Travaria

Geographical and Political Database

Database Design Project

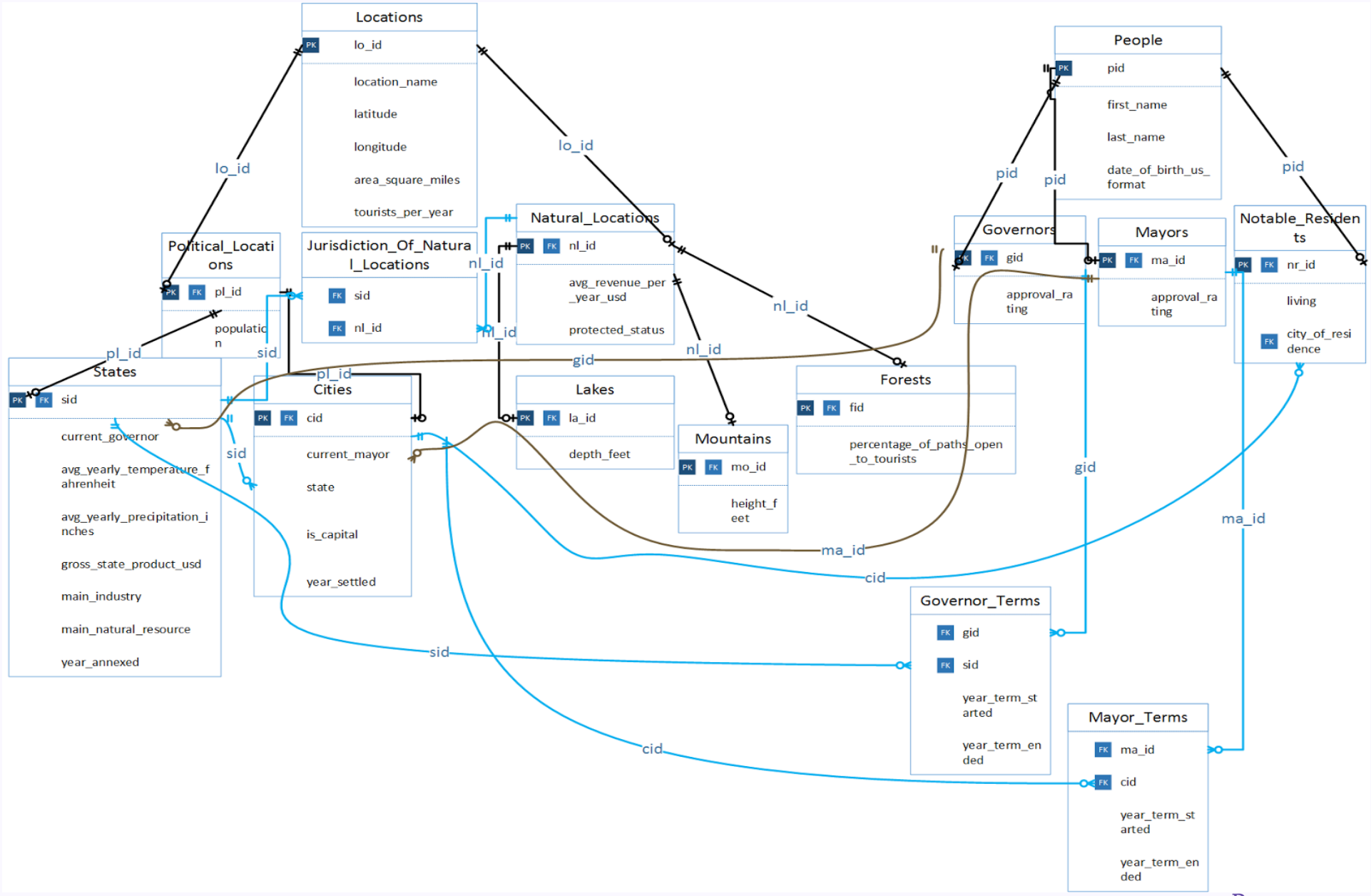
Melissa Iori

Table of Contents

Executive Summary:

Travaria is a fictional country in the midst of an economic crisis. To combat this crisis, the federal government of Travaria has created a plan to organize and document the history, economy and tourism status of the different locations in Travaria. Through the power of relational database design they will be able to identify key sources of revenue and use these results to boost their economy. Also, part of the database will be made available on the Internet for the consumption of citizens and non-citizens alike, promoting education about Travaria's history, leaders and current status.

