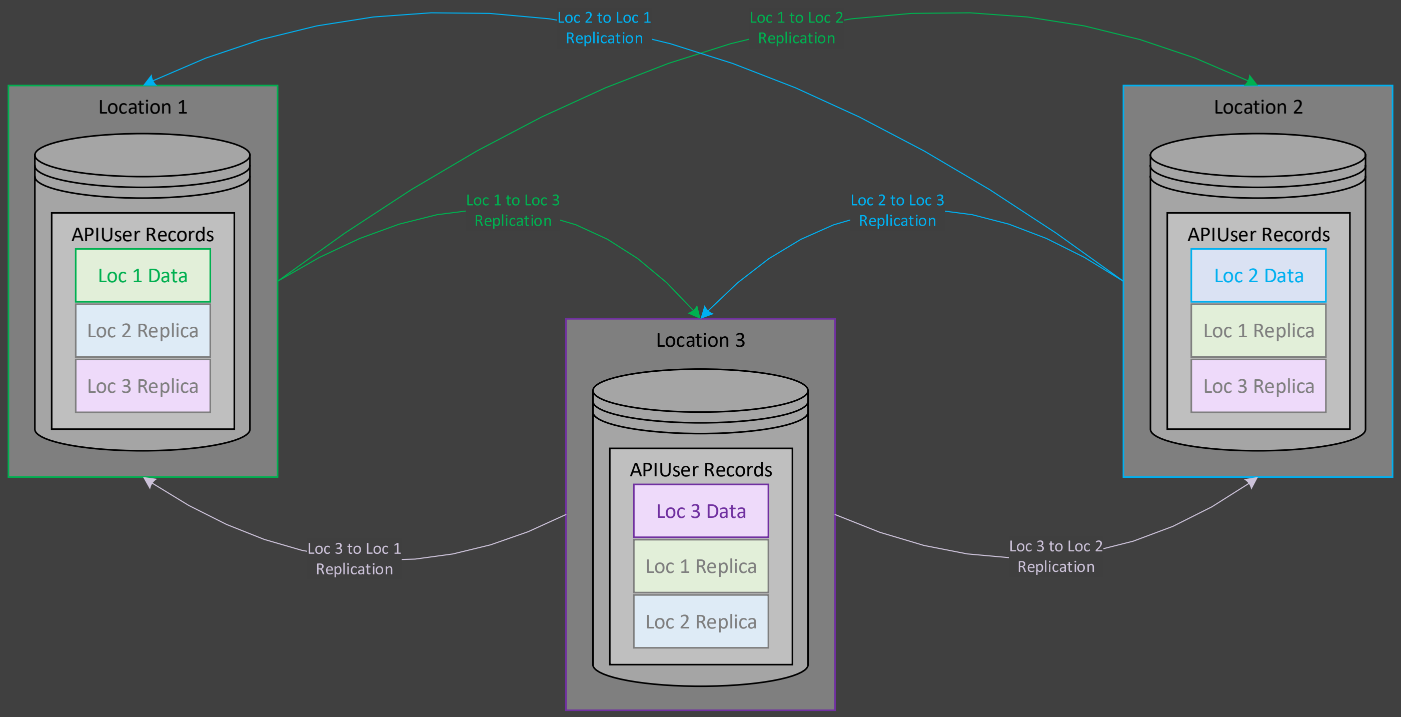
DGP Merge Replication of Data



When using the DGP Replica or Hybrid schema types, 1) each database is duplicated in multiple geographically distributed locations, and 2) each table in the database is treated as an append-only collection of immutable records. The asynchronous replication process runs constantly in the background to quickly merge the records created at each location into the corresponding tables of the other locations, creating an eventually consistent union of the records from all the locations (similar in concept to CRDT's). Replication from each source to each destination is configured and managed independently.

Each location functions autonomously, and does not attempt to verify the consistency of the data across other locations prior to writing new immutable records into its own local database. Instead, merge conflicts are corrected as they are detected during the replication and verification processes, and the state value of each record is adjusted accordingly.

