OSS Database (OSSDB)

This section describes the database for the "hello world" phase of the OSS project.

The database stores observational data sent from the VLA and GBT sites.

The OSS API software is the only software interacting with the database and, thus, is responsible for loading and retrieving all data into/from the database.

A PostgreSQL server running on the test computer c3po is hosting the database. The computer is located at the DSOC.

After the development and test phase, the database will move to the production computer (TBD). (chewbacca, turing?)

The **database** is named ***oss***. *

It has two **usernames**: ***oss***, which has read-write permissions, and ***oss_ro***, which has read-only permission.

*A script named *create-database*, developed and maintained by the SSA group to standardize NRAO databases, was used to initialize and create the database and usernames.

Schemas

- Latest version: OSSDB schema json v0.1.pdf
- Original version: OSSDB schema original v0.2.pdf

Table(s)

Currently, the database uses a single **table** named ***oss_data*** for storing data. A second table for storing site data is under consideration.

Table: oss_data

The table has three **columns**: ***id***, ***created_utc***, and ***site_data***.

- ***id*** is an auto-incrementing value of type SERIAL and the primary key for the table. (4-bytes, allows for 2B+ rows)
- ***created_utc*** is an auto-generated TIMESTAMP value representing the creation date/time of the row.
- ***site_data*** is a field of type JSONB for storing the JSON-formatted data sent from each site.

Sample data

The data stored in the *site_data* column are JSON documents sent from the VLA and GBT sites. Here's a sample from the VLA:

{"notes": "inAdv:True", "obs_id": "23A-214.sb44109940", "src_id": "J1415+1320", "src_ra": 213.99507291249998, "site_id": "vla", "src_dec": 13.339920166666666, "site_lat": 34.07874917, "site_lon": -107.6177275, "site_elev": 2124.0, "freq_lower": 4000000000.0, "freq_upper": 8000000000.0, "src_end_utc": "2023-07-21T05:22:56.000393", "trk_rate_ra": 0.0, "trk_rate_dec": 0.0, "src_start_utc": "2023-07-21T05:21:16.000939"}

Table Design Considerations

- Initially, the plan was to structure *oss_data* as a typical relational database table with one column for each data item. Later, we decided to leverage the JSON support in Postgres and store the JSON document rather than deconstructing the document and storing the data in separate columns. This approach allows us to take advantage of the flexibility and data structures of JSON and avoid the rigidity of a standard database table.
- JSON support in Postgres provides the data types JSON and JSONB. The JSON data type stores a JSON document as text, maintaining white space and the order of keys, but is less efficient during queries as it requires re-parsing the document each time. The JSONB data type, on the other hand, stores each JSON key in a query-efficient binary format and allows for indexing on individual keys of the JSON document. The ability to create indexes on specific JSON keys is the primary reason for selecting JSONB.

Indexes

Index on the JSON key ***site_data->>src_end_utc***:

- Creating the index required the addition of an IMMUTABLE user-defined function to convert the JSON field from a TEXT value to a TIMESTAMP. The function is named *text_to_timestamp*.
- An SQL statement must include the function to use the index.
- An example SQL statement using the function:
    - SELECT * FROM oss_data WHERE text_to_timestamp(site_data->>'src_end_utc') > now();
- Using EXPLAIN ANALYZE shows the improvement in execution time gained by using the index.
    - Query (without index): SELECT * from oss_data WHERE (site_data->>'src_end_utc')::timestamp > now();
    - Result:
        - Seq Scan on oss_data (cost=0.00..2775.74 rows=10989 width=453) (actual time=56.409..56.421 rows=8 loops=1)
          Filter: (((site_data ->> 'src_end_utc'::text))::timestamp without time zone > now())
          Rows Removed by Filter: 32930
        Planning time: 0.732 ms
        Execution time: 56.470 ms
    - Query (with index): SELECT * FROM oss_data WHERE text_to_timestamp(site_data->>'src_end_utc') > now();

- Result:
  - Index Scan using oss_data_text_to_timestamp_idx on oss_data (cost=0.29..8.35 rows=3 width=453) (actual time=0.044..0.046 rows=6 loops=1)
    Index Cond: (text_to_timestamp((site_data ->> 'src_end_utc'::text)) > now())
    Planning time: 0.512 ms
    Execution time: 0.078 ms

Table: site_data

(TBD)

Backup and Recovery

To be addressed after the "hello world" phase.