The objective of this document is to provide a close look at the design and implementation of Travaria's relational database system using PostgreSQL 9.3. This database will be discussed in the context of different user requirements for the final system, as well as the security, referential integrity, and Boyce-Codd Normal Form compliance of the underlying design, which will provide the system with scalability for a long time to come.


Locations
lo_id
location_name
latitude
longitude
area_square_miles
tourists_per_year
People
pid
first_name
last_name
date_of_birth_us_format
Political_Locations
pl_id
population
Jurisdiction_Of_Natural_Locations
sid
nl_id
Natural_Locations
nl_id
avg_revenue_per_year_usd
protected_status
Governors
gid
approval_rating
Mayors
ma_id
approval_rating
Notable_Residents
nr_id
living
city_of_residence
States
sid
current_governor
avg_yearly_temperature_fahrenheit
avg_yearly_precipitation_inches
gross_state_product_usd
main_industry
main_natural_resource
year_annexed
Cities
cid
current_mayor
state
is_capital
year_settled
Lakes
la_id
depth_feet
Mountains
mo_id
height_feet
Forests
fid
percentage_of_paths_open_to_tourists
Governor_Terms
gid
sid
year_term_started
year_term_ended
Mayor_Terms
ma_id
cid
year_term_started
year_term_ended
lo_id
pid
nl_id
pl_id
sid
gid
ma_id
cid

Tables:

Enumerated types:

To sanitize inputs, the government of Travaria has compiled a list of the most common industries and natural resources at play in the country. In PostgreSQL, I created data types so that only the listed values can be accepted as valid industries and natural resources.

```
create type industry as enum('mining', 'timber', 'oil_refinement', 'technology',
'agriculture', 'fishing', 'textiles');
create type natural_resource as enum('gold', 'silver', 'oil', 'wood', 'petroleum', 'salmon',
'soil');
```

People:

The People table keeps track of notable people in Travaria, including governors of states, mayors of cities, and other notable residents of the area. This simple table keeps track of atomic values including the person's first name, last name, and DOB.

```
create table People (
    pid text primary key,
    first_name text,
    last_name text,
    date_of_birth_us_format text
);
```

Functional dependencies: pid → first_name, last_name, date_of_birth_us_format

Governors:

The Governors table tracks all elected governors of states in Travaria - past and present. Each governor had an overall approval percentage rating (0% - 100%). We ensure that the percentage entered won't be above 100. The Governors table is a child to the People table, creating an entity subtype and one-to-one relationship because all Governors are People, but not all People are Governors.

```
create table Governors (
      gid text references People(pid),
      approval_rating int,
      check (approval_rating <= 100),
      primary key(gid)
);
```

Functional dependencies: gid → approval_rating

Mayors:

Highly similar to the Governors table, the Mayors table has essentially the same purpose but instead tracks the mayors of cities. Mayors, too, are People. It is possible to be listed as both a Governor and a Mayor. To prevent update anomalies, information such as first name, last name and date of birth is stored in just one place - the People table, so there is no chance of having one name as a Governor and another name as a Mayor. There will be no question that it is the same person in both roles.

```
create table Mayors (
      ma_id text references People(pid),
```

```
    approval_rating int,
    check (approval_rating <= 100),
    primary key(ma_id)
);
```

Functional dependencies: ma_id → approval_rating

Locations:

The Locations table tracks geographical locations in Travaria. This includes a plain-text name, a latitude and longitude, the area of the place in square miles, and the number of tourists it usually draws in a year. As we will see, it is possible to have smaller locations nested within broader locations. The entity subtype deals with this by storing basic information in this table and breaking down into more detail as we continue along the tree.

```
create table Locations (
    lo_id text primary key,
    location_name text,
    latitude int,
    longitude int,
    area_square_miles int,
    tourists_per_year int,
    unique(latitude, longitude)
);
```

Functional dependencies: lo_id → location_name, latitude, longitude, area_square_miles, tourists_per_year

*(latitude, longitude) was a candidate key, but was composite, not minimal.