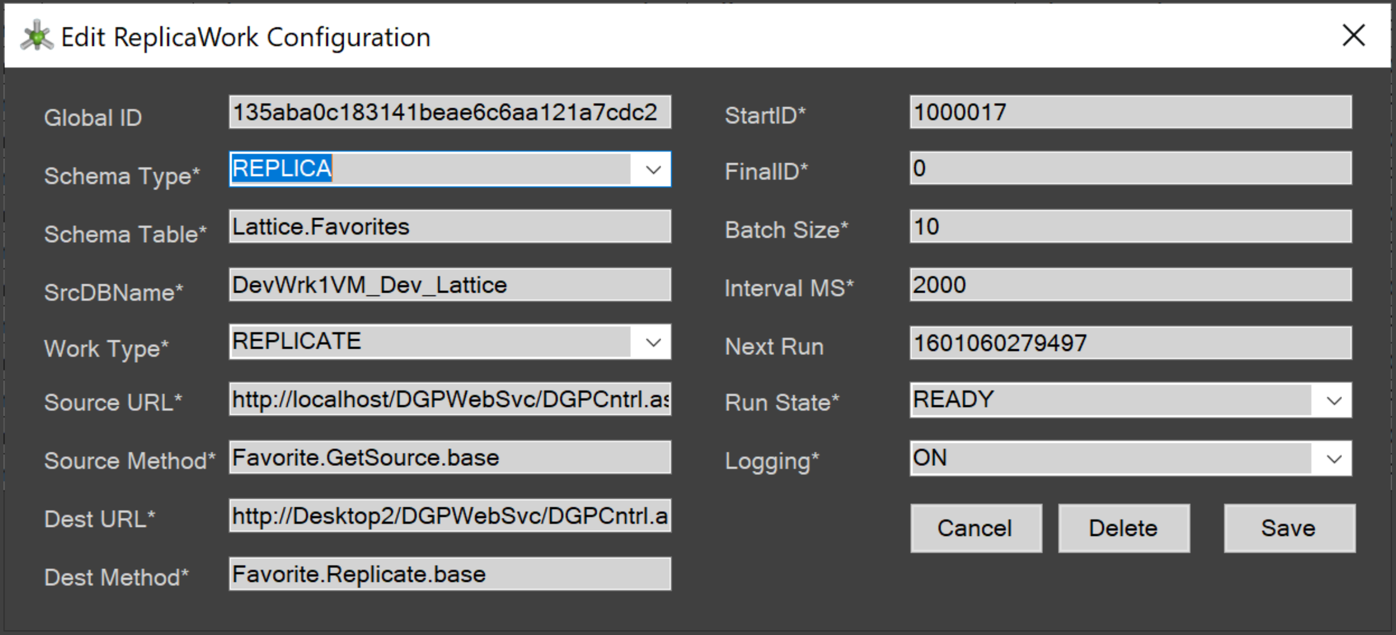
Configuring Merge Replication

To create configuration records for replication, some information from various sources needs to be collected.

- The schemaname.tablename combination
- The name of the source database, which is used for several of the queries in the process (the name of the destination database is not needed, and is determined by the destination URL and destination method name used).
- The URL of the web service used to get the source records
- The name of the method used to poll for records in the source table
- The URL of the web service used to merge the destination records
- The name of the method used to merge source records into the destination table

When creating new configuration records, the schemaname.tablename combinations and the names of the API methods for replication, count checks, etc. are documented in the API section under DGPWork/SchemaTables. The URL's of the source and destination web services are determined by the network topology of the various locations of an environment for a given system.

*Important Note: the URL's used in the configuration records must be able to be resolved from wherever the DGPWinSvc will be run.


