Use big data technology to set up an active archive for the data from the enterprise data warehouse. Learn how a big data solution based on Hadoop is an ideal platform to build an active archive for historical data in a data warehouse. This article covers the architecture, tools, and techniques that you can use to design an active archive.

View more content in this series

Other articles in this series

Part 1 of this series covers the current data warehouse landscape and identifies the data warehouse augmentation use cases. Part 2 describes how to use the big data platform as the landing zone for the data warehouse. This article explores building an archive that can be queried by using the big data platform. Part 4 describes how to use the big data platform for data exploration.

Enterprises have built data warehouses for the last few decades. After it's built and put into production, a typical enterprise data warehouse becomes the backbone of information for the company. Today organizations generate more data because of the increased capability to capture data from various data sources. Databases, ETL tools, and reporting tools are better at handling large data. But the rate of growth of data in complex data warehouses that serve many lines of business make it necessary to archive less frequently used data and keep only the actively accessed data in the warehouse.

Hadoop is fast becoming the standard for large-scale data analysis. High-level query languages and developer aids make it easier to produce and process MapReduce jobs. Mechanisms such as JDBC and ODBC drivers enable better integration with existing tools.

Data warehouse archiving

Data warehouse archiving is the process of moving data that is not likely needed to conduct business operations from the data warehouse to a medium where it can be stored for long-term
retention and retrieval if needed. Requirements for data warehouse archival need to be identified early and included in the business requirements for the warehouse because they affect the design of the warehouse. These business requirements include:

- Mechanism to select criteria to specify the records that can be moved to the archive
- A format and storage repository for archived data
- Method for specifying the likelihood of retention and the time frame of retrieval

Figure 1. Data lifecycle in a data warehouse

![Data Lifecycle Diagram]

After the data is archived, it is not available for querying unless it is restored from the backup archive and loaded back into the data warehouse. To enable business continuity and to prepare for disaster, organizations often store backup data in a different location or data center. Because retrieval requires coordinating between multiple groups, the process can be long.

Factors that influence decisions about what data to retain

If an organization requires the data to conduct business, the data must be retained. In some cases, historical data is saved to compare historical trends with current events. Often, data scientists analyze existing data to discover insights that can yield business opportunities. Analytically oriented companies have a stronger need for long-term data retention.

The growing number of regulations and the need for organizations to be in compliance affects the decision of what data to retain. Regulations such as the Sarbanes-Oxley Act, HIPAA, and BASEL II govern how long data must be retained.

Organizations are retaining more data over longer periods of time. This additional data to be retained creates the need for new policies, procedures, methods, and software to support storage, management, and access of archived data.

Typical data warehouse archival implementation

In data warehouses that are designed to store large volumes of data, it is common to store transaction-level details for a few years. Depending on requirements, data older than two to three years is moved to an archive medium such as tape. Summarized aggregated data is stored in the warehouse for historical analysis. However, after data is archived, the transaction-level details must be restored to the data warehouse to analyze the historical data. Figure 2 shows the typical data warehouse archival process.
Figure 2. Traditional archiving of data from an enterprise data warehouse

Major database vendors provide data partitioning, which plays a major role in implementing a backup strategy in a warehouse. It is typical for data warehouses to partition data by using time as the criterion for selecting records to archive. Traditional archiving solutions rely on advanced backup techniques such as incremental backup and the ability to take partitions offline for backup purposes without affecting the operations.

Any implementation of data warehouse archive strategy must consider backup and restore requirements. Archiving to offline, auxiliary storage as shown in Figure 3 is not suitable for situations in which detailed data must be restored to enable historical analysis. Because data cannot be retrieved quickly from auxiliary storage, business users might omit detailed history from an analysis or make estimates that are based on aggregated data.

Figure 3. Typical data warehouse archive implementation

Implementing a data warehouse archive can be complicated by several factors:

- Data warehouse structure and schema might change to accommodate business requirements. Schema changes within a data warehouse must account for backup and restore compatibility. In complex warehouses, these changes can cause problems.
- Offline data archive media such as tape and disk are prone to failure with age and require planned mock tests to ensure data availability.
- Restore operations usually require coordination with multiple groups and people to minimize downtime and to accommodate off-site storage of backup media.
- Restoring historical data might affect normal warehouse operations and must be carefully planned.
Changes in tape formats and requirements to migrate to the most current format affect the backup data that is stored on tape.