Political_Locations:

A politically defined location is-a Location defined by the government and is usually considered a place where People reside, thus the inclusion of the population column.

```
create table Political_Locations (
      pl_id text references Locations(lo_id),
      population int,
      primary key(pl_id)
);
```

Functional dependencies: pl_id → population

Natural_Locations:

A naturally defined location is-a natural Location in Travaria. Unlike a political location, natural Locations are considered as possible tourism opportunities and in some cases, business ventures. The average revenue per year is recorded here. (Travaria's currency is the U.S. Dollar). Some natural Locations are protected from certain developments by the Grace Act of 1920. If that is the case, protected_status is true for that Location.

```
create table Natural_Locations (
      nl_id text references Locations(lo_id),
      avg_revenue_per_year_usd int,
      protected_status boolean default false,
      primary key(nl_id)
);
```

Functional dependencies: nl_id → avg_revenue_per_year_usd, protected_status

States:

States are large areas of land annexed by the government and delegated as states. States contain many cities. A state is also a Political Location, and by transitivity, a Location as well. A state can have one governor at a time, and a governor can't govern two states at once, hence the unique constraint. The foreign key restraint ensures that only valid governors can fill this spot. Some climate information about the state is tracked, including average yearly temperature in Fahrenheit (used in Travaria) and average yearly precipitation levels in inches (the imperial system is used in Travaria).

Gross state product is of type bigint because it is usually in the billions or higher. The GSP is the total Dollar amount of all goods and services in that state, compiled annually. Also recorded is the main industry of that state from an approved list, and its main natural resource. Finally, the year the state was annexed to Travaria is recorded. Travaria was first settled as a country in the year 1777, so states cannot be added before that year.

```
create table States (
      sid text references Political_Locations(pl_id),
      current_governor text references Governors(gid),
      avg_yearly_temperature_fahrenheit int,
      avg_yearly_precipitation_inches int,
      gross_state_product_usd bigint,
      main_industry industry,
      main_natural_resource natural_resource,
      year_annexed int default 1777,
      check (year_annexed >= 1777),
      primary key(sid),
      unique(current_governor)
);
```

Functional dependencies: sid → current_governor, avg_yearly_temperature_fahrenheit, avg_yearly_precipitation_inches, gross_state_product_usd, main_industry, main_natural_resource, year_annexed

Cities:

Cities are Political Locations within states. Like the governors of states, each city has one mayor, and the mayor cannot be mayor to more than one city at a time. Every city is within a state, and the state foreign key ensures that only valid states can be specified. If the city is a capital city, the is_capital attribute will be true. The year settled is very similar to the year annexed, and no year before 1777 can be specified.

```
create table Cities (
      cid text references Political_Locations(pl_id),
      current_mayor text references Mayors(ma_id),
      state text references States(sid),
      is_capital boolean default false,
      year_settled int default 1777,
      check (year_settled >= 1777),
      primary key(cid),
      unique(current_mayor)
);
```

Functional dependencies: cid → current_mayor, state, is_capital, year_settled

Notable_Residents:

Notable Residents are People included in the database who have achieved something great or become famous/infamous in the larger world. This table connects interesting notable residents with some limited data about them, mostly for research purposes. If the 'living' attribute is true, then the person is still living currently and can be reached for interviews, public appearances etc. Otherwise, the resident is deceased, and may become an object of memorial services or studies. Also included is a foreign key link to the city they lived in.

```
create table Notable_Residents (
```

```
    nr_id text references People(pid),
    living boolean default true,
    city_of_residence text references Cities(cid),
    primary key(nr_id)
);
```

Functional dependencies: nr_id → living, city_of_residence

Lakes:

Lakes are Natural Locations (and therefore, Locations). These bodies of water have depth, measured in feet.

```
create table Lakes (
    la_id text references Natural_Locations(nl_id),
    depth_feet int,
    primary key(la_id)
);
```

Functional dependencies: la_id → depth_feet

Forests:

Forests are Natural Locations (and therefore, Locations). The Forests of Travaria can be very large and even treacherous, so only some percentage of the forest paths are open to tourists. This can range from 0% (privately owned) to 100% (all paths open to tourists). This is an important piece of information if there is to be revenue made from any given forest depending on its policies.

```
create table Forests (
    fid text references Natural_Locations(nl_id),
    percentage_of_paths_open_to_tourists int,
```

```
    check (percentage_of_paths_open_to_tourists <= 100),
    primary key(fid)
);
```

Functional dependencies: fid → percentage_of_paths_open_to_tourists

Mountains:

Mountains are Natural Locations (and therefore, Locations). Each Mountain has a height given in feet.

```
create table Mountains (
    mo_id text references Natural_Locations(nl_id),
    height_feet int,
    primary key(mo_id)
);
```

Functional dependencies: mo_id → height_feet

Governor_Terms:

This associative entity allows former governors to be mapped to Travarian states in the interest of history. There must be a start year given for the term, which must be 1777 or later. If the governor is currently in office, the end year can be the predicted end of a four-year appointment. Otherwise, the given end year must be later than the given start year for a term.

