TO-BE Architecture

One way to design a sustainable datalake for HPE Finace is through creating a generic, data-agnostic table that imports data from different resources and for variety of purposes. We develop automated reports from this data table. These reports will be used by analysts and manafers in Excel, SQL, Spreadsheets, Visualization Software, etc.

For this purpose, we go back to the accounting discipline that implement a very versatile concepts: "Accounts".

Data Structure

The following example illustrates a typical and generic transaction.

Example Transaction:

> "on 11/12/17, bill #123 shows Andy spent 20 euro (from cash) for 100 boxes chocolate."

The Accounting Rule:

> Each bill must have zero balance.

When a new record of financial data comes in, we dispatch it to the appropriate account. Summarization and aggregation for reporting will be done on account groups, automatically and periodically.

STRUCTS: Programmer Point Of View

Data Types from the programmer point of view has the most flexibility in definition.

Data Types

- type Icode int
  Represents numeric code
- type Bcode string
  Represents alpahnumeric code
- type Transaction struct {
  - TS datetime
  - Bill *Icode
  - Account *Account
  - Value float
  - Currency Currency
  - Quantity float
  - Unit string
  - Item string
  - Note string
  - ... }
- type Account struct {
  - Name string

    - Level int
    - BusinessAreas []*Account
    - ProfitCenter []*Account
    - CostCenter []*Account
    - Owner []*Account }
- Type Exchange struct {
    - Currency Currency
    - Date date }

Variables

- var transactions []Transaction
- var accounts map[Bcode]Account
- var conversions map[Exchange]float
- var items map[Icode]string
- var bills map[Icode]string

NoSQL: Big table Point Of View

If we want to store all data in a big (wide) table style (similar to HBase), the following shows the big-table elements:

- Row_Key: BA.PC.PnL.Owner
  Each row represnets one accounting bucket similar to As-Is BW table records.
- Column_Family: {cost, discount, revenue, promotion, return, void, info, Monthly, ...}
  Each family shows the account type
- Column Table <Name, Time, Values> for each Column Family:
  -- Cost Table: <Account, Time, Item, Value, currency>
  example: 1234, today, travel, 1000, euro
  -- Info Table: <Type, level, parents>
  example: Business Area, 3, [HPE, HPS, Fin] -- Monthly Summary Table: <Header, Time, Value>
  example: Total, June 17, $200000

The revenue, discount, promotion, return, ... are similar to Cost Table
The Daily, Yearly, Quarterly, ... Columns are similar to Monthly Summary

When recording into this table (the Ledger), each entry dispatches to its place in the big table. Periodically it will be aggregated and written into monthly columns.

In this way we have detailed data, aggregated data by time for each "defined" account, which covers most of reports that are needed. No other table is required.

SQL: Relational Point of View