Edit ReplicaWork Configuration

Global ID	135aba0c183141beae6c6aa121a7cdc2	StartID*	1000017
Schema Type*	REPLICA	FinalID*	0
Schema Table*	Lattice.Favorites	Batch Size*	10
SrcDBName*	DevWrk1VM_Dev_Lattice	Interval MS*	2000
Work Type*	REPLICATE	Next Run	1601060279497
Source URL*	http://localhost/DGPWebSvc/DGPCntrl.as	Run State*	READY
Source Method*	Favorite.GetSource.base	Logging*	ON
Dest URL*	http://Desktop2/DGPWebSvc/DGPCntrl.a		
Dest Method*	Favorite.Replicate.base		

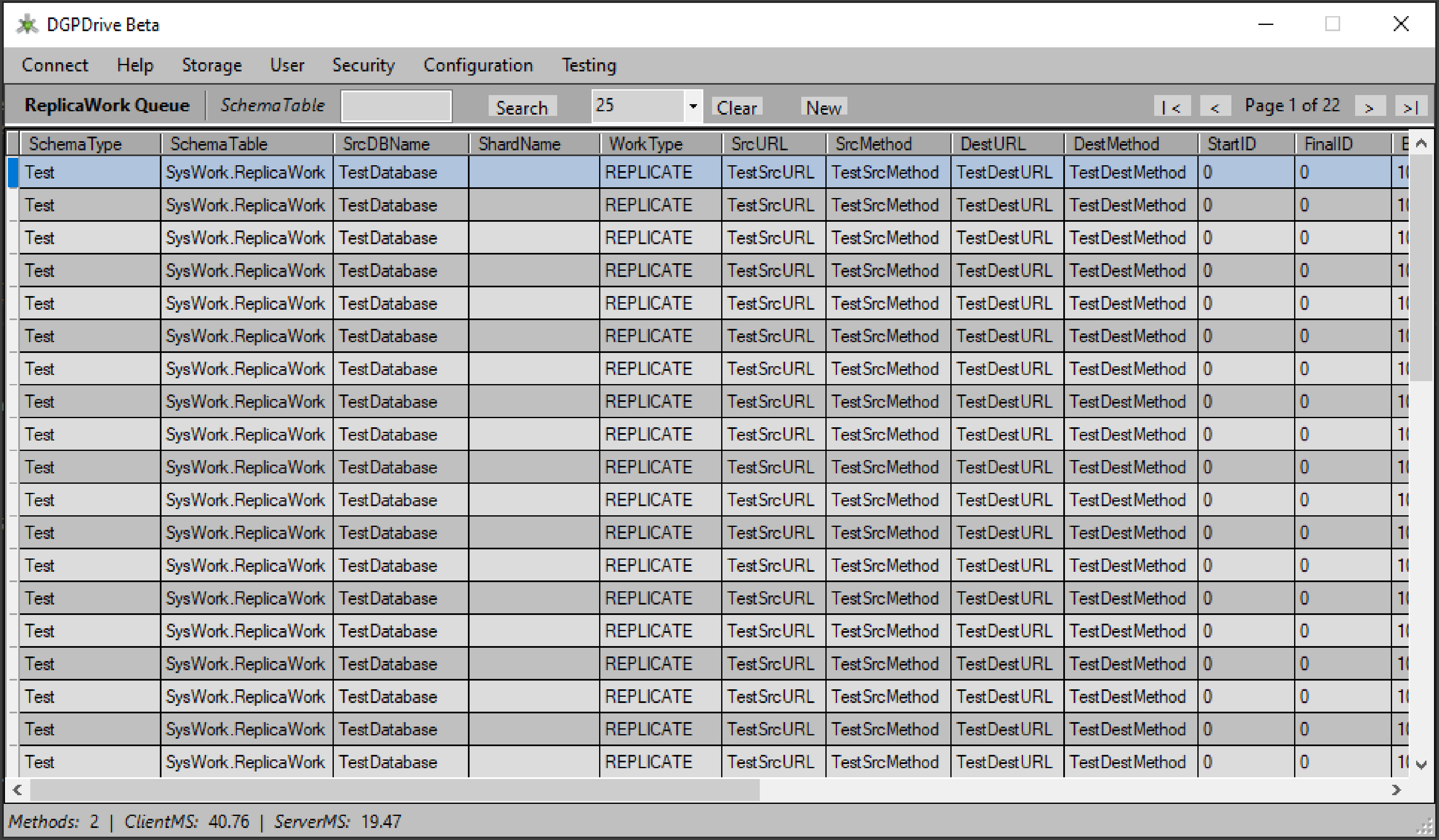
Field Name	Field Values	Description
SchemaType	REPLICA, HYBFILE, HYBREC	• REPLICA is the basic DGP schema that supports merge replication, and is the default value. • HYBFILE is a hybrid of a database table and a collection of shard file servers to store unstructured content (Lattice Filestore). • HYBREC is a hybrid of a database table and a collection of shard database servers to store large structured content (not used).
WorkType	REPLICATE, VERIFY used in ReplicaWork queue	• REPLICATE is the standard merge replication process • VERIFY is merge replication rolled forward from the StartID to fix gaps of missing records (stops itself when it catches up to the main replication process)