This entity lacks an artificial key. However, there does exist a unique composite key - (gid, sid, year_term_started). This ensures a unique term. Governors can, of course, govern a state under multiple terms, so (gid, sid) alone is not unique.

```
create table Governor_Terms (
    gid text references Governors(gid),
    sid text references States(sid),
```

```
        year_term_started int not null,
        year_term_ended int,
        check (year_term_started >= 1777),
        check (year_term_ended >= 1777 and (year_term_ended >= year_term_started or
year_term_ended is NULL))
    );
```

Functional dependencies: (gid, sid, year_term_started) → year_term_ended

Mayor_Terms:

Highly similar to the Governor_Terms table, only with Mayors and Cities. All constraints are the same for Mayors as for Governors.

```
    create table Mayor_Terms (
        ma_id text references Mayors(ma_id),
        cid text references Cities(cid),
        year_term_started int not null,
        year_term_ended int,
        check (year_term_started >= 1777),
        check (year_term_ended >= 1777 and (year_term_ended >= year_term_started or
year_term_ended is NULL)
    );
```

Functional dependencies: (ma_id, cid, year_term_started) → year_term_ended

Jurisdiction_Of_Natural_Locations:

Because Natural Locations such as Lakes, Forests, and Mountains are very large in area and often span across multiple states, this artificial entity was created to map Natural Locations to the states they are located in. This allows for Natural Locations that are contained within one state, as well as multiple states. (sid, nl_id) is a unique key.

```
create table Jurisdiction_Of_Natural_Locations (
     sid text references States(sid),
     nl_id text references Natural_Locations(nl_id)
);
```

Functional dependencies: None.

Population of Test Data:

```
insert into People (pid, first_name, last_name, date_of_birth_us_format)
values ('p01', 'Glen', 'Forester', '10/20/1950'),
        ('p02', 'Esther', 'Segwin', '05/06/1965'),
        ('p03', 'Alex', 'Grace', '01/07/1890'),
        ('p04', 'Lillith', 'Hall', '06/23/1960'),
        ('p05', 'Steven', 'Jobs', '02/24/1955'),
        ('p06', 'Louis', 'McVay', '11/10/1947');

insert into Governors (gid, approval_rating)
values ('p01', 90),
       ('p06', 85);

insert into Mayors (ma_id, approval_rating)
values ('p02', 97),
       ('p06', 94);
```

```
     insert into Locations (lo_id, location_name, latitude, longitude, area_square_miles,
tourists_per_year)
     values ('l01', 'New Opalshire', 220, 291, 50000, 87000000),
             ('l02', 'Travopolis', 304, 178, 200, 9000000),
             ('l03', 'Lunar Lake', 304, 123, 100, 1000000),
             ('l04', 'Grace Forest', 330, 167, 500, 500000),
             ('l05', 'Mount Gulden', 203, 90, 200, 1000000),
             ('l06', 'Jasmia', 302, 66, 30000, 60000000);

     insert into Political_Locations (pl_id, population)
     values ('l01', 20000000),
          ('l02', 4000000),
          ('l06', 10000000);

     insert into Natural_Locations (nl_id, avg_revenue_per_year_usd, protected_status)
     values ('l03', 40000000, false),
             ('l04', 10000000, true),
             ('l05', 32000000, true);

     insert into States (sid, current_governor, avg_yearly_temperature_fahrenheit,
avg_yearly_precipitation_inches, gross_state_product_usd, main_industry, main_natural_resource,
year_annexed)
     values ('l01', 'p01', 67, 133, 4000000000000, 'fishing', 'salmon', 1882),
          ('l06', 'p06', 62, 120, 4600000000000, 'mining', 'gold', 1890);

     insert into Cities (cid, current_mayor, state, is_capital, year_settled)
     values ('l02', 'p02', 'l01', true, 1892);
```

```
insert into Notable_Residents (nr_id, living, city_of_residence)
values ('p03', false, 'l02'),
        ('p04', true, 'l02'),
        ('p05', false, 'l02');

insert into Lakes (la_id, depth_feet)
values ('l03', 900);

insert into Forests (fid, percentage_of_paths_open_to_tourists)
values ('l04', 80);

insert into Mountains (mo_id, height_feet)
values ('l05', 19000);

insert into Governor_Terms (gid, sid, year_term_started, year_term_ended)
values ('p01', 'l01', 2008, 2012),
     ('p06', 'l01', 2004, 2008);

insert into Mayor_Terms (ma_id, cid, year_term_started, year_term_ended)
values ('p02', 'l02', 2008, 2012),
     ('p06', 'l02', 2004, 2008);

insert into Jurisdiction_Of_Natural_Locations (sid, nl_id)
values ('l01', 'l03'),
        ('l01', 'l04'),
        ('l01', 'l05'),
        ('l06', 'l04');
```

