Database Storage

The data storage tier consists of the SQL Server relational database engine for both the main database schemas and the shard schemas. DGP has several custom types of database schemas:

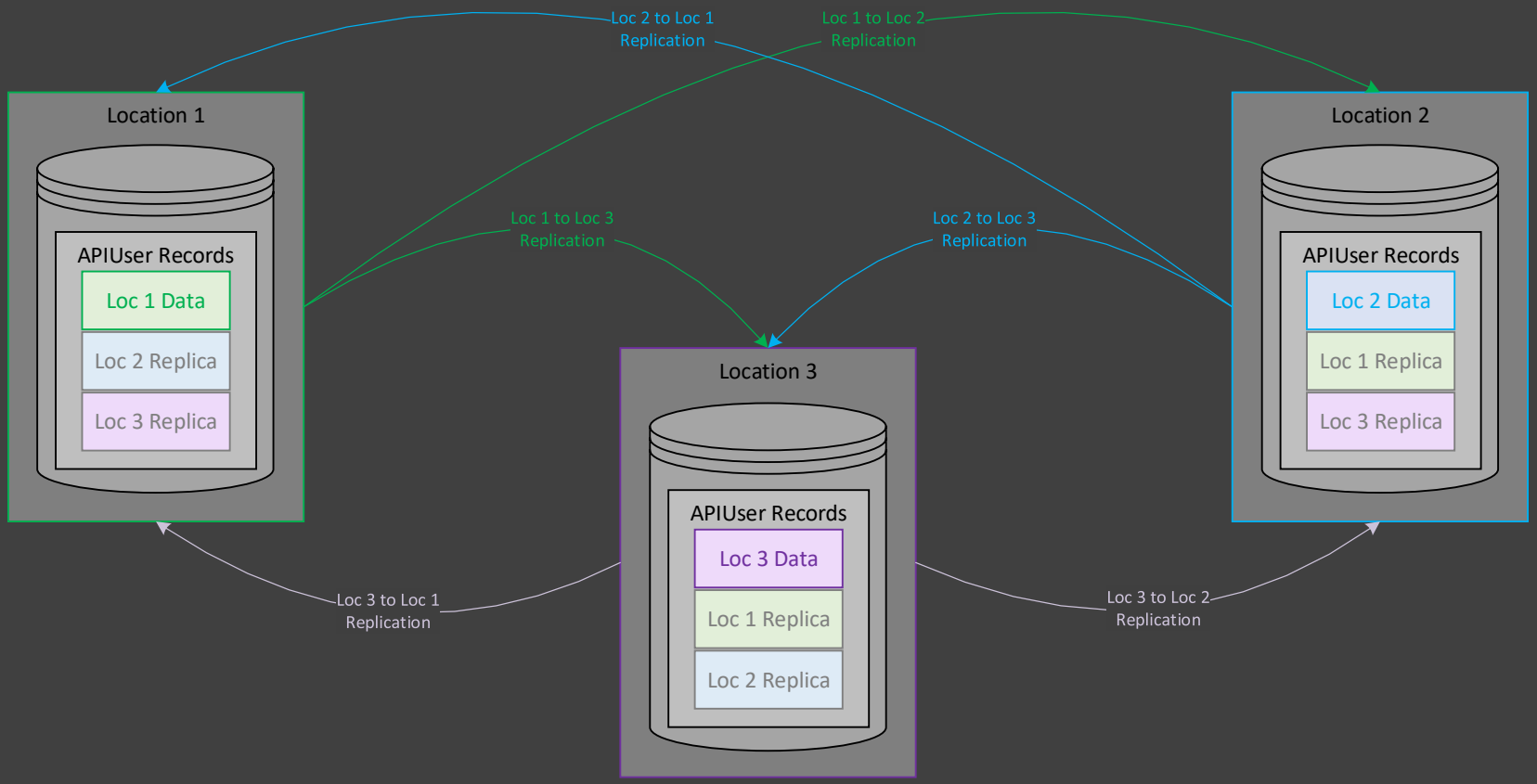
- Standard
- Replica
- FileShard

Standard databases are the normal, commonly used SQL Server databases. Replica databases, in contrast, treat each table as an append-only collection of immutable records, similar to CRDT's in terms of functionality. FileShard databases modify the replica type using a simple shard mechanism to scale out the fact tables of star and snowflake schemas. For example, the FileStore application uses the FileShard type of database shard schema to store the segments of its file content.

