Technical White Paper

Everpure

Data Virtualization for SQL Server with Everpure

Modernizing and optimizing SQL Server to support new and emerging workloads

Everpure

Contents

Executive summary 3

Solution overview 3

Solution benefits 4

Deployment considerations 5

Read performance 5

Data format 5

Unlocking operational data for new workloads 6

Loading data to Parquet from SQL Server 6

Optimizing SQL Server for active data 8

Protecting retained data 9

Conclusion 9

Resources 9

Executive summary

SQL Server environments often contain years of historical data that must remain available for business, regulatory, or compliance reasons, even when that data no longer needs to be a part of the active operational workload. This historical data can come at a cost. As data grows, maintenance operations consume increasing amounts of time and system resources. At the same time, organizations are increasingly looking for new ways to get value from their data through modern analytics, AI, and other emerging workloads.

Everpure™ object storage on FlashBlade® provides a simple way to address both challenges using SQL Server–native data virtualization capabilities. Data can be exported from SQL Server in open formats such as Parquet, making it accessible to external workloads while active operational data remains on SQL Server as the system of record. Everpure provides a high-performance, resilient, object-based data platform without requiring database administrators (DBAs) to deploy or manage a separate object storage environment.

By keeping active data where it is most useful and moving retained data to a purpose-built, easily administered platform, organizations can simplify SQL Server operations while extending the useful life of their existing data for new workloads.

Solution overview

Combining object storage on FlashBlade with SQL Server–native data virtualization features gives organizations a high-performance, resilient, and massively scalable platform to support modern data initiatives and unlock operational data currently residing on SQL Server.

SQL Server 2022 and later can read and write Parquet data and read Delta Lake data directly on FlashBlade using PolyBase external tables. SQL Server 2025 extends read access via OPENROWSET with no need to install PolyBase. Object storage on FlashBlade provides the object target underneath, supporting trillions of objects with consistent performance as the volume of data grows.

Because this data is written in open formats, it becomes directly readable by other databases, analytics engines, and AI workloads without a separate extract, transform, and load (ETL) pipeline or a second copy to keep in sync. SQL Server remains the system of record, while data in object storage becomes a shared asset the rest of the organization can use as-is.

The same mechanisms used to copy data to object storage can also be used to migrate that data to object storage, removing it entirely from the operational database. This pattern leaves the active operational data on SQL Server to support day-to-day operations, while infrequently accessed data is moved to object storage, where it can be ingested by other systems, still queried by SQL Server, or simply used to retain data for compliance purposes. Either way, the result is the same—less data in operational databases reduces strain on maintenance operations and increases the operational efficiency of SQL Server. In lab testing, this data reduction produced a proportional reduction in the work required for maintenance operations, cutting a maintenance window from 4.5 hours to roughly 12 minutes (a reduction of about 95%).