Views

```
create view Politicians_As_Governors_and_Mayors as
    select first_name as First_Name,
        last_name as Last_Name,
        g.approval_rating as Governor_Approval_Rating,
        m.approval_rating as Mayor_Approval_Rating
    from People p, Governors g, Mayors m
    where p.pid = g.gid and p.pid = m.ma_id;
```

This view returns the names and approval ratings of People who have been both Governors and Mayors at some point.

	first_name text	last_name text	governor_approval_rating integer	mayor_approval_rating integer
1	Louis	McVay	85	94

Views

```
create view Climate as
     select l.location_name as state,
     avg(s.avg_yearly_temperature_fahrenheit) as temp,
     avg(s.avg_yearly_precipitation_inches) as precipitation
     from States s, Locations l
     where s.sid = l.lo_id
     group by state
     order by state;
```

This view gets climate data for all states.

	state text	temp numeric	precipitation numeric
1	Jasmia	62.0000000000000000	120.0000000000000000
2	New Opalshire	67.0000000000000000	133.0000000000000000

Reports

```
select l.location_name as Location_Name,
       (nl.avg_revenue_per_year_usd / l.tourists_per_year) as
Revenue_Per_Tourist_USD
       from Locations l, Natural_Locations nl
       where l.lo_id = nl.nl_id
       order by Revenue_Per_Tourist_USD desc;
```

This view shows which natural locations are gaining the most revenue per tourist.

	location_name text	revenue_per_tourist_usd integer
1	Lunar Lake	40
2	Mount Gulden	32
3	Grace Forest	20

Reports

```
select l.location_name as Location_Name,
       (s.gross_state_product_usd / pl.population) as GSP_Per_Capita
       from Locations l, Political_Locations pl, States s
       where l.lo_id = pl.pl_id and pl.pl_id = s.sid
       order by GSP_Per_Capita desc;
```

This query report shows the GSP (Gross State Product) per capita for each state, which is simply the total of all goods and services produced in that state over a year divided by the population. It is sometimes used as a quality of life metric.

	location_name text	gsp_per_capita bigint
1	Jasmia	460000
2	New Opalshire	200000

Reports

```
select max(l.location_name) as Location_Name,
    min(l.tourists_per_year) as Tourists_Per_Year
        from Locations l, Political_Locations pl
        where l.lo_id = pl.pl_id
            and l.lo_id in(
                select sid
                from states
            )
        group by pl.population
        having pl.population > 5000000
```

```
        order by Tourists_Per_Year desc;
```

This query report provides the tourists per year for all states that have a population higher than 5 million people.

	location_name text	tourists_per_year integer
1	New Opalshire	87000000
2	Jasmia	60000000

Stored Procedures

```
create or replace function getStatesHostingNaturalLocation(varchar(100),
REFCURSOR) returns refcursor as
$$
    declare
```

```
          NaturalLocation varchar(100) := $1;
          resultset REFCURSOR := $2;
     begin
           open resultset for select forState.location_name as States
          from Locations forState, Locations forNatural,
Jurisdiction_Of_Natural_Locations j
          where j.sid = forState.lo_id and j.nl_id = forNatural.lo_id and
forNatural.location_name = NaturalLocation;
     return resultset;
     end;
$$
language plpgsql;

select getStatesHostingNaturalLocation('Grace Forest', 'results');
Fetch all from results;
```

This stored procedure gets back a set of all states that share the given natural location. In the example, Grace Forest is used. It spans over two states:

	states text
1	New Opalshire
2	Jasmia

Triggers

```
create or replace function check_governor() returns trigger as $check_governor$
    begin
        if
        exists
            (select g.year_term_started
            from Governor_Terms g
            where g.gid = NEW.gid
            and g.year_term_started = NEW.year_term_started)
        or exists
            (select m.year_term_started
            from Mayor_Terms m
            where m.ma_id = NEW.gid
```

```
            and m.year_term_started = NEW.year_term_started)
        then
            raise exception 'This governor has already started a term as
governor or mayor this year. Cannot serve concurrent terms.';
        end if;
        return new;
    end;
$check_governor$ LANGUAGE plpgsql;

create or replace function check_mayor() returns trigger as $check_mayor$
    begin
        if
        exists
            (select g.year_term_started
            from Governor_Terms g
            where g.gid = NEW.ma_id
            and g.year_term_started = NEW.year_term_started)
        or exists
```

```
            (select m.year_term_started
            from Mayor_Terms m
            where m.ma_id = NEW.ma_id
            and m.year_term_started = NEW.year_term_started)
        then
            raise exception 'This mayor has already started a term as governor
or mayor this year. Cannot serve concurrent terms.';
        end if;
        return new;
    end;
$check_mayor$ LANGUAGE plpgsql;

create trigger check_leader before insert on Governor_Terms
For EACH row execute procedure check_governor();

create trigger check_leader before insert on Mayor_Terms
For EACH row execute procedure check_mayor();
```

These triggers check that governors and mayors respectively aren't already serving a term as a governor or mayor. Because gid and ma_id are equivalent keys, content-wise, we can make both checks in the same query. p06, Louis McVay, already served a term in 2004. Now, when we try to make him governor:

```
insert into Governor_Terms (gid, sid, year_term_started, year_term_ended)
values ('p06', 'l01', 2004, 2012);
```

Output pane

Data Output | Explain | **Messages** | History

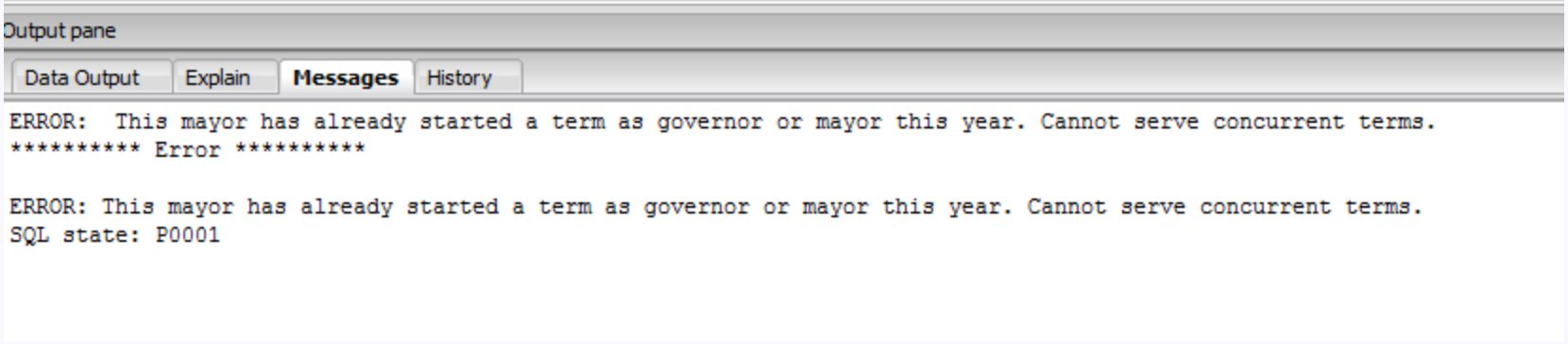
```
ERROR:  This governor has already started a term as governor or mayor this year. Cannot serve concurrent terms.

********** Error **********

ERROR: This governor has already started a term as governor or mayor this year. Cannot serve concurrent terms.
SQL state: P0001
```

Same with the mayor position:

```
insert into Mayor_Terms (ma_id, cid, year_term_started, year_term_ended)
values ('p01', 'l02', 2008, 2012);
```



Security

```
grant select,update on Governors,Mayors to political_analyst;
grant select,update on Locations,Political_Locations,Natural_Locations to
tourism_analyst;
```

Among many possible roles for this database, I provided for two different types of analyst - a political analyst who is interested in the approval ratings of political figures, and a tourism analyst who needs information on tourists and revenue. While they are not permitted to insert new instances, they should be able to keep the database up to date with the latest results of their polls and findings.

```
grant select on Forests to tourism_analyst;
```

The Forests table contains the percentage_of_paths_open_to_tourists column, which may be useful to the tourism analyst in deciding whether a certain forest may be worth pursuing for more tourist revenue.

Implementation Notes / Known Issