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 uses several types of database schemas:

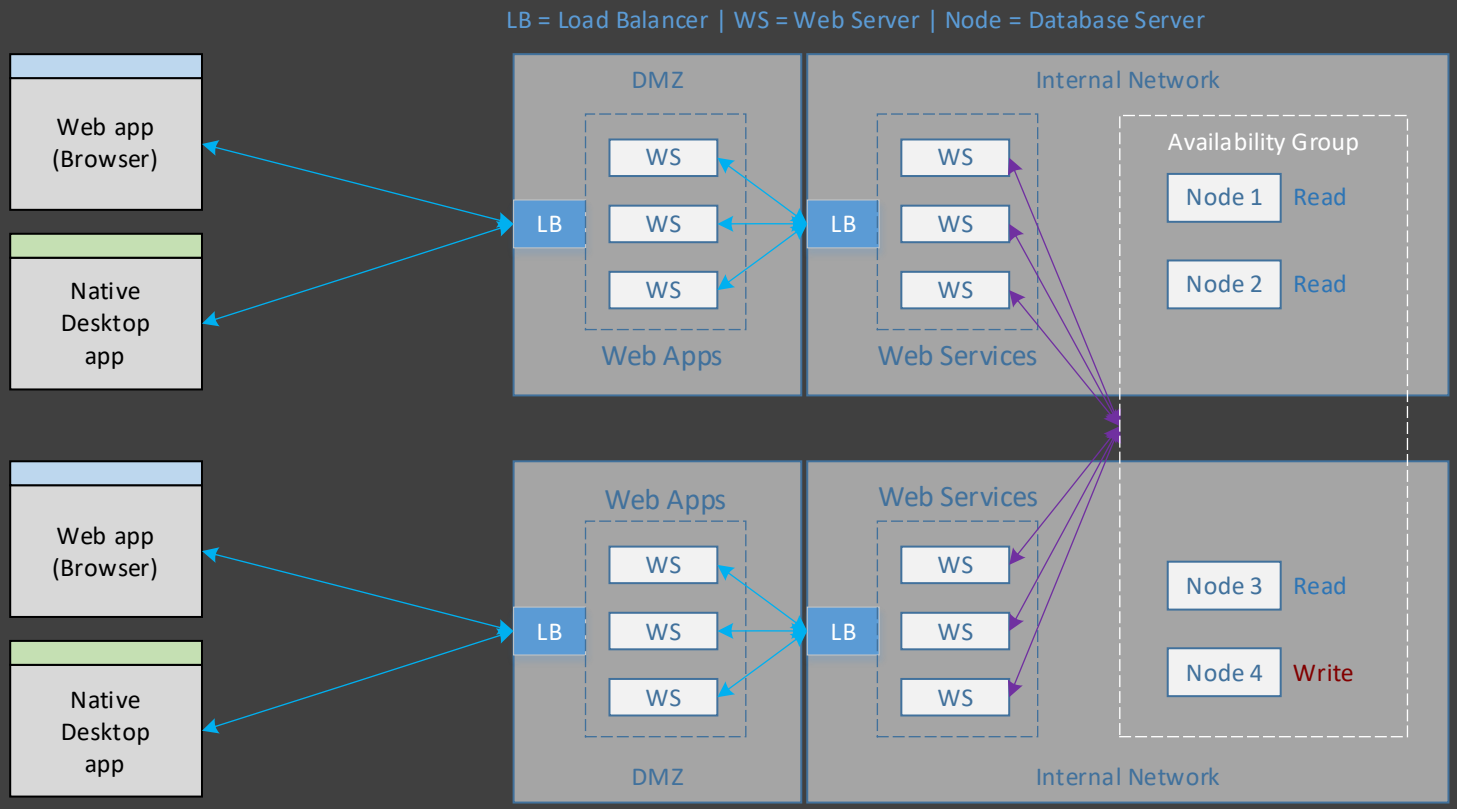
- Standard – a normal database schema
- Replica – a DGP table schema that follows an immutable append-only convention for records, which is the foundation for DGP's omnidirectional data replication functionality
- Shard – a DGP replica table schema that complements certain replica schemas to provide incremental horizontal scalability

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. Shard databases modify the replica schema using a simple shard mechanism to horizontally scale out the central fact tables of star and snowflake schemas. For example, the FileStore application uses the FileShard type of database shard schema to store the segment records of its file content.