Multi-master Merge Replication

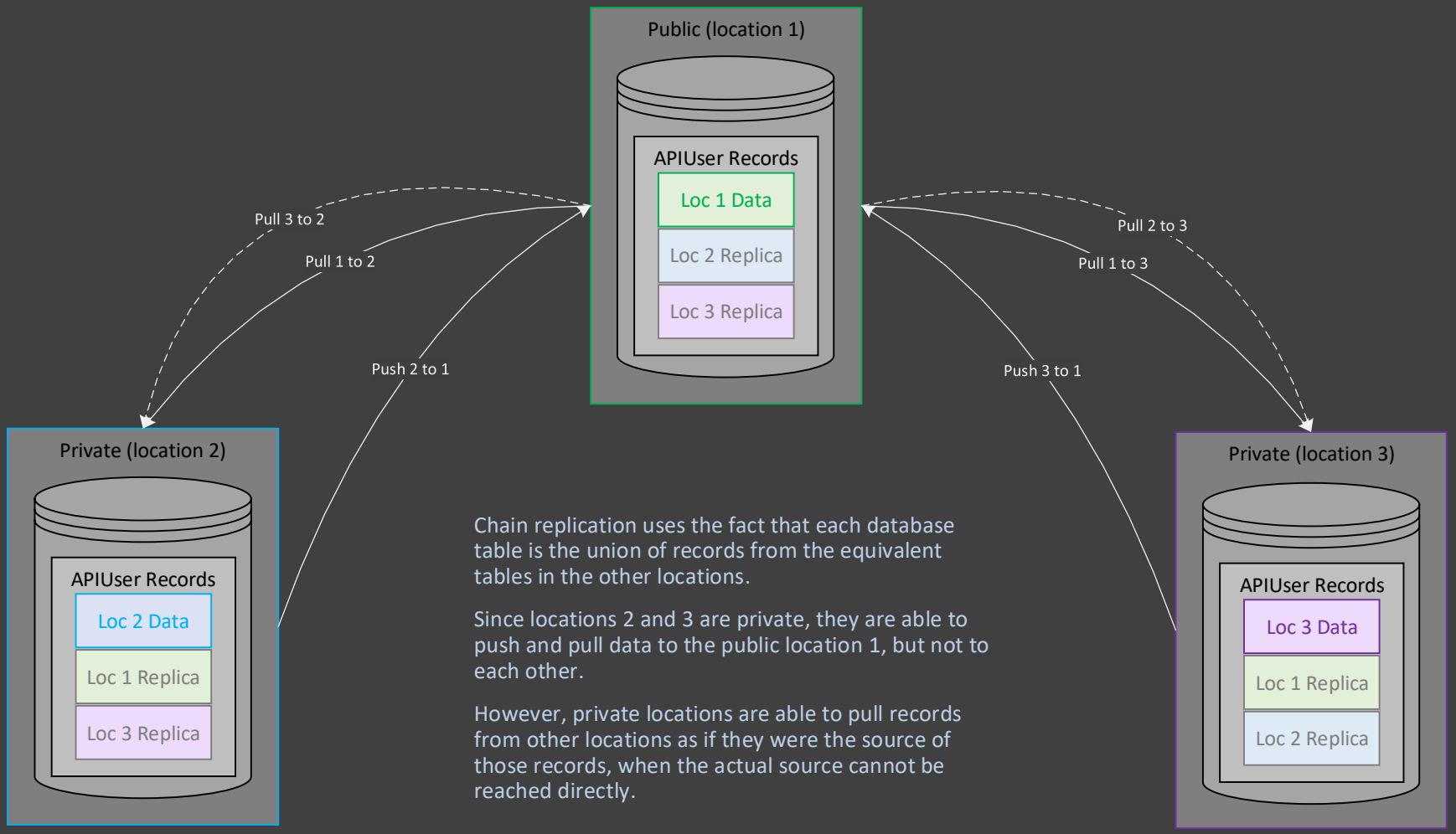
Given that DGP replica and hybrid databases treat tables as append-only collections of immutable records, replication becomes a process to create eventually consistent unions by merging the immutable records created at other locations into each table. Replica databases merge the records from each location into each of the other locations, while hybrid databases add an extra step to also replicate the content of the shards as well. The diagram below shows the merge replication process for replica databases.