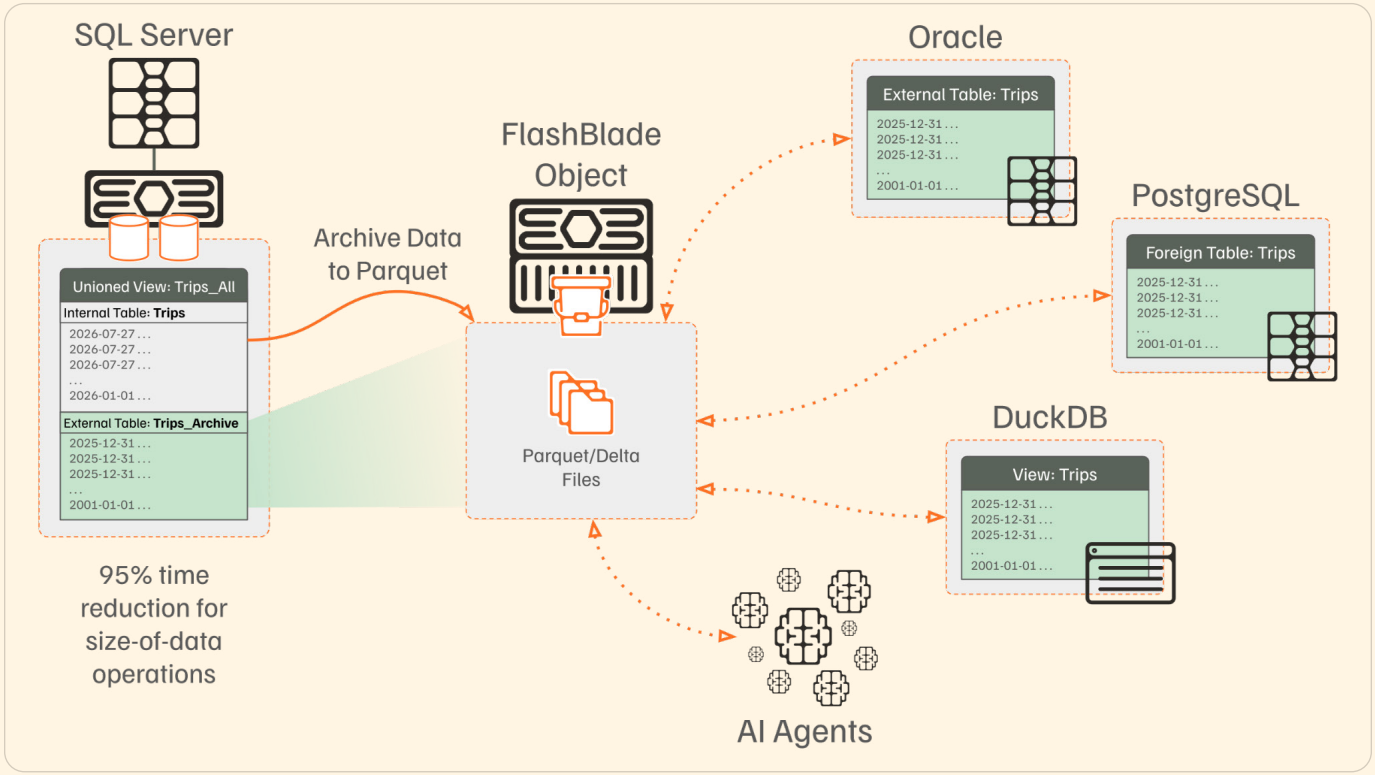


FIGURE 1 Overview of the solution architecture. The solution enables data access across multiple applications while reducing overall data size and accelerating maintenance operations.

Solution benefits

This single architecture can increase access to valuable data while also reducing maintenance overhead.

- **Data accessibility**: Exporting data to object storage presents a single copy of retained data to multiple database systems and platforms, opening new analytics and workload opportunities without additional copies to keep in sync.
- **Reduced maintenance windows**: Tiering lesser-used data to object storage shrinks the primary operational database, reducing the time required for size-of-data operations like index maintenance and consistency checks.
- **Reduced backup sizes**: Because backups only target the data still resident in the operational database, tiering can produce a dramatic reduction in backup size—and correspondingly faster backups and restores.
- **No query performance reduction for hot data**: With "hot" data left in place, tiering has no negative impact on queries reading it and can improve scan performance: retained data no longer competes for space in the buffer pool, so fewer active queries get pushed to disk.
- **Data protected against site loss**: Retained data can be asynchronously replicated to another FlashBlade system or to public cloud object storage, protecting the archive against a site failure independent of whatever high availability or disaster recovery configuration protects the operational SQL Server database.
- **Immutable retention for compliance and ransomware protection**: Object Lock protects retained data from deletion or modification for a defined retention period. This can support the same regulatory retention requirements that often drive data to be kept in the first place, while also closing off a common ransomware attack path.

Deployment considerations

The SQL Server implementation of Parquet over object storage is the biggest factor to keep in mind when deploying external tables on any object platform. Limitations in the data read path, and lack of support for Hive layouts and Delta Lake table registrations, can offer some challenges depending on your goals.