**Active archive with the big data platform**

Apache Hadoop provides an alternative for implementing an active archive that you can query. Hadoop is especially suited to situations in which a large volume of data must be retained for a time. As shown in Figure 4, the big data platform includes storage, compute, and query layers that make it possible to implement an active archive.

**Figure 4. Archiving by using the big data platform**

Storage layer

Historical data that is kept in a data warehouse to accommodate archive policies can be moved to the storage layer of the big data platform. The storage layer acts as the archive medium for the data warehouse. The Hadoop Distributed File System (HDFS), which is the base on which most big data platforms are built, is ideal for storing large amounts of data that is distributed over commodity nodes. HDFS is a write-once system and historical data is typically backed up once and is never overwritten. HDFS features such as scalability, fault tolerance, and streaming data access make it suitable for active archives.

Compute layer

Any computation or processing that needs to be done on an active archive must be able to process large amounts of data. The MapReduce framework provides a way to distribute computations over a large set of commodity nodes that are stored on top of a distributed file system such as HDFS. During the map phase, input data is processed item by item and transformed into an intermediate data set. During the reduce phase, data is converted into a summarized data set. MapReduce jobs can be written in a number of languages, including the Java™ programming language and Python. Other high-level scripting and query languages make it easier to do computations in the MapReduce framework.
Query layer

Active archive must be able to query the data easily and perform computations, aggregations, and other typical SQL operations. Although all operations need to be done as MapReduce jobs in the big data platform, writing MapReduce jobs in languages such as the Java programming language makes the code less intuitive and less convenient. The query layer provides a way to specify the operations for analyzing, loading, and saving data from the distributed file system. It orchestrates the processing of jobs in the big data platform. Many commercial and open source variants such as Hive, Pig, and Jaql can be used to query data easily from the active archive that is stored in the big data platform.

Active archive design

Data warehouse is designed based on business requirements and the active archive is an extension of the warehouse. Active archive must provide a way to query and extract data with semantics that are consistent with the data warehouse. Metadata information that is developed for the warehouse must be applied to the active archive so that users familiar with the warehouse are able to understand the active archive without too much difficulty. Carefully select an infrastructure, data layout, and system or record for the active archive.

Infrastructure

If an organization chooses to build an active archive as the first step in implementing a big data platform, it must analyze whether to build or buy the services that are needed. In many cases, an active archive is only one part of a larger data analytics strategy. Building a big data platform from scratch requires expertise and infrastructure to set up a scalable and extensible platform that can address current needs and scale to accommodate future requirements. Although Hadoop-based systems can run on commodity hardware, a big data solution includes system management software, networking capability, and extra capacity for analytical processing. Big data appliances such as IBM® PureData™ Appliance for Hadoop provides a big data infrastructure to address present and future needs.

The active archive Hadoop infrastructure must be sized to accommodate the amount of data to be stored. Consider the replication factor within Hadoop and the efficiency that can be achieved by compressing data in Hadoop. The replication factor represents the number of copies of each data slice. Depending on the number of data nodes, one or more management nodes might be needed. The number of racks that are needed for data nodes are also an important factor to be considered during infrastructure design.

Data layout

One of the most important design decisions is the layout of the data in the big data platform. Because the active archive must store a huge volume of data, an appropriate structure to organize the data is important. This structure affects query performance and how computations are done against the data within the active archive. The structure must be scalable so that data can be added incrementally from the data warehouse when the data is ready to be archived.
As shown in Figure 5, the partition scheme that is used in the data warehouse can be used to arrange the folder structure within the distributed file system. Any distribution or clustering key that is used for the main fact tables can be used to organize subfolders. For example, a telecommunications data warehouse stores call data records that are partitioned by months. If the daily volume of data is in the millions, the active archive might use month as the main folder with the data for individual days that are stored in subfolders.

**Figure 5. Layout of archived data in the big data platform**

![Diagram showing the layout of archived data in the big data platform.](image)

**System of record**

As shown in Figure 6, data from the data warehouse must be moved into flat files in the big data platform for the active archive. A system of record in the data warehouse refers to the combination of data elements to describe a business process across tables. A data warehouse system of record must be converted to a Hadoop system of record so that the active archive can serve a similar purpose for historical data. If the purpose of the active archive is merely to store historical data for analytical purposes and the restore function is not required, the system of record in Hadoop can be designed to suit the requirements and does not have to mirror the data warehouse system of record. For example, data elements that are needed from various tables in the data warehouse can be combined into a Hadoop system of record for the active archive.

**Figure 6. Layout of data elements in active archive**

![Diagram showing the layout of data elements in active archive.](image)

If the restoring of data is required, all data elements that are required in the data warehouse must be moved into the big data system. The system of record in Hadoop might span multiple files,
and queries on the big data platform might require operations similar to SQL join statements. In this case, converting a system of record from the data warehouse to Hadoop during the archive operation (or the reverse) during a restore operation must be possible.

**Figure 7. Alternative layout of archived data in the big data platform**

![Alternative layout of archived data in the big data platform](image)

**Tools and techniques**

The big data platform provides many tools to help implement an active archive. Open source tools and commercial products support moving data in and out of the active archive. Tools range from ODBC and JDBC drivers for client connectivity, data import and export tools, scripting languages, and other helper utilities for managing the active archive. Techniques available in these tools can be used to implement an efficient archive system.

**High-level query languages (Hive, Pig, and Jaql)**

One of the important aspects of active archive is the need to query the data that is backed up in the archive for data analysis. Hive is a data warehousing application with HQL as the query language, which is conceptually similar to SQL. Pig is a large-scale data processing system in which data-processing pipelines can be written by using Pig Latin. Jaql is developed by IBM primarily for JSON data, but it can be used to analyze structured and unstructured data that is stored in HDFS. In each case, code that is written in a high-level language is compiled into MapReduce code internally by the compilers. This approach saves users from having to learn to code MapReduce functions.

**SQOOP**

SQOOP is an Apache open source project that is designed to import large volumes of data from relational databases to HDFS. SQOOP can run data load jobs in the MapReduce cluster and use fault tolerance and parallel computing capabilities. SQOOP can be used to connect to any relational database that supports JDBC drivers and extract data by using ANSI standard SQL queries. It can extract data that can be put into the big data platform. SQOOP can also load the data into HIVE or HBASE for further analysis.

Large tables can be split into multiple jobs with multiple mappers by using a column in the database. Data that is extracted can be compressed and written to HDFS to save space.
ETL tool integration with big data

All major commercial ETL tools provide capabilities to integrate with the big data platform. IBM InfoSphere® DataStage® provides big data file stage (BDFS), which can connect to Hadoop and read or write data into the HDFS directly. A benefit of using an ETL tool to export and import data is the ability to integrate the archive and restore process with the ETL process and use the scalability features of DataStage. These features include the ability to partition data and the ability to scale the process by using a UI-based design. Because archiving and restoring is a periodic process, consider how it affects existing ETL processes and ETL performance. One disadvantage of using ETL tools for active archive processes is that a large volume of data is sent over the network because an ETL tool typically runs on a different infrastructure on the network.

Flume

Flume is an Apache open source, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. Flume is used to collect data from various sources such as JMS, Solr, and log-based data sources. Flume-Next Generation (Flume-Ng), the latest version of the tool, has a three-tier design: agent, collector, and storage. To collect data from relational database tables, use Flume agents for the archive process or it can create flat files for the restore process that can be loaded by database loaders. These functions are also available with SQOOP, which is more commonly used to read or write data from relational databases. For the archive process, you can set different agents to collect from different tables to parallelize the workload in Flume, based on events.

During the restore process, agents get data from HDFS through collectors. The agents can send the data through the collectors to produce flat files, which can be used to load data back into the data warehouse tables.

IBM Big SQL and Cloudera Impala

IBM Big SQL and Cloudera Impala are query engines that enable SQL-like queries on data that is stored in Apache Hive and HBase. These tools are important components of an active archive because they enable data warehouse users to extract data through their client tools. Both these engines provide JDBC and ODBC connectors and create SQL that is compliant with ANSI standards so that the business intelligence tools can connect to the active archive and retrieve data for reporting. Data warehouse users can write SQL-like queries that get translated into MapReduce jobs internally to use the MPPMassive parallel processing point queries: Simple queries that do not need MapReduce, which are detected and executed as point queries on a single machine. Reduces overhead. capability of the big data platform. These engines also support point queries, which operate at low latency and can return results quickly.

