International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Effective Payroll Processing system for Retail Employees using Databricks Ingestion Framework and Salesforce Approval Workflow

Vamshi Krishna Malthummeda

mvamsikhyd@gmail.com

Abstract:
An effective payroll processing system is a hallmark of a well-functioning organization which provides key benefits like accurate calculations of wages, taxes, benefits, adherence to compliance & regulations and finally Improved Employee satisfaction along with low attrition rate. This paper proposes a solution for payroll processing by leveraging Databricks (built on Apache Spark which can handle massive datasets and perform complex computations) and Salesforce approval workflow (helps in review & routing of timesheet approval requests, enables quicker decision making, helps in tracking progress and status of timesheet approvals and provides comprehensive audit trail). In the proposed solution the databricks job will process raw data files containing employee information, retail chain hierarchy information, benefits information, employee task assignment information, login activity information, scheduling information etc. and prepare the timesheet dataset and pushes it into Salesforce connected App for approval process using out of the box bulk API 2.0 REST call. The solution saves lot of time for application maintenance team during the weekly payroll processing day, very adaptable to the new compliance rules, provides higher levels of satisfaction by compensating timely and helps higher management in making informed decisions. The proposed framework offers a resilient, scalable, secured and accurate solution which results in better analytics and workforce retention.

Solution Description:
Initially the datasets required for payroll processing of the store employees identified which includes:

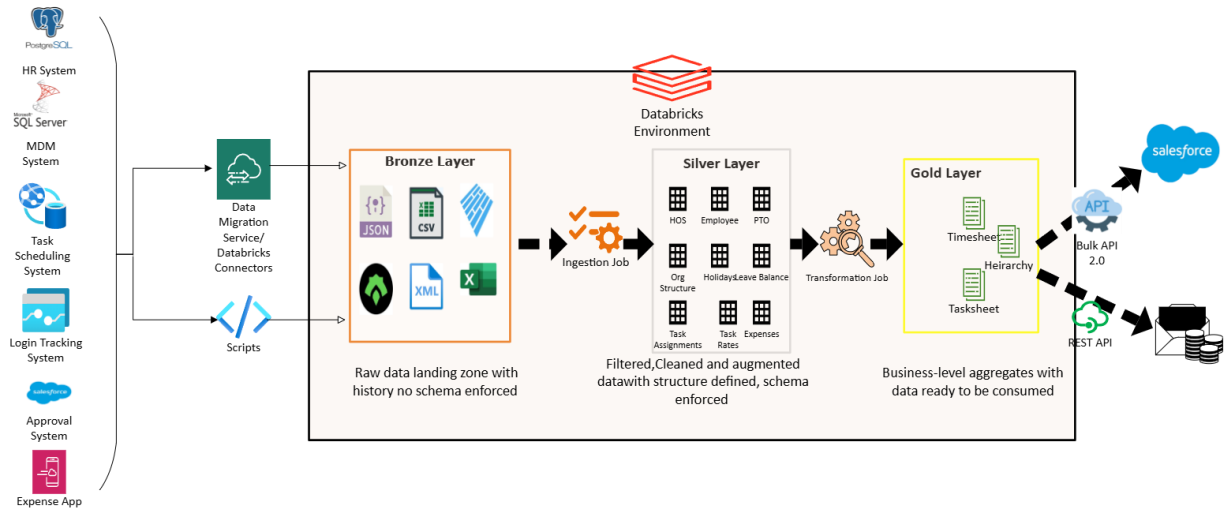
- Employee Information (Name, ID, P&L Area, Employment Type (Part-Time/Full-Time), Employment Status (Active, Inactive, Terminated), Joining Date etc.).
- Retail chain Hierarchy information (Employee-ID, Manager-ID, Department-ID, etc.)
- Benefits Information (Casual, Sick, Bereavement Leaves, Over-time, Per-Diem, Weekend, Night Shift Allowances, Performance, Safety Bonuses).
- Employee Task Assignment Information (Task ID, Task Name, Task rate, Task Type etc. (some employees are paid per task not hourly)).
- Scheduling Information (Schedule related information like whether the work schedule is 3-day/4-day/5-dy-/6-day a week for the given employee, start time and end time, start day of the schedule, end day of the schedule).
- Login Tracking Information (hours of service for all the employees).
- Approval Information (Provides the approval status of timesheets & timesheet information for payroll processing).
- Miscellaneous Expense Information (Travel Costs, Lodging & Stay costs, Cell-Phone charges etc.)

The data is received from various systems into the databricks data lake in the following ways:

- CDC/Full snapshot data from databases will be pulled into databricks data lake volumes using Amazon Data Migration Service or Databricks Connectors on a given schedule or gets triggered on availability of CDC/Full snapshot as per the configuration and the data will be stored in parquet format.
- Some source systems will push the data into databricks volume as per the schedule in various types of formats like JSON, CSV, Excel, XML, ORC etc.
- Data from some source systems will be pulled into databricks data lake using python/power shell scripts by calling the source system REST API and stored in formats like JSON, CSV etc.

Following are the various phases in the payroll processing system:

- **Data Extraction Phase**: During this phase data from various systems will be extracted in various ways as specified above into the databricks data lake volumes and stored in the native format. Except parquet & ORC formats, all other formats doesn't have schema defined.
- **Data Ingestion Phase**: During this phase data will be loaded into the Spark data frames as spark can read various file formats including excel (needs to install com.crealytics.spark.excel plugin which is available in the form of jar files and all other formats are natively supported). Once the data is loaded into the spark data frames it is subjected to cleansing, filtering and augmenting. Also, schema is attached to the data using configurable schema.json files which doesn't have schema inherently with them. Data will be ultimately stored in databricks unity catalog tables.
- **Data Transformation Phase**: By this phase all the data required for applying the business rules will be available and during this phase data is transformed and aggregated using the business rules and the outputs will be again stored as separate UC tables which will be in ready to consume state.
- **Data Export Phase**: During this phase the **transformed data**(Timesheet data and Employee Hierarchy data) will be exported to Salesforce using Salesforce Bulk API 2.0(which will be discussed later) or **Payroll data** (Approved timesheet data from Salesforce will be extracted into Databricks data lake in CSV format and will be subjected to Ingestion and Transformation Phases) which will be exported into Payroll system using the standard REST API call.



Jobs Description:

The jobs related Data Ingestion, Data Transformation & Data Export phases are executed in Databricks whereas the Data Extraction phase related jobs are executed outside of databricks. To orchestrate the entire data pipeline, we use orchestration tools like Apache Airflow, ActiveBatch or BMC software. There are 3 types of jobs executed in the entire Payroll Processing system workflow.

- **Bi-Daily ActiveBatch/Airflow Job**: This job executes daily twice which prepares the timesheet dataset and exports to Salesforce to kick-off the approval workflow of timesheets
- **Weekly Job**: This job ensures all the timesheets are approved and sends the reminders to approvers to approve the pending approval timesheets and escalates to their superiors if not worked on.
- **Payroll Processing Weekly Job**: This Job prepares the payroll file after all the business validations & approvals. Sends it to the Payroll system to generate Paystubs and for direct deposits to Employee accounts.

Export to Salesforce Process:

- **Step 1** : Need to get authenticated with Salesforce Connected App by providing following details in REST API post request payload at the authentication Url which ends with "services/oauth2/token":

```
payload ={
        'grant_type': 'password',
'client_id': client_id,
'client_secret': client_secret,
'username': username,
'password': password + security_token(optional)
}
```

On successful authentication will receive access_token and instance_url in the response.

- **Step 2**: After successful authentication, Before export to the salesforce the record count and size of the dataset to be exported is determined. If the record count < 2000 records then export using synchronous post API call by providing following details using the access toke and instance url from above authentication response:

- API_VERSION = 'vxx.x' (Use your own desired API version)
- OBJECT_NAME = 'Contact' (can be any standard object or custom object. Here Contact is used for example)
- API_ENDPOINT = f"{instance_url}/services/data/{API_VERSION}/sobjects/{OBJECT_NAME}/
- headers = {
  'Authorization': f'Bearer {access_token}',
'Content-Type': 'application/json',
'Accept': 'application/json'
 }

The request payload should be in JSON format

If the record count > 2000 and recordset size < 150 MB then we need to initiate an asynchronous call which involves following steps:

- **create_bulk_job**(post request)
- BULK_API_URL = f"{INSTANCE_URL}/services/data/{API_VERSION}/jobs/ingest"
- headers = { 'Authorization': f'Bearer {ACCESS_TOKEN}', 'Content-Type': 'application/json', 'Accept': 'application/json' }
- job_payload = { "operation": operation, (either insert or upsert)

```
"object": object_name,(Name of the object like Account, Contract or TransactionJournal or any other  custom
        object
"contentType": contentType,
"externalIdFieldName": "Id" # Optional: Define if you are upserting
 }
```

- The response object contains the job_id which will be used in subsequent steps
- **upload_data_to_job**(put request)
- upload_url = f"{BULK_API_URL}/{job_id}/batches"