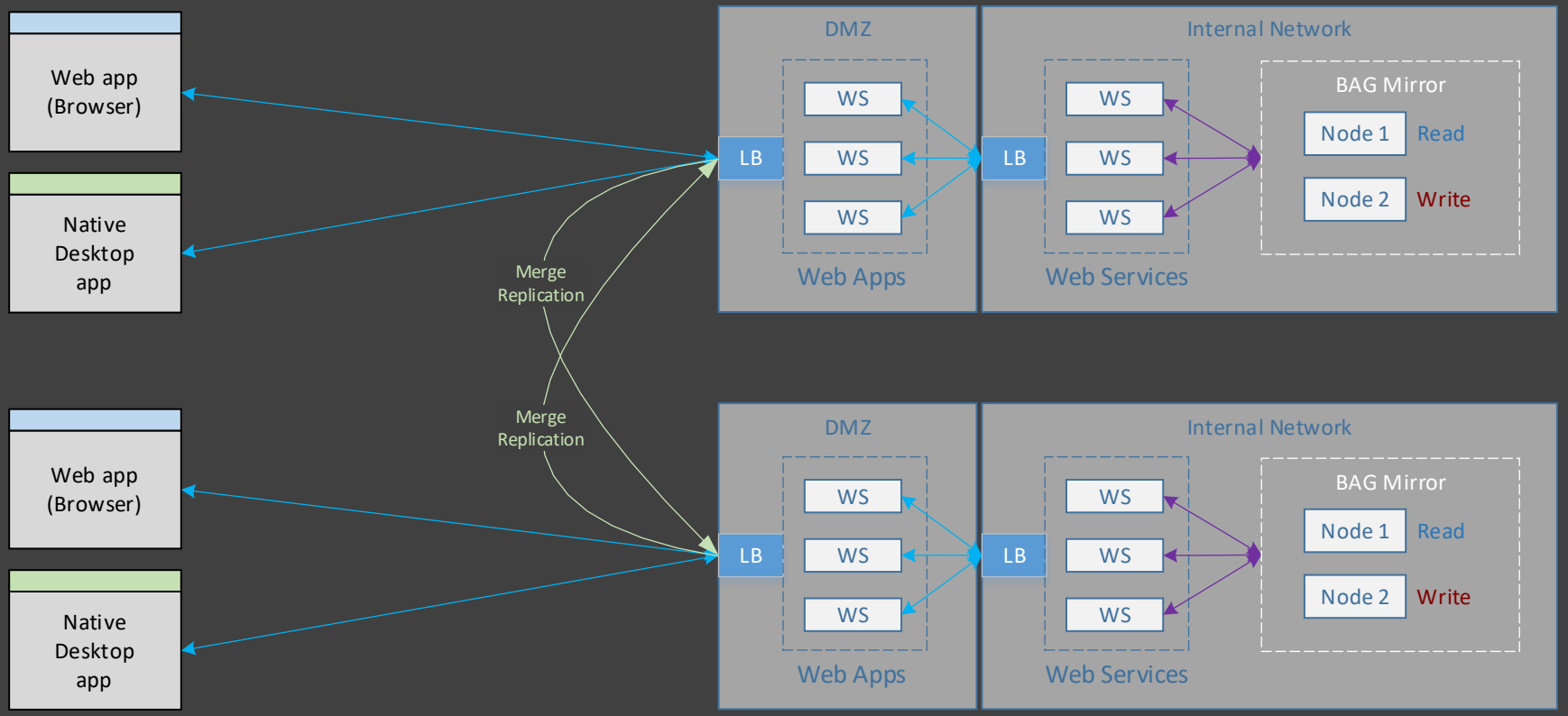
Option 1: SQL Server Availability Groups

