Exploring Postgres Internals: Takeaways

Syntax

- Getting all the tables within the public schema of a Postgres database:

```
conn = psycopg2.connect(dbname='dq', user='hud_admin', password='eRqg123EEkl')
cur = conn.cursor()
cur.execute("""
    SELECT table_name FROM information_schema.tables
    WHERE table_schema='public' ORDER BY table_name
""")
```

- Converting the name of a table to a Postgres string:

```
from psycopg2.extensions import AsIs
table_name = "state_info"
proper_interpolation = cur.mogrify("SELECT * FROM %s LIMIT 0", [AsIs(table_name)])
cur_execute(proper_interpolation)
```

- Converting a dictionary into a JSON string:

```
import json
d = {'int': 1, 'list': [1, 2, 3], 'dictionary': {'k': 1}}
json_string = json.dumps(d, indent=4)
```

- Converting a JSON string into a dictionary:

```
import json
json_string = '{"int": 1, "list": [1, 2, 3], "dictionary": {"k": 1}}'
dictionary = json.loads(json_string)
```

Concepts

- In every Postgres engine, there are a set of internal tables Postgres uses to manage its entire structure. These contain all the information about data, names of tables, and types stored in a Postgres database.
- We can use the `information_schema` table to get a high-level overview of what tables are stored in the database.
- The `information_schema.tables` structure is as follows:

Name	Data Type	Description
table_catalog	sql_identifier	Name of the database that contains the table (always the current database)
table_schema	sql_identifier	Name of the schema that contains the table
table_name	sql_identifier	Name of the table
table_type	character_data	Type of the table: BASE TABLE for a persistent base table (the normal table type), VIEW for a view, FOREIGN TABLE for a foreign table, or LOCAL TEMPORARY for a temporary table
self_referencing_column_name	sql_identifier	Applies to a feature not available in PostgreSQL
reference_generation	character_data	Applies to a feature not available in PostgreSQL
user_defined_type_catalog	sql_identifier	If the table is a typed table, the name of the database that contains the underlying data type (always the current database), else null.
user_defined_type_schema	sql_identifier	If the table is a typed table, the name of the schema that contains the underlying data type, else null.
user_defined_type_name	sql_identifier	If the table is a typed table, the name of the underlying data type, else null.
		YES if the table is insertable into,

is_insertable_into	yes_or_no	NO if not (Base tables are always insertable into, views not necessarily.)
is_typed	yes_or_no	YES if the table is a typed table, NO if not
commit_action	character_data	If the table is a temporary table, then PRESERVE, else null. (The SQL standard defines other commit actions for temporary tables, which are not supported by PostgreSQL.)

- In Postgres, schemas are used as a namespace for tables with the distinct purpose of separating them into isolated groups or sets within a database.
- `AsIs` keeps the valid SQL representation of a non-string quoted instead of converting it.
- Using an internal table, we can accurately map the types for every column in a table.
- The JSON format is an open standard for writing dictionary-like data into a text file that is easy for a human to read.

Resources

- The Information schema
- System catalogs
- pg_type table description
- pg_aggregate table description
- JSON format
- json Python library