Technical Considerations

Assumptions

Every data owner not using the multi tenant node will host their own harmonized data store.

The database meets the standards defined by the community based on participation in use cases. (Some use cases use different databases than others)

All data providers implement the view layer so that all sql that uses them would get the same results with the same data.

RDS vs NOSQL

The intent is for openIDL to be serviced and maintained by standard and junior resources. The majority of data oriented developers have worked with ANSI SQL for multiple years; with an eye on staffing a Relational Data Store (RDS) is recommended. A graph database can be loaded from RDS at a later date if relationships and interconnectivity of data elements becomes a strict requirement. ANSI SQL will satisfy all Regulatory Reporting use cases in an affordable and manageable way.

RDS Options

A full list of DBs can be found: https://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

Top candidates are Postgres, MariaDB, MySQL, SQL Server, Oracle.

Top candidate for Reference data is Postgres.

Minimum ANSI Requirements

SQL operations cheatsheet, James Madison: http://www.qa76.net/sql_analytics

ANSI HISTORY: https://blog.ansi.org/2018/10/sql-standard-iso-iec-9075-2016-ansi-x3-135/#gref

ANSI SQL-92 (MIN): https://en.wikipedia.org/wiki/SQL-92; Advanced Data Types, Case statements, BASIC SQL

ANSI SQL-2003 (Recommended): https://en.wikipedia.org/wiki/SQL:2003; SQL with window functions

ANSI SQL-2016 (Most): https://en.wikipedia.org/wiki/SQL:2016 ; SQL with JSON, prone to security and governance risk.

We will Develop a ruleset that defines what commands can be used. A list of key databases must be supported.

Modeling Considerations

HDS will be a multi layer cube (KS a multi layer cube? Isn't it just relational?) optimized for error free loading. The model will also provide business level views to assist with Business Requirements.

This will be archived by utilizing a raw model, and materialized views will be built on top of raw tables for Business Users (BU)s.

Model Layers

1. Physical Tables.
    a. Staging Tables
    b. Raw Tables
    c. Transaction JSON object.
    d. Rarely change
    e. Mix of concrete and abstract constructs to allow the schema to support new attributes easily
    f. Is under the control of the database owner.
    g. A reference implementation is provided.
    h. Must support the standing view layer.
2. Standing View Layer
    a. Views to bring raw to logical
    b. Views to allow for BU analysis with less joins
    c. More will be added with time, less governance on adding.
    d. Represents the standard data model agreed by the community.
    e. A reference implementation is provided.

    f. If the physical tables don't match the reference model, then the database owner must implement these views.
    g. A verification test bed is provided to ensure the view layer meets expectations.
    h. Materialized as needed for performance
    i. Persistent from time of creation onwards until removed.
3. Dynamic View Layer
    a. Support Extraction Patterns that occur multiple times
    b. May represent standing extractions, like stat reporting
    c. Build and Tear Down in course of Extraction Transaction
    d. Part of the standard model.
4. Query Layer
    a. An SQL processor with

Reference Tables

A collection of tables holding reference data.

Queries use these to augment data.

States are an example.

Allows data to have less redundancy.

Governance/Maintenance

Since the models must change overtime, there must be governance and service level agreements.

Since the physical model is owned by the database owner, it is updated on the owners cadence.

The physical model uses abstract techniques to make it possible to support logical model changes without requiring ddl changes.

The standing view layer changes whenever the standard changes. This will require some maintenance in the dbms. This should be ok, since the etl that loads the data will have to change as well.

The community will define SLAs for data retention and availability.