Read performance

In most cases, retrieving data from an external table in SQL Server is not as fast as querying that same data from a database table. This is why it is important to understand the data access patterns of a system before attempting to implement data tiering or other use cases where the data will continue to be queried from SQL Server. Additionally, external table queries take long-living object locks on the external table, or the view referencing the external table, for each query. The length of this lock can depend on a number of factors, including object file format, number of objects, and number of directories being referenced by the external table. For this reason, external tables are not well-suited for highly concurrent access.

Data format

SQL Server supports several data formats for data virtualization, including Parquet and Delta Lake table. Parquet is a popular open format used across many data platform tools. From analytics to AI, Parquet allows efficient querying of data from a set of files without the typical complexities of managing a database system.

The Delta Lake table format takes Parquet and adds table semantics, which adds a host of features such as (but not limited to): ACID-compliant transactions, point-in-time queries, UPDATE/DELETE/MERGE support, more intelligent data pruning, and controlled schema. This functionality is added while still maintaining interoperability with systems that read Parquet directly.

The choice of which format to use depends on the systems consuming it. If SQL Server is the only system accessing the data, a flat directory of Parquet files will work. If the majority of the systems consuming the data support Delta Lake table format, it may be worth the extra effort to implement that format.

Hive partitioning

Hive partitioning is a naming convention used when naming directories within a hierarchy of Parquet files to represent logical data partitions. For example, a dataset representing taxi cab pickup and drop-off times might be partitioned by year and month. Systems that support Hive partitioning can prune whole directories from the total set of files that will be read to fulfill a query. This can increase performance dramatically, as it avoids having to recursively list the contents of the entire object space where the files reside.

SQL Server cannot use Hive pruning to increase query performance. While it can be tempting to forgo using a Hive partition layout, if any systems other than SQL Server will access the data on object storage, a Hive layout could be preferable to increase the performance and efficiency of those systems. While SQL Server does not support loading partitioned data into Parquet, there is a pattern that can be used to accomplish this if a Hive layout is desired. This pattern is discussed later in this paper.

Delta Lake table

SQL Server has read support for the Delta Lake table format, allowing it to use the Delta log to determine which files need to be read. It is important to note that while Delta Lake table format adds support for a number of features, the SQL Server implementation only supports using the Delta log to determine which files to read. The other features such as ACID-compliant transactions are not supported. Depending on the layout of the files on the object store and the object platform being used, Delta Lake table format can have a speed advantage over directly accessing Parquet files.

Unlocking operational data for new workloads

Exporting relational data from SQL Server for use in analytics tools, reporting systems, and data lakes is a common operation for any data engineer. With each export comes a set of challenges:

- Data formatting
- Load generated on the source system
- Human error
- Monitoring and logging
- Data validation

With traditional ETL processes, these challenges get multiplied every time a new workload is added to the existing environment. Using SQL Server data virtualization with object storage on FlashBlade, a data engineer can export the data to a single shared destination in a commonly used format that enables ingestion from a number of different data platforms and ad hoc tools. This approach can reduce repeated extraction and duplication while minimizing the need for downstream teams to repeatedly query production online transaction processing (OLTP) systems.

Loading data to Parquet from SQL Server

When downstream consumers only require file-based analytical access, SQL Server can use `CREATE EXTERNAL TABLE AS SELECT` (CETAS) to export relational data directly to Parquet. Additional design is required when consumers depend on Hive-style partitioning or Delta Lake table semantics.

Hive partitioning

To accomplish Hive partitioning on export, the simplest method is to create external staging tables. The only method to write data to Parquet from SQL Server is via a CETAS query. CETAS queries create and load the external table in one execution and can append to an existing object location using the same syntax.

To accomplish Hive partitioning with SQL Server CETAS queries, the pattern is fairly straightforward:

- Select a partitioning key (year, month, day, and so on).
- Write each partition to its own Hive-style path (for example, `/offloaded_data/year=2026/month=08`).
- Drop the newly created external table.

Dropping an external table simply removes the metadata from SQL Server; the object data is untouched. Because of this behavior, CETAS queries can be used to create partitions, and the table can be dropped immediately after.

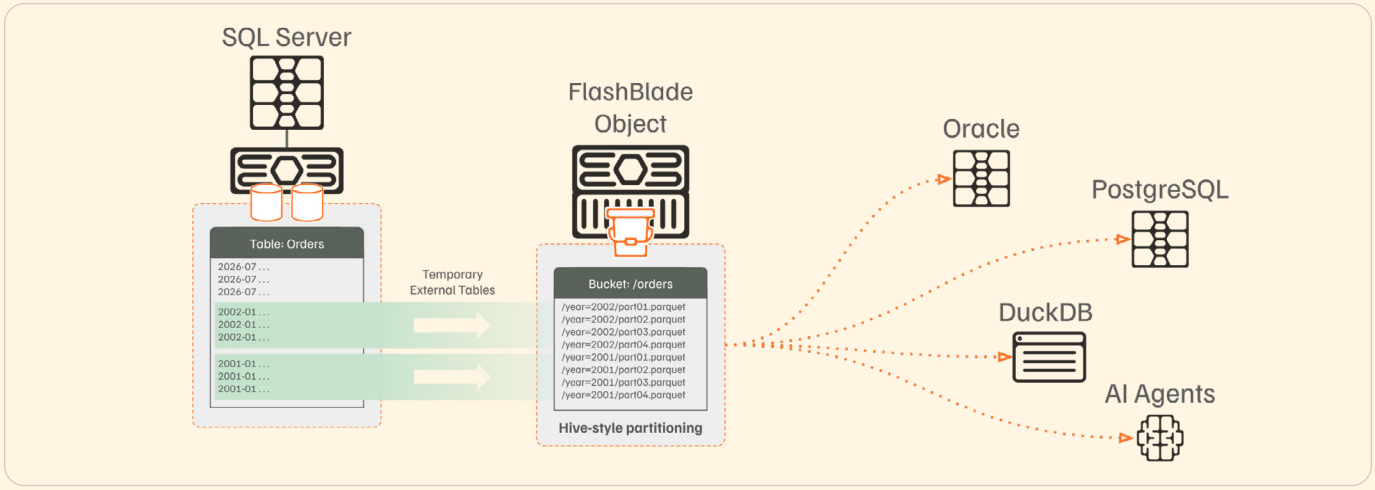


FIGURE 2 How data can be exported to FlashBlade object storage using Hive-style partitioning and made available to other applications.

Delta Lake table

Since SQL Server does not have native support for creating the Delta Lake table format, it can only produce the Parquet files. A separate writer or orchestration service is required to create and maintain the Delta Lake transaction log.

A possible workflow for implementing this pattern would look something like this:

- SQL Server writes Parquet via CETAS queries.
- An external process updates the Delta transaction log. This can be triggered by a job or automated task, or it could potentially be triggered by a REST call from within SQL Server (SQL Server 2025+).
- Consumers read the resulting Delta table.

The Delta Lake table format does add access latency when the external table is being queried from within SQL Server. As with Hive-style partitioning, the choice should be made based on downstream consumers.

Scaling data accessibility

Publishing operational data to Parquet can make that data available to more applications without repeatedly querying the production database. When combined with the replication features of object storage on FlashBlade, the same data can also be made available across remote sites and cloud environments.

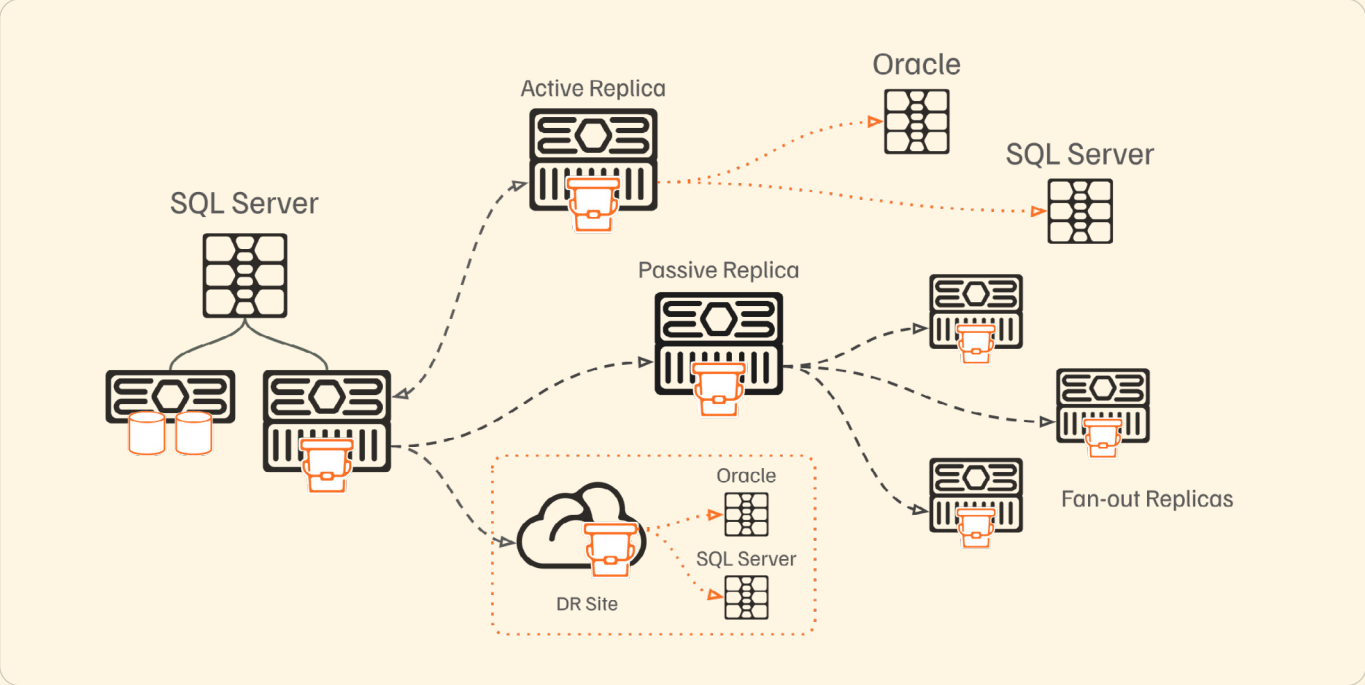


FIGURE 3 Overview of some of the replication topologies supported by object storage on FlashBlade. All links are eventually consistent. Active-passive, active-active, and fan-out are supported.

FlashBlade supports a number of different replication topologies, including fan-out, active-active, and active-passive. Read more in the FlashBlade object replication FAQ.

Optimizing SQL Server for active data

Operations like CHECKDB, backup/restore, and index maintenance are critical for keeping production systems operating efficiently and safely, but they are also size-of-data operations. As operational data grows, maintenance operations take longer to execute and put more strain on production systems. In environments with fixed maintenance windows, this can lead to compromises that increase operational risk.

In some cases, organizations may not be able to delete stale and unused data due to compliance or internal reporting needs. In cases where the access patterns of this retained data are well known, the same architecture used to provide access to other platforms can be used to move infrequently accessed data out of operational database systems and into object storage. The data remains accessible through external tables and can still be queried from the system of record.

In lab testing, a 100GB database was partitioned to keep the most recent year of data in local relational tables, and the remaining data was placed in an external table. This resulted in an ~95% reduction in total database footprint and a ~25 times reduction across several metrics:

- Time to execute a full CHECKDB
- Backup size
- Time to execute index rebuild/reorg operations

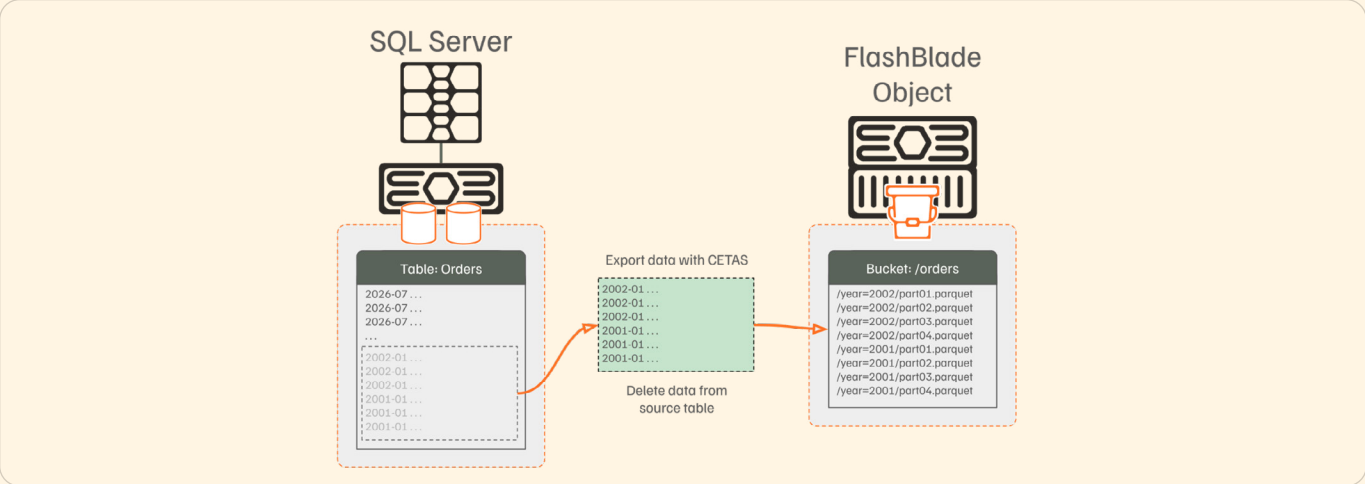


FIGURE 4 Moving data to FlashBlade object storage can reduce maintenance overhead of operational databases while retaining data for compliance and analytical needs.

This pattern is best suited to infrequently accessed or immutable data with predictable access requirements. It is not a transparent replacement for active relational tables with frequent updates, relational constraints, transactional behavior, or intensive indexing requirements. SQL Server external tables are not designed for use cases that require low latency or high concurrency.

Protecting retained data

Data retained for compliance or reporting purposes often carries a requirement that it remain unaltered for a defined period. Object Lock on FlashBlade enforces this directly at the storage layer: once applied, a retained object cannot be deleted or modified until its retention period has passed, including by an administrator. This supports regulatory retention requirements while also keeping data safe from modification.

Conclusion

SQL Server data virtualization with object storage on FlashBlade allows organizations to retain and share access to operational data without keeping every byte in the active database or creating a separate copy for each consumer. Parquet-based data can be queried from SQL Server and shared with analytics, reporting, AI, and other platforms, while infrequently accessed data can be moved out of the operational footprint to reduce maintenance overhead.

FlashBlade provides a massively scalable object platform built to support these patterns as retained data grows over time, without requiring a rearchitecture as object counts increase. With that foundation in place, object storage on FlashBlade extends the operational data in SQL Server into a shared, scalable data access layer without replacing the relational database at the center of the environment.

Resources

- FlashBlade object services
- FlashBlade object replication FAQ

Learn More About FlashBlade Object Storage

Visit Our Website | **800.379.PURE**

©2026 Everpure, the Everpure P Logo, Pure Storage, FlashBlade, and the marks in the Everpure Trademark List are trademarks or registered trademarks of Everpure, Inc. or its licensed subsidiaries in the U.S. and/or other countries. The Trademark List can be found at everpuredata.com/trademarks. Other names may be trademarks of their respective owners.

ID55987-01-en-09/26