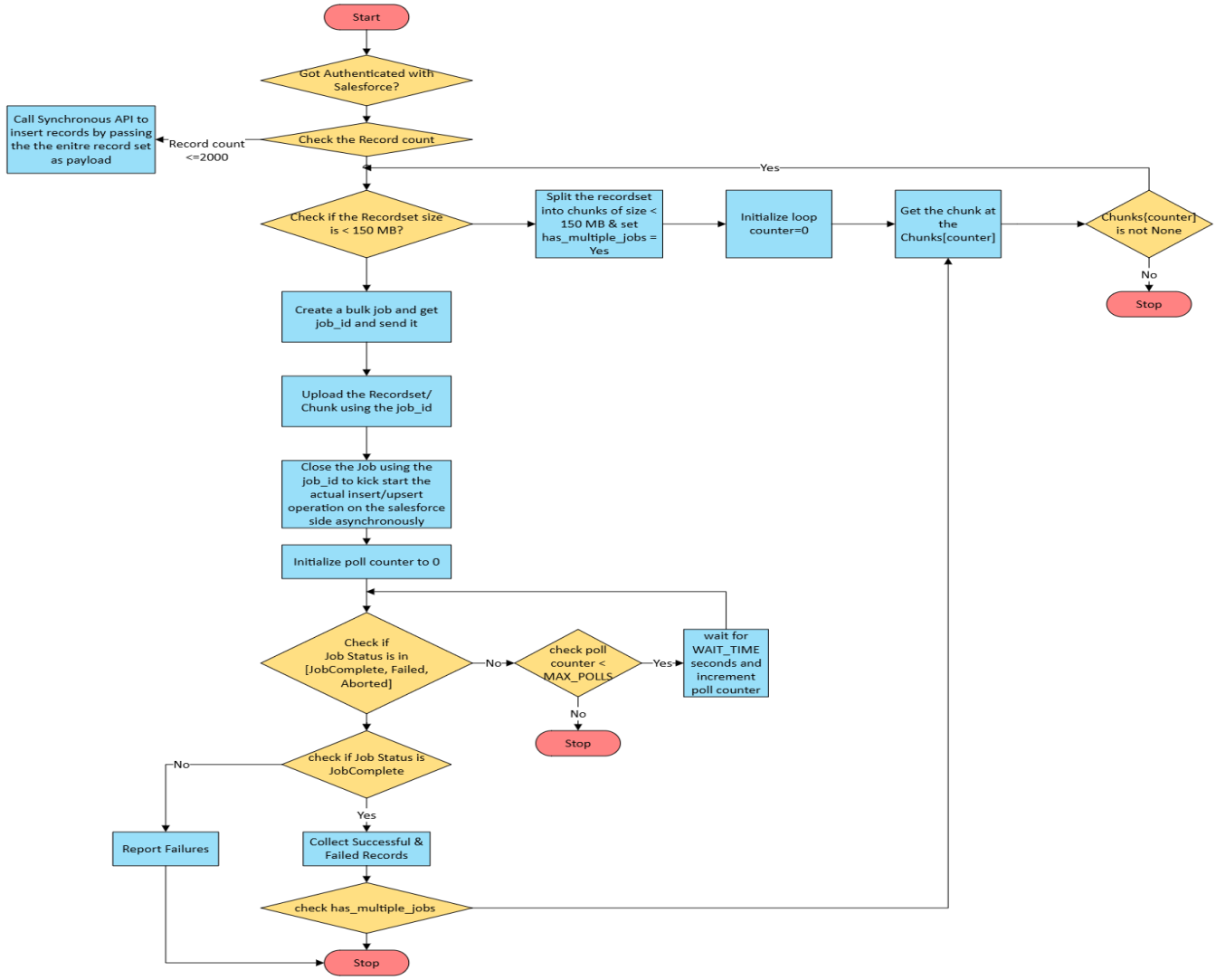
- headers = { 'Authorization': f'Bearer {ACCESS_TOKEN}', 'Content-Type': 'text/csv', }(only csv data can be passed for API 2.0)
- **close_job**(patch request)
- Sends a PATCH request to set the job state to 'UploadComplete'
- job_url = f"{INSTANCE_URL}/services/data/{API_VERSION}/jobs/ingest/{job_id}"
- headers = { 'Authorization': f'Bearer {ACCESS_TOKEN}', 'Content-Type': 'application/json', 'Accept': 'application/json' }
- payload = { "state": "UploadComplete" }

After the close_job call is finished then insert/upsert is initiated asynchronously on the Salesforce side.

- **check_job_status**: to monitor the job status
- **collect_results**: to collect the successful record salesforce ids and failed record salesforce_ids

If the record set size > 150 MB then multiple asynchronous calls need to be initiated but only 150 million records can be initiated in a day.

Below is the flow chart explaining the entire Salesforce Bulk 2.0 API call



CONCLUSION

The above solution is the Payroll Processing system created with minimum investment with all the heavy lifting happening in the Databricks environment and utilizing the out of the box features on Salesforce.

REFERENCES:

1. AWS Documentation for DMS: What is AWS Database Migration Service? - AWS Database Migration Service
2. Databricks Documentation: What is a Medallion Architecture?
3. Salesforce Documentation: Bulk API 2.0 | Bulk API 2.0 and Bulk API Developer Guide | Salesforce Developers.