StartID	64-bit integer	This value is used as the starting value of a process placeholder. It will generally be set to zero for most processes. However, large sets of records can be segmented using the StartID and FinalID fields.
FinalID	64-bit integer	This value sets the upper limit of a process placeholder for some processes. It will generally be set to zero, and is ignored in most cases, but when paired with the StartID, it can be used to break up very large numbers of records into smaller segments.

Some of the other values include BatchSize, which is used for iterative processes such as merge replication, but not for the detection processes. The IntervalMS sets the delay between iterations of the process when it has “caught up” with the workload – whenever there is a backlog of work to be done, the interval is ignored and the process is scheduled to run immediately. The Next Run value is the Unix Time in milliseconds that the process is scheduled to be run – initially it can be set to zero. Setting the Run State value to READY is equivalent to turning a process on to begin execution.

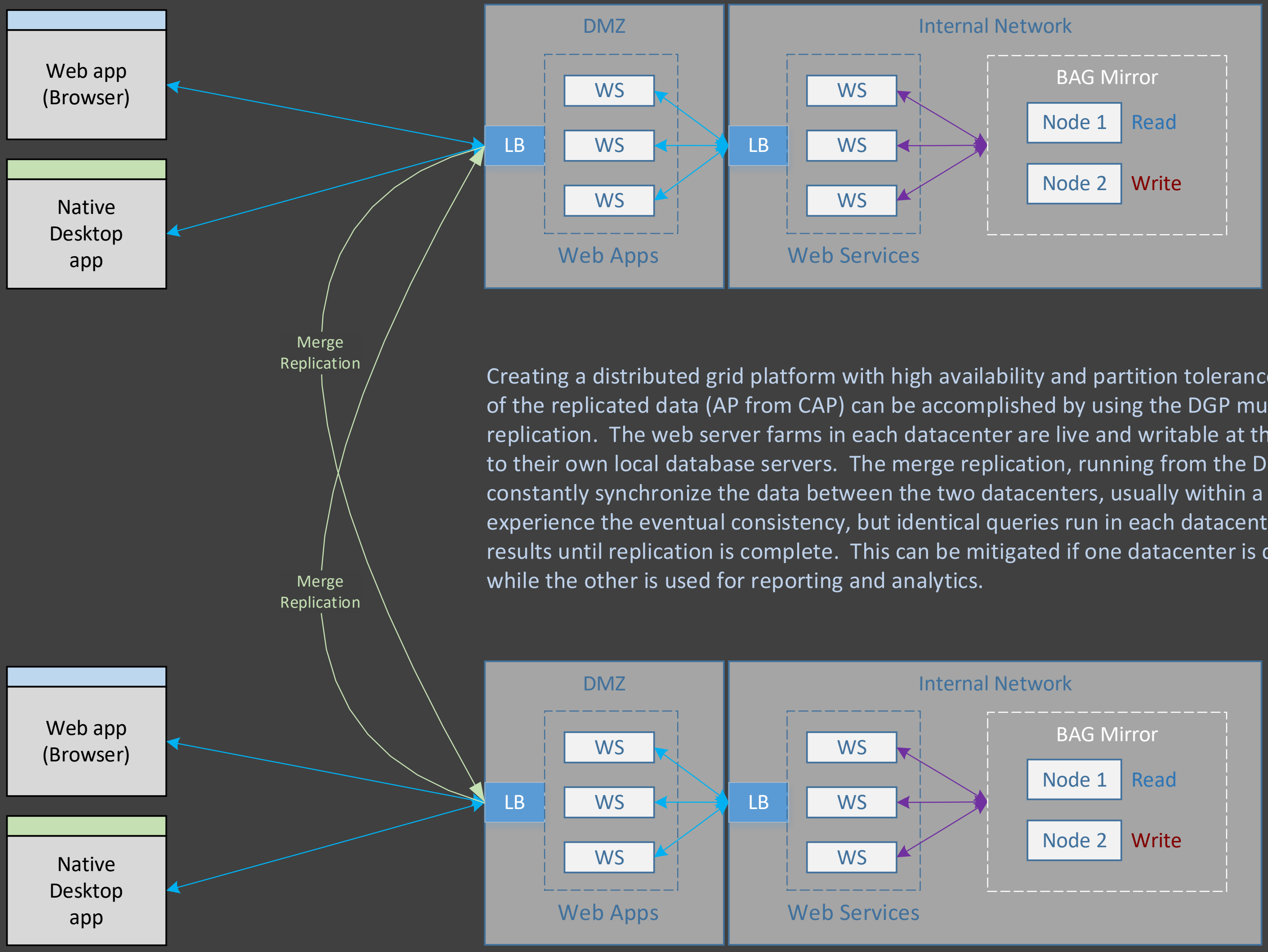
Replicate vs. Verify Mode

Verify processes fix gaps in destination data by rerunning the replication process alongside the main REPLICATE process. There is no need to create Verify configuration records until they are needed. They are created with the “Clone” option of the context menu in the data grid, and then switching the work type to “Verify”. Start and Final placeholder values should by default be set to zero. The IntervalMS value is not used, so setting it to a default value of 1000 or 2000 is fine. Verify replication processes will automatically stop themselves when they catch up to the placeholder of the main REPLICATE process.



SchemaType	SchemaTable	SrcDBName	ShardName	WorkType	SrcURL	SrcMethod	DestURL	DestMethod	StartID	FinalID	E
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1
Test	SysWork.ReplicaWork	TestDatabase		REPLICATE	TestSrcURL	TestSrcMethod	TestDestURL	TestDestMethod	0	0	1

```
Methods: 2 | ClientMS: 40.76 | ServerMS: 19.47
```



Creating a distributed grid platform with high availability and partition tolerance with eventual consistency of the replicated data (AP from CAP) can be accomplished by using the DGP multi-master merge replication. The web server farms in each datacenter are live and writable at the same time, each writing to their own local database servers. The merge replication, running from the DMZ for security, works to constantly synchronize the data between the two datacenters, usually within a few seconds. Users do not experience the eventual consistency, but identical queries run in each datacenter could return different results until replication is complete. This can be mitigated if one datacenter is designated as a primary, while the other is used for reporting and analytics.