Of the 2 alternatives available to provide 100% system uptime, a SQL Server Availability Group configured to span across two datacenters is the preferred solution for business systems at this time. It provides all of the functionality for automatic failover and automatic recovery (including the replicated data). In addition, it also provides the crucial ability to patch and maintain the servers in an availability group individually, while the overall availability group remains in continuous use, with no maintenance windows of planned system downtime required.

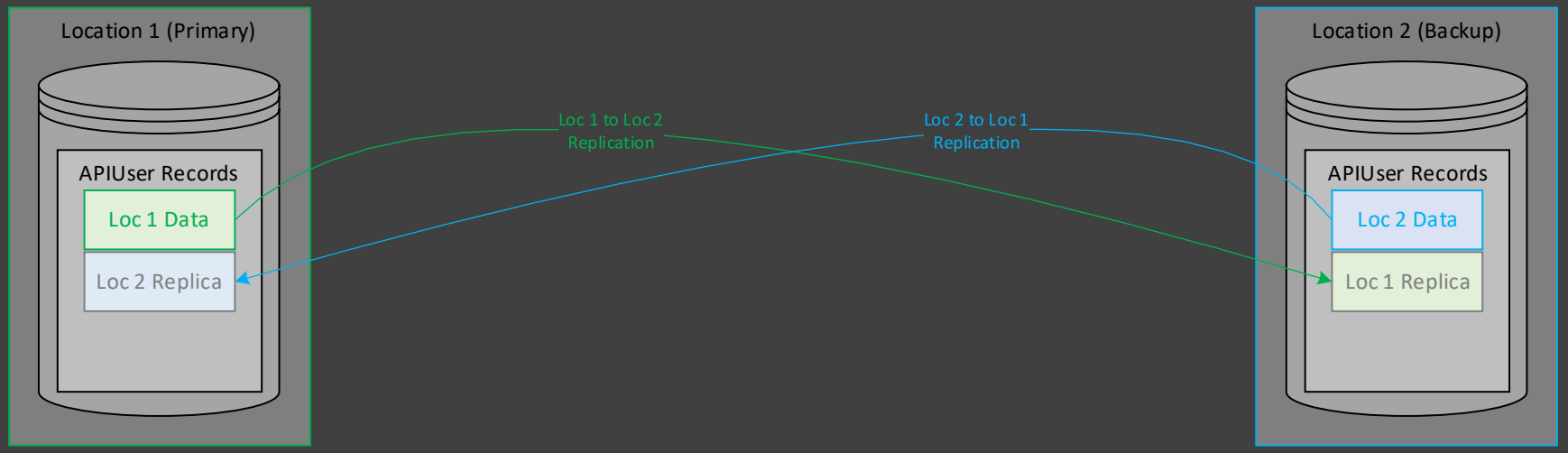
Delivering this functionality requires a combination of HA subsystems (DTC on Windows, Pacemaker on Linux, etc.) plus the operating system working together with the database engine in order for the 100% uptime functionality of availability groups to be possible. The main advantages of availability groups are that they are stable, polished, secure and well supported by Microsoft. The main disadvantage is the high cost of the software licenses.



Option 2: DGP Omnidirectional Merge Replication



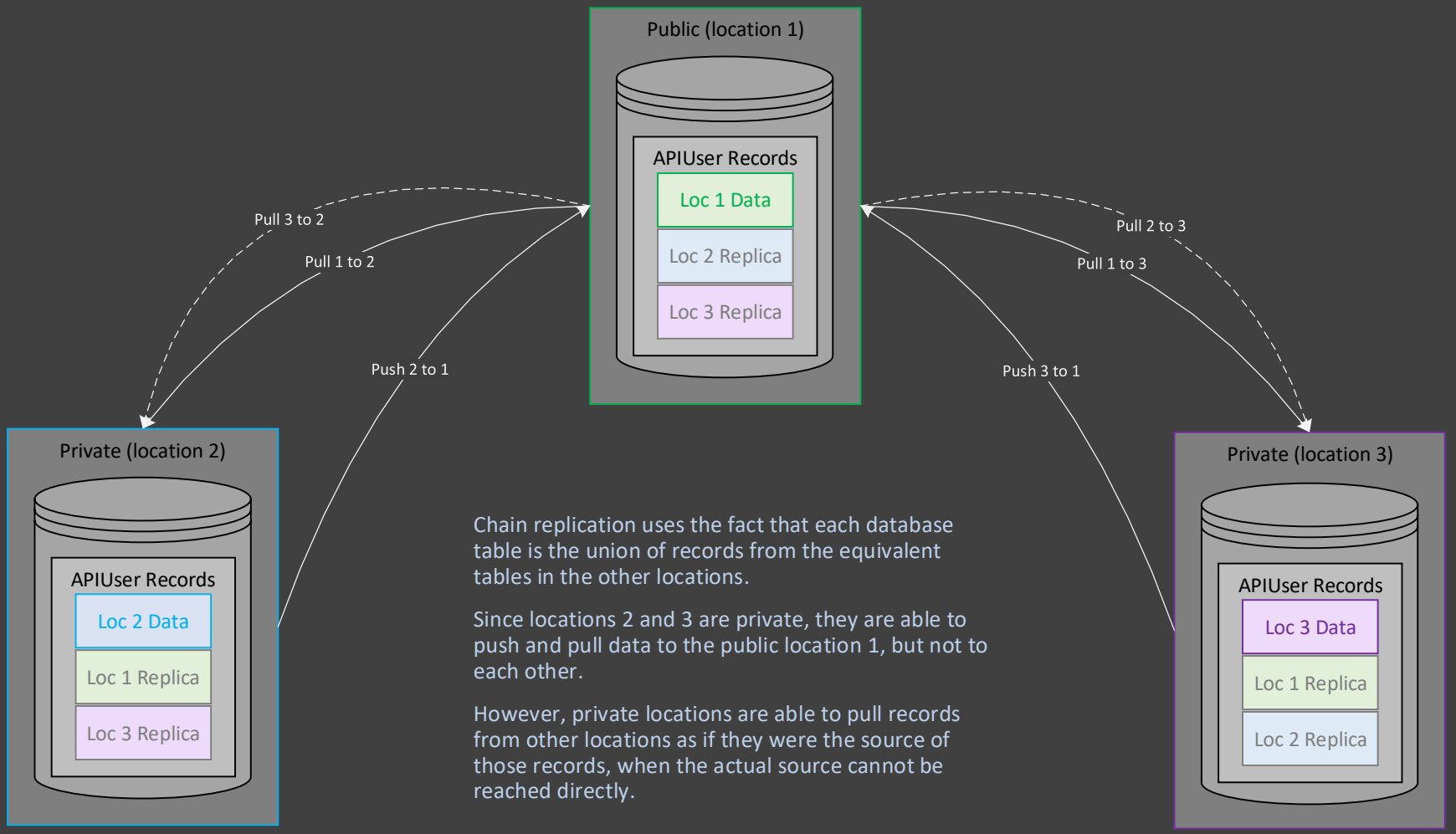
The AutoWork subsystem (not shown in the diagram above) runs the replication, data check, data repair, and other processes continuously in the background to synchronize the data between locations, typically within a few seconds. In large scale systems such as the ones shown in the diagram, the AutoWork scheduler service would be run from the DMZ for improved security of the internal network.



When using the DGP Replica or FileShard 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 between 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.

