PECAN STREET

Water Data Integration Toolkit

Recommended Data Standardization Guidelines & Process Manual

Executive Summary

With the advent and proliferation of smart water meter systems throughout the US, utilities will find that as they adopt these systems, they will need to be able to deal with data in a way they never had to before. The amount of data collected in six hours using a modern AMI system will surpass a year of data collection with a non-smart meter system. Existing methodologies in place at water utilities for data storage need to be updated to account for the influx of data and need to account for the ways in which a utility will be sharing the data with their users, other utilities, and commercial partners. This document should serve as a primer to assist in the creation of data standardization guidelines.

To best assist water utilities with implementation of the recommended Data Standardization Guidelines and Data Schema, Pecan Street has developed the following Data Integration Process Manual. The manual is intended to streamline the data integration process and assist in avoiding common data integration and database management mistakes.

There are three important phases to consider and plan for when attempting to process and integrate data:

1. **The collection phase**: where data in initially stored by the utility;
2. **The storage phase**: when the utility chooses a storage schema for data; and,
3. **The export phase**: when the utility extracts data from an existing data store for integration.

Each of the phases plays an important role in the success of an overall management approach. This manual is organized by phase to serve as a planning guide and reference manual.

Phase 1: Data Collection

When planning for integration of existing and new data streams, consideration must be made when developing the data collection plan. A data collection plan should include, at minimum, the following components:

- Mapping of existing datasets
- Mapping of planned future datasets
- Data import rules
- Data entry rules
- Data naming and convention rules

- Edge cases rules

When developing the initial schema, one should make sure that as data is collected, the methods for input match the designed schema. The less business logic necessary to import the data that has been collected into the datastore, the less error-prone and erratic the data behavior and matching. However, it is much easier to clean data before it is inserted into the database than it is to clean data in an existing database. A plan should be developed to handle edge cases including names with varying length, accents, spaces, apostrophes, or foreign characters.

After the development of the data import rules, the rules should allow for easy data entry which adheres to the general schema. If information is entered online, give helpful error messages such as "No accents are allowed in names." By performing these steps, it easy for people to input information that follows data clarity standards. Also, having the original source modify their input also prevents potential errors in the translation of data like turning accents into apostrophes between characters.

Finally, remember that naming and addressing conventions vary regionally. If operating across a diverse geographical area, make sure that rules that are applicable to all service locations for data comparability in the future.

Phase 2: Data Storage

There are currently many buzzwords around data – data lakes, data warehouses, relational databases, columnar databases, NoSQL, among others. While care should be taken to evaluate the needs and uses for collected data in order to determine the optimized database and data storage platform, Pecan Street has found that SQL-based relational databases are often the best choice. Relational databases have the advantage of being the oldest commonly-used computerized information storage systems, so they tend to be battle-tested and have support available. Recently, relational databases have adopted a lot of new features from other kinds of datastores so there is no longer a big tradeoff between flexibility and reliability that historically surrounded relational databases. Also, a relational database is best because of the highly structured nature of the data itself. The highly structured nature allows for faster execution, better organization, and easier maintenance of the system under a relational database schema.

As data is stored within the database system, consideration should be provided for the reduction of information, while still retaining the data. This process is known as data normalization. For example, in a table that tracks usage:

customer_id	first_name	last_name	phone	usage_date	bill_date	usage

It can be seen that the customer_id along with indentifying and contact might be redundant, as that information might already be stored in a billing information table. The table definition could be simplified as such:

customer_id	usage_date	bill_date	usage

This is a simpler schema and duplicate data is not being entered and stored with each data entry record. Not only does this reduce storage, it also makes it much easier to update the customer's information. If a phone number were to change, it would only be necessary to update the phone number in one place, rather than throughout the entire billing table or other non-normalized table. It is also very easy to reintegrate the normalized data, should the need arise.

Data labeling and cleansing is also a key aspect of data storage. In the example schema above, usage_date is ambiguous. If at all possible, the initial data storage planning should prioritize avoiding confusion about what each piece of data is. Continuing the example above, it might be more helpful if the schema was instead:

customer_id	billing_period_start	billing_period_end	bill_sent_date	usage_amount	bill_total

This is also where data normalization comes back into play. Only bill_total is listed above. Realistically, there is additional information that is sent to the customer on their bill. To store that data, a bill details table should be created that is similar to the following:

bill_id	charge_amt	explanation

In this example, the explanation field should be a text field where a charge description could be listed. It would also be wise to amend our billing information table so it includes this new bill_id field to the tie the tables together like:

customer_id	bill_id	billing_period_start	…

As data is continuously collected, it might prove to be advantageous that you should desire to add customer_id or bill_sent_date to the bill detail table to make it easier to find that information without querying multiple tables.

As always, utilities will want to remain cognizant that data collected might contain Personally Identifiable Information (PII). If the database has billing information that contains PII, it is important to ensure that access to that database is then restricted to individuals with clearance to view PII.

A few simple steps that can be taken to ensure customer privacy when PII data is included in a data table or database. First, encrypt the PII datastore. This will ensure that even if an outside intruder gets access to the computer where the PII data is stored, they will be unable to see any of the information that it contains. Turn on access logs and audit them regularly. By logging how the database is used and performing regular checks on that, the organization can make sure no employees are improperly searching for customer information. Lastly, aim to keep the database off the internet, if possible. This is harder and harder, especially as cloud computing becomes more common. The more difficult it is to access the database, the less likely it is someone will be able to steal critical or sensitive data.

Phase 3: Data Export (Extraction and Retention)

Once data is stored in a database, there are many activities that can be applied to the existing data to significantly increase its usefulness to the organization, including visualizing data, exporting data, or even linking the data store to customer support systems for the ability to quickly cross reference and join information.

Computers are appliances, and like all appliances they eventually break down. In proper planning, hardware failure should not cause data loss. When creating a data plan, be sure to budget for backups. In general, best practice is to have one backup copy of your data on-site for immediate data availability, and for another backup off-site in case there is a physical disaster at the primary location. It is also advisable to consider a fourth backup that is done at regular intervals and is not connected to the other backups, so if something were to corrupt the database and the backups, the data could be restored with minimal loss. This is important when planning for resiliency to ransomware. Finally, it is important to test backups regularly. A testing schedule that evaluates backups each quarter is recommended. By testing backup systems, this allows a check that backups are working properly and provides time to fix backup schedules and plans if that is not the case, before a critical event takes place that forces a test of the system.