IBM BigSheets

IBM BigSheets, available in IBM InfoSphere BigInsights™ is a spreadsheet-like interface to easily manipulate, query, and visualize data. Users can use BigSheets, an alternative to traditional business intelligence tools, to query the active archive easily. BigSheets can be used to create
data sets by adding new columns that specify formulas such as those used in a spreadsheet. It can be used to perform join and union operations on data without writing code in SQL or in other scripting languages. BigSheets includes visualization charts that can be used as is, without configuration.

**Active archive architecture**

As shown in Figure 8, the active archive interacts with other components and processes, including:

- Client applications
- The archive process
- The restore process

**Figure 8. Architecture for the active archive**

Client applications

Client applications for the active archive are primarily the applications that connect to the big data system to get the data into analytical systems, such as the business intelligence (BI) tools for reporting. As shown in Figure 9, most of the BI tools let you use JDBC/ODBC drivers to send SQL-compliant queries that can be converted to MapReduce functions. The results are streamed back to the BI system. Data scientists can use MapReduce techniques or high-level query languages such as Hive or Pig to perform historical trend analysis on the active archive. These tools also let you move data out of the active archive to other systems so that the data can be combined and insights can be derived.
The archive process

The archive process must identify and move all the records from the data warehouse that need to be moved, based on the requirements and all the supporting information. As shown in Figure 10, the system of record from the data warehouse needs to be converted to the Hadoop system of record. After the record is written to the big data system and is available in the active archive, the record in the data warehouse can be marked as ready to be purged. The archive process must ensure that records archived in a particular batch are appended to the existing data set based on the data layout design. The archive process must make the data available for client applications through queries. Typically, the archive process is scheduled and runs on a periodic basis.

The restore process

The restore process is almost a reverse of the archive process. As shown in Figure 11, the restore process reads from the big data platform and identifies the records to be restored based on requirements. The Hadoop system of record needs to be converted to the data warehouse system of record. Unlike the archive, restore operations can be complex. The write operations that send
Data back to the data warehouse need to accommodate foreign keys, indexes, surrogate keys, and similar factors. The operations must be in the correct sequence. The design of the archive process and the restore process must include operations that clean up the data warehouse and the active archive so that future archive and restore operations will work correctly. Even though the restore process occurs only on demand, frequent test runs need to be performed to make sure the restore operations perform correctly with the current state of the data warehouse.

**Figure 11. Restore process**

![Restore process diagram]

**Summary**

Dealing with historical data for data mining and regulation purposes is an important part of data warehouse design. For large data warehouses, historical data is a big data problem, which ideally suits the characteristics of systems that Hadoop is designed for. Big data platforms that are based on Hadoop offer a cheaper alternative for creating active archives that client applications can use. Part 4 of this series describes how to use big data technologies for initial data exploration.
Resources

- **Architecting a big data platform for analytics**: Provides an end-to-end overview of architectural solutions and strategies for a big data platform.
- **IBM Big data warehouse augmentation**: Video that describes various types of data warehouse augmentation scenarios.
- **Why IBM for big data**: Describes the software that makes up the IBM big data platform.
- "**Big data architecture and patterns**" (Shweta Jain, Shrikant L Khupat, Divakar Mysore, IBM developerWorks, September 2013): Covers various architectural patterns with big data.
- **Big data’s new use cases: Transformation, active archive, and exploration** by Amr Awadallah: Provides Cloudera's point of view on data warehouse augmentation and active archives.
- **IBM Redbooks: Performance and capacity implications for big data**: Gives an overview and benefits of big data, includes critical factors in managing big data successfully, and describes emerging technologies that address big data challenges.
About the author

Arun Balasubramanyan

Arun Balasubramanyan is an information architect with IBM in the Global Business Services (GBS) division. He has more than 15 years of IT experience in architecture, design, and implementation of large-scale data warehousing and business intelligence projects across various domains. He also has hands-on experience in big data technologies and data warehouse augmentation. His main areas of interests include data visualization, open data, and big data patterns.

© Copyright IBM Corporation 2014
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)