**Content**

**Data Model Design**

**Background**

**Requirements**

**Architecture diagram**

**Solution**

**ETL Architecture**

**Overall Purpose of ETL Architecture**

**How this ETL Architecture is Possible**

**Conclusion**

**Data Model Design**

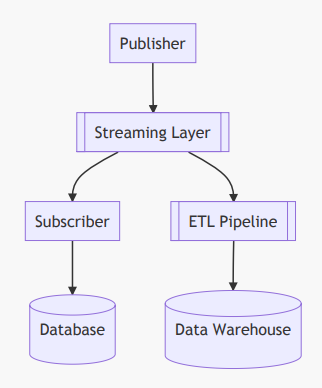
**Background**

Cohesion connects with commercial building's automation systems that manages Heating,Air Conditioning and Ventilation (HVAC) and lighting controllers. These systems employs various devices and sensors that generates important information to understand building operations and system efficiency. Cohesion is interested in storing data from these devices and analyzing with other data sources.

**Requirements**

Need to Model a design for the Data Warehousing system, with emphasis on what would be different from the streaming data model and with the help of Big Data plan/technologies.

**Architecture diagram**

****

**Solution:**

Since Need to design a warehousing system for streaming data from automation systems, I would like to load the data into azure blob storage at certain interval if need to extract insight in near real time.

That means I would like to load the data into Azure blob storage in certain time interval in its original file format , that is json file which unstructured file format.

Then once the data loaded into blob storage, then we could leverage Azure databricks for transforming the loaded data because Databricks provides widerange of service from getting secured connection to blob storage to processing data without compromising qualities of data using Spark execution engine with python api.

Using databricks runtime where having wide range of components like spark execution engine, sql queries ,python api, we can analyze data effectively. Along with that ,Delta Lake and Delta table support from Databricks enable data read and write in Parquet will be very efficient.

Further with Help of unity catalog we have wide range facilities like

1.Complete control over the data using the hierarchies Metastore -> catalog -> schema -> Table/view (also called 3 level namespace)

2. Can run data sql queires against the delta table loaded from databricks after transformation.

3.Using Delta sharing we can share final output/data to outside in secured way for further analysis like BI report /Dashboard creation.

**ETL Architecture :**

**Extract**: Load data from source IoT in certain time interval in batch mode into azure blob storage in its original form.

**Transform**: Using Azure Databricks ,perform required data transformation

**Load:** Finally load the data into any sql database .For example azure sql database.

**Overall Purpose of ETL Architecture**

**Data Warehousing System**:

* + **Purpose**: Used for storing and managing large volumes of historical data for analysis, reporting, and business intelligence. The data typically comes from multiple sources, is cleaned, transformed, and stored in a structured format.
  + **Architecture**: The architecture for a data warehouse is often **batch-oriented** and typically involves an **ETL (Extract, Transform, Load)** process. The data is pulled from transactional systems or external sources, transformed (cleaned and aggregated), and loaded into a **star schema** or **snowflake schema** for analysis.
  + **Storage**: Typically stored in relational databases like **Azure SQL Data Warehouse** (now **Azure Synapse Analytics**) or **Azure SQL Database**. This supports SQL-based querying, OLAP operations, and reporting.

**Data Ingestion**

* + **Batch Ingestion**: Data is typically ingested in batches, often scheduled at intervals (daily, hourly, etc.). Tools like **Azure Data Factory** are used to automate this ETL/ELT process.
  + **Source**: Data comes from transactional databases, external APIs, or flat files like CSVs. Often involves historical, static data.

**Data Storage Models**

* + **Structured Data**: Uses a **relational model**, typically in a **star schema** or **snowflake schema**, which organizes data into facts and dimensions. Each fact table represents measurable business events (e.g.,device), while dimensions provide context (e.g., device model,device reading).
  + **Data Lake**: Optionally, raw data might first be landed in a **Data Lake (Azure Data Lake Storage)** for long-term storage and later transformed and loaded into a warehouse.

**Data Processing**

* + **Batch Processing**: Data is typically processed in batches using **ETL/ELT processes**. Tools like **Azure Data Factory** or **Azure Datafactory** orchestrate these processes, where raw data is cleansed, aggregated, and transformed before loading into the data warehouse.
  + **Transformations**: The transformations are often complex and involve historical aggregation, summarization, and normalization, optimized for **OLAP (Online Analytical Processing)**.

**Data Querying & Analytics**

* + **Querying**: Data is queried using **SQL**-based queries, optimized for complex queries and multi-dimensional analysis. Tools like **Power BI** or **SQL Server Management Studio (SSMS)** are used to generate reports, perform analytics, and create dashboards.
  + **OLAP Operations**: Common operations include **drill-down**, **slice and dice**, **pivoting**, and **aggregation**, often using **cube** models for rapid query responses.
  + **Latency**: Queries are typically less frequent but involve complex operations, meaning higher-latency responses (acceptable because the data is historical).

**Scalability and Performance**

* + **Scalability**: Azure Synapse Analytics or Azure SQL Data Warehouse provides high scalability with features like **distributed querying** and **massively parallel processing (MPP)**. However, these are designed for large, complex batch jobs rather than continuous workloads.
  + **Performance**: Query optimization and data indexing are critical for enhancing the performance of large-scale queries in a data warehouse.

**How this ETL Architecture is Possible:**

Azure provide wide range of service to extract data from external sources in both real time and batch mode and load the extracted data into azure storage location from where after getting connected to the storage location with help of secret scope and key in Azure Key-valute using Blob service client module

This Blob Service client module support for creating blob level access,so that we can download/read and write data from/to storage account irrespective of it original storage format and to delta format.

Once establishing connection with azure storage, Spark dataframe’s schema evolution we can convert the unstructured json file into structured table format for further analysis.Below gives the method to convert json to dataframe with auto schema evolution.

**df = spark.read.json("path\_to\_your\_json\_file.json")**

Else, if need to define schema we can do it as follows.

**df = spark.read.schema(schema).json("path\_to\_your\_json\_file.json")**

Here schema: schema defined as follows.

schema = StructType([ StructField("id", IntegerType(), True),

StructField("name", StringType(), True),

StructField("age", IntegerType(), True) ])

If we need to merge multiple json files provided if they have common key, then we can join them into unified structured table which will make further analysis easy.

Finally loading the combined structed table in delta format as follows.

**df.write.parquet("output\_path")**

**df.write.format(‘delta’).option(mode,’overwrite’).load(path)**

**Conclusion: Key Differences**

1. **Data Flow**: Data warehouses rely on batch processing (historical data), while streaming models process data in real-time.
2. **Storage**: Data warehouses use structured relational databases, while streaming data systems work with raw or semi-structured data.
3. **Processing**: Data warehouse processes are optimized for complex, historical queries, while streaming data systems are optimized for real-time event processing and decision-making.
4. **Latency**: Data warehouses are designed for lower-frequency, high-complexity queries, whereas streaming systems focus on high-frequency, low-latency data processing.

In short, a **Data Warehousing system** focuses on analyzing historical, structured data