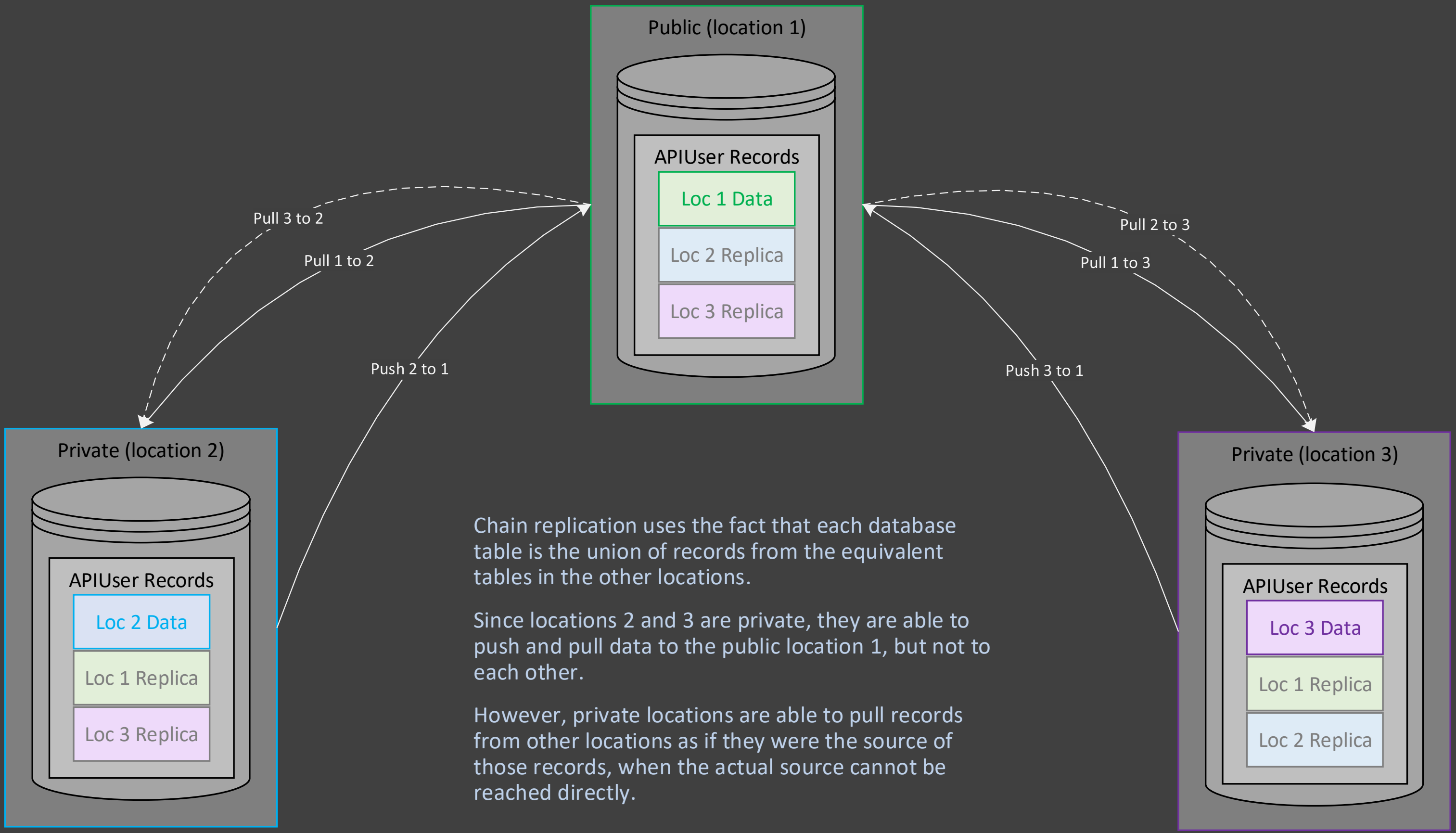
Chain Replication

Hierarchical chain replication makes use of the fact that each replica table is a union of all the records from all the locations to use source records replicated into a destination table as a source for those records when the actual source is not accessible due to the network topology, security, etc. It is therefore polling a destination table for records that have been replicated to it from an original source (which itself is not accessible).

The configuration of chain replication is the same as regular replication, but will generally be a "pull" from a remote destination table acting as a source table. The only difference is the URL of the web service used to poll for source records, and the fact that those records will not be available until the replication from their original source has been completed. Chaining destination tables together as source tables can be as many layers deep as necessary to work within a network topology.

The last 2 diagrams below show examples of this type of chain replication.

DGP Hierarchical “Chain” Merge Replication of Data

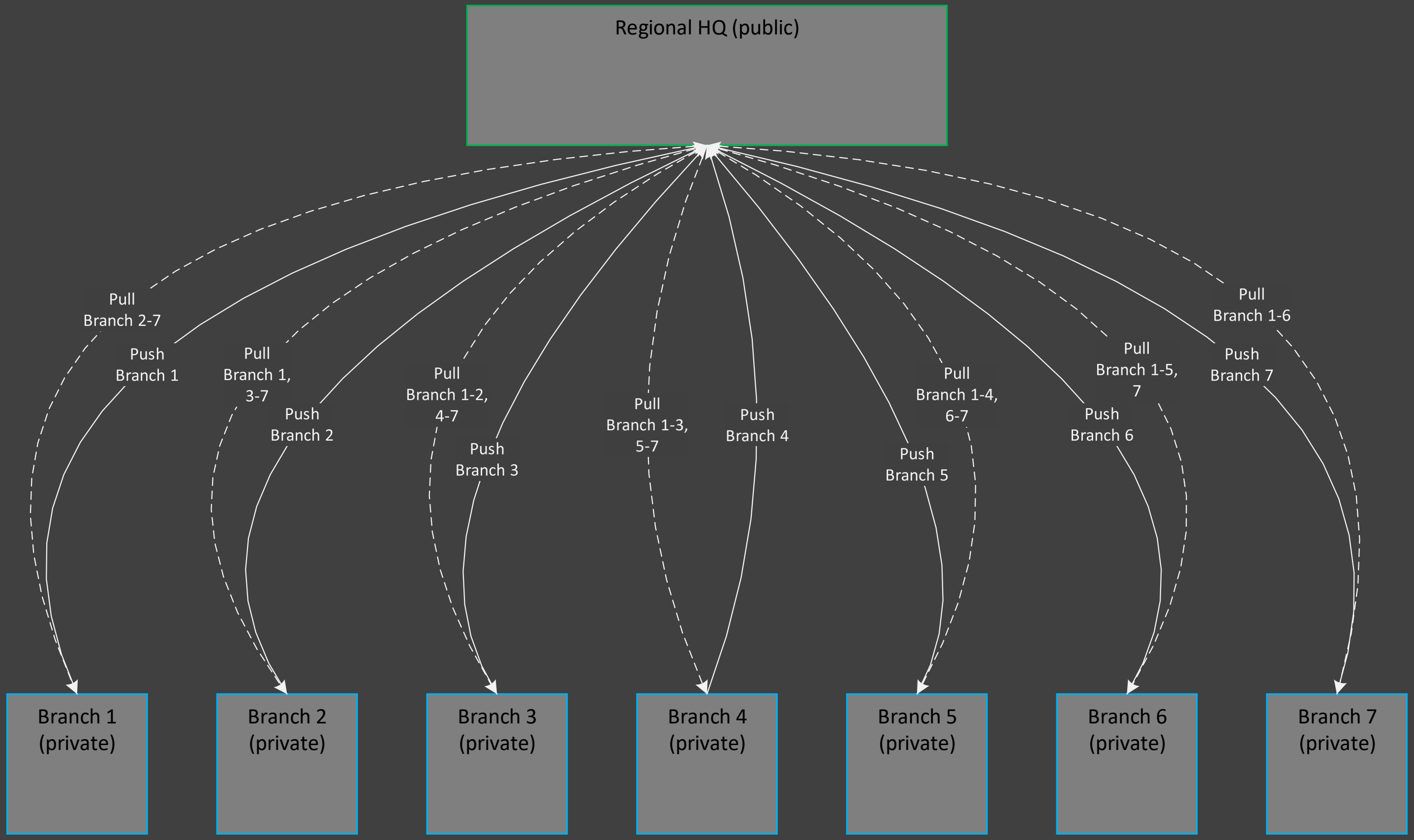


Chain replication uses the fact that each database table is the union of records from the equivalent tables in the other locations.

Since locations 2 and 3 are private, they are able to push and pull data to the public location 1, but not to each other.

However, private locations are able to pull records from other locations as if they were the source of those records, when the actual source cannot be reached directly.

DGP Hierarchical “Chain” Merge Replication of Data: Example



In this example, the wide-area network links each branch to the Regional HQ, but there is no connectivity among the branches to each other. Merge replication is configured to use the Regional HQ as a clearinghouse for the records of all the branches, so their records can be pulled to each of the other branches without any direct connectivity. The same technique can be used to create a hierarchical chain of replication between the Regional HQ’s and the central Corporate HQ, and so on.