The AutoWork subsystem executes the replication, data check and data repair processes continuously in the background to synchronize the data between locations, typically within a few seconds. The processes themselves consist of two parts: an "at least once" polling process to guarantee the execution of each merge iteration, combined with an "only once" idempotent process to insure that each record is replicated only once. Combined, they guarantee that each record is replicated "exactly once".



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.



Each leg of the merge replication is configured and executed independently, which allows the merge replication to be configured into hierarchical “chains”. Also, all replication is implemented as API methods for easy configuration of both “push” and “pull” merges.

DGP Database Shards

The Lattice FileStore app is an example of the use of DGP database shards. The shard mechanism only works well for star and snowflake schemas that store the bulk of their data in their main fact tables. DGP database shards first vertically partitions the schema into a main schema and a shard (fact table) schema. Then, the shard data is horizontally partitioned amongst one or more shard servers.

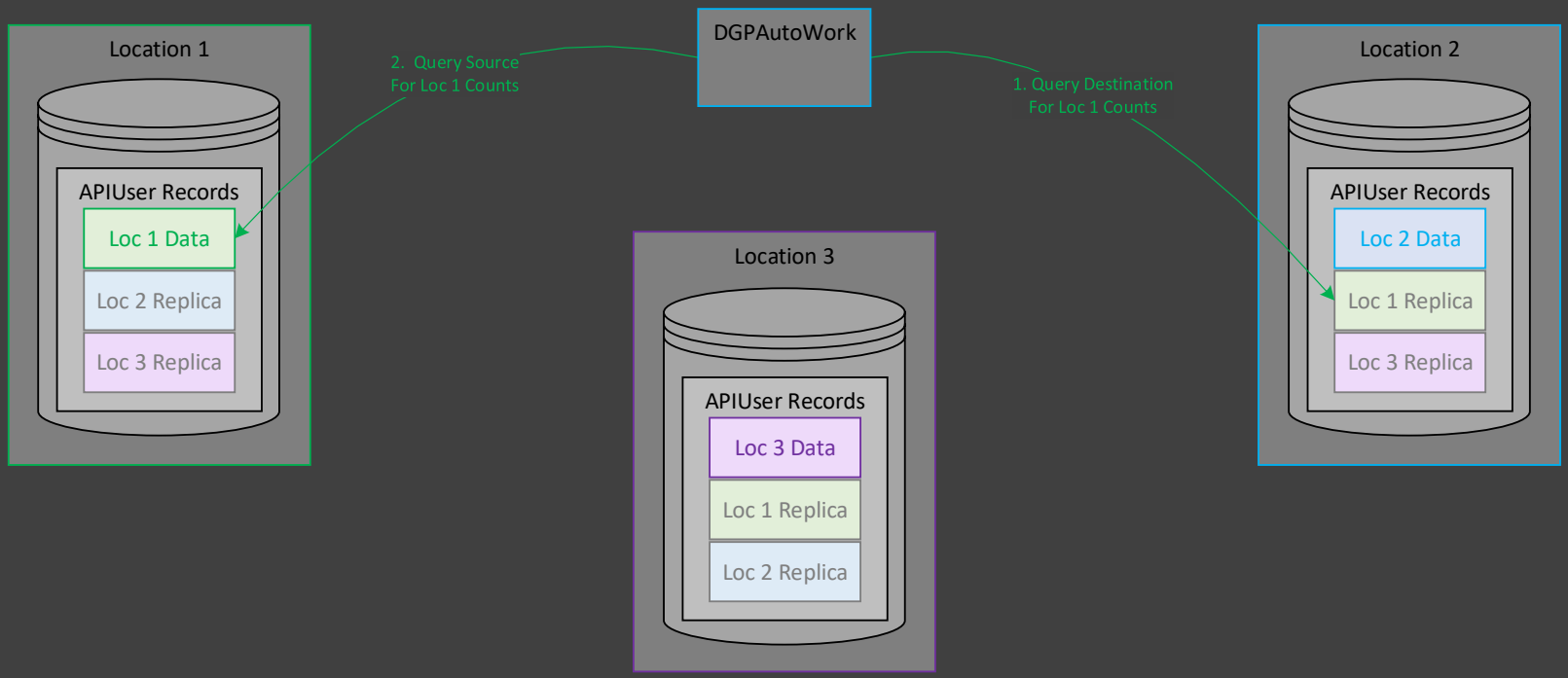
All of the ADO.NET database connection strings are managed in the Web.config file of the web services.

```
<!-- ADO.NET connection strings -->
<add key="SysInfo" value="Server=localhost;Database=DGPSysInfo_Win10Dev;User ID=UserName;Password=Password" />
<add key="SysWork" value="Server=localhost;Database=DGPSysWork_Win10Dev;User ID=UserName;Password=Password" />
<add key="SysMetrics" value="Server=localhost;Database=DGPSysMetrics_Win10Dev;User ID=UserName;Password=Password" />
<add key="TestDB1" value="Server=localhost;Database=DGPTestDB1_Win10Dev;User ID=UserName;Password=Password" />
<add key="TestDB2" value="Server=localhost;Database=DGPTestDB2_Win10Dev;User ID=UserName;Password=Password" />
<add key="FileStore" value="Server=localhost;Database=DGPFileStore_Win10Dev;User ID=UserName;Password=Password" />
<!-- Shard Names -->
<add key="FileShard1" value="Server=localhost;Database=DGPFileShard1_Win10Dev;User ID=UserName;Password=Password" />