In terms of CAP theorem, DGP is an AP system with eventual consistency of the data. However, using one location as the Primary for all the work of users (reads and writes) results in a single writable node, which provides strong data consistency. DGP replication then becomes nothing more than a very fast, incremental backup process running continuously in the background.



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.

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

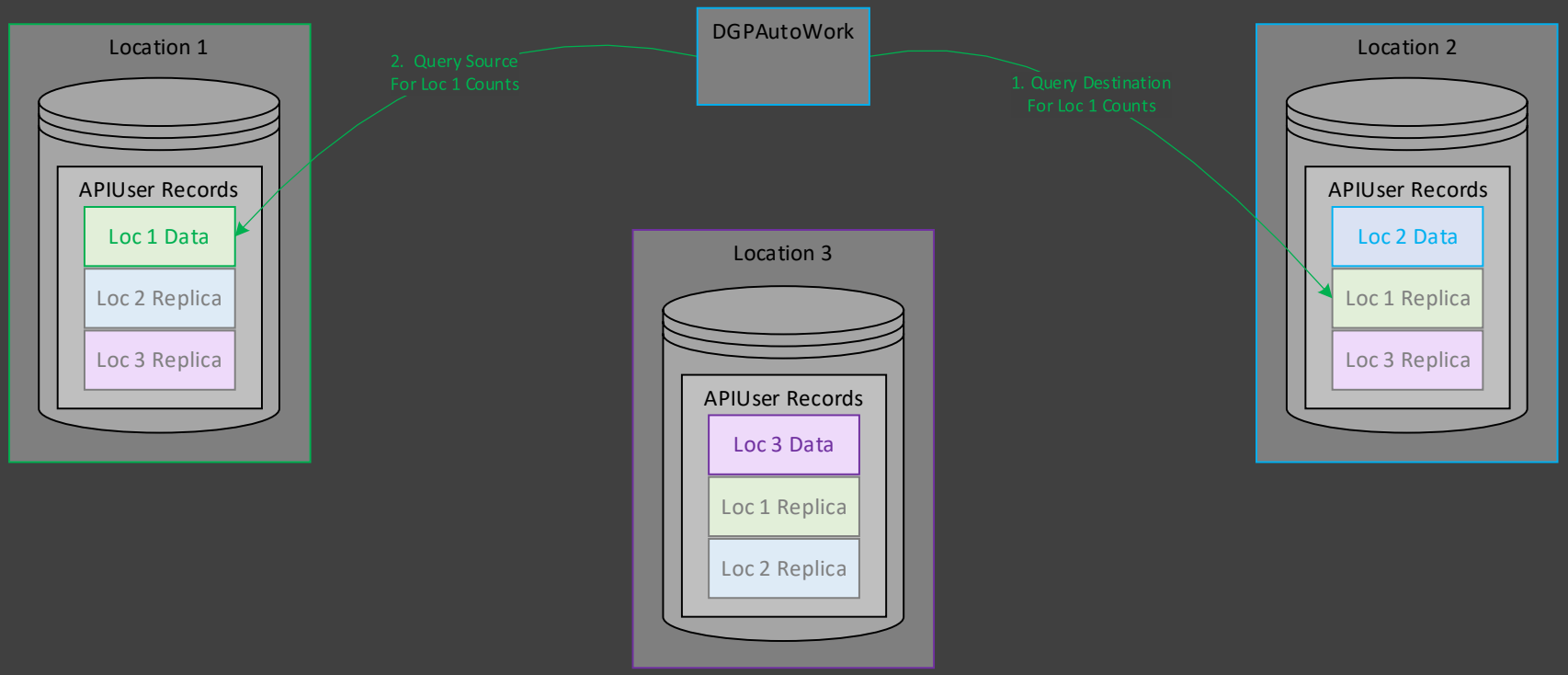
All of the ADO.NET database connection strings are managed in the Web.config file of the web services.

```
<!-- 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="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 is 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 ensure 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.