<add key="FileShard2" value="Server=localhost;Database=DGPFileShard2_Win10Dev;User ID=UserName;Password=Password" />
<!-- Writable Shards (comma delimited list of writable shard names) -->
<add key="WritableShards" value="FileShard1,FileShard2" />
```

The list of ADO.NET connection strings is used for all of the database schemas for both reads and writes. The list of Shard Name connection strings are used to read and write records using a specific shard (the Shard Name is stored in each FileStore record and FileShard record for this purpose). The WritableShards key contains a delimited list of one or more shard names that are to be used to store new records in the collection of shards. Most systems will only need one writable shard at a time, but if that constraint becomes a performance bottleneck when inserting new records, then the write workload can be shared amongst multiple shards. In that case, a utility method randomly selects one of shard names in the comma delimited list of writable shards to be used.

The maximum size of a Web.config file is 250K by default. This limit can be increased if necessary: https://docs.microsoft.com/en-ca/archive/blogs/httpcontext/cannot-read-configuration-file-because-it-exceeds-the-maximum-file-size-web-config

DGP Data Repair



The count checks are the detection part of the in-place data repair processes. At the time the detection process is run, in the example above it starts by querying for the maximum src_id value for location 1 records in the location 2 database table. It then queries for the total count of records that are less or equal to than the maximum ID value, and the total count of active records.

It then runs all of these same count queries at the source, but uses the maximum src_id from the destination (not from the source) for the total rows and total active row counts. The difference between the maximum destination src_id and the maximum row_id of the source indicates the size of the replication lag. The row count values should be the same at the location and the destination. A lower count at the destination indicates missing replicated records, which is fixed by running a Verify scan. A higher count at the destination indicates some sort of duplication that must be investigated by admin staff.

Security

SQL Server accounts are used in the ADO.NET connection strings to work with the SQL Server databases. Those connection strings are stored in an encrypted section of the web service web.config files for QA and Prod environments using the aspnet_regiis.exe utility. Domain accounts can also be used for database access, but in general security is a bit stronger when using specialized accounts dedicated to only allow access to the various databases.

Security within the data access classes is maintained by enforcing the rule that all SQL logic is strictly parameterized, with no exceptions. In this way, the SQL syntax itself is immutable and cannot be altered by any user inputs (which makes the data access logic immune to injection attacks). All user inputs are assigned as values for a specific SqlCommand parameter.

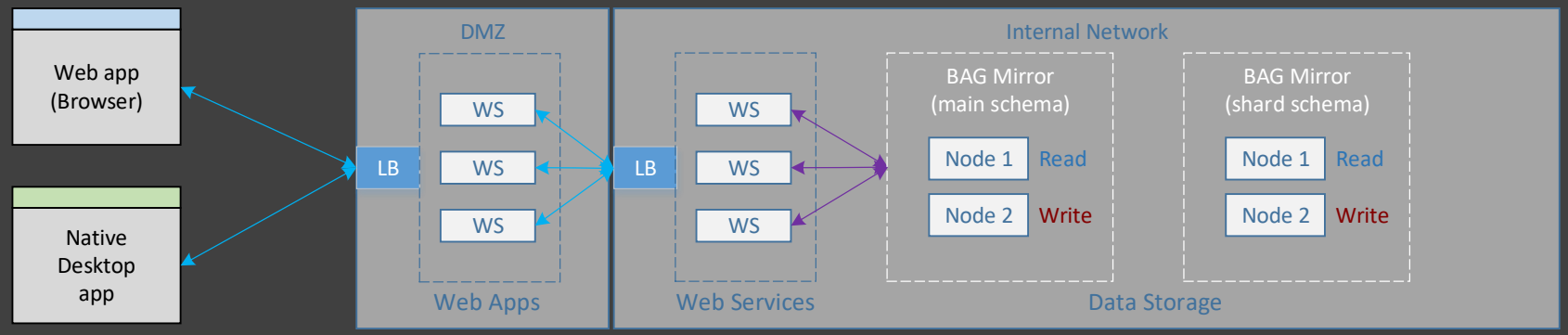
File server security is enforced by the web services impersonating system accounts to authorize access to the network shares. The system account credentials are also stored in the web.config files of the web services.

Availability, Scalability and Performance

The performance of data storage is greatly improved by using many TB of fast solid-state storage in each server. This type of storage should provide a minimum of 500,000 or more IOPS, which helps to guarantee that storage IO will not become a performance bottleneck as the system grows. This level of IOPS applies both to database storage and unstructured (file) storage.

In addition, the performance of the data access logic is tested, measured and improved using SSMS to analyze the query plans of the SQL syntax, and insure that there are enough indexes on each table to prevent the occurrence of full table scans. This becomes more and more important as the number of records in each table increases over time.

The diagram below shows the scalability of a location for the HYBFILE DGP database schema used by the optional Lattice FileStore application. Mechanisms to shard data will generally require both vertical and horizontal partitioning of the system data. The file storage content is scaled out horizontally as file server shards, while the separate database schema represents the vertical partitioning necessary to support the shard mechanisms.



LB = Load Balancer | WS = Web Server | FS = File Server

Scaling the 4-tier architecture within a location is a matter of first separating each tier onto its own hardware, from which point each tier is able to scale independently of the others. The web server farms handle both N + 1 fault tolerance and incremental horizontal scalability for both the DMZ and internal network. The SQL Server Availability Groups provide redundancy for the various pairs of database servers. The main schema servers can only scale vertically. The shard schema servers provide horizontal scalability for shard data by incrementally adding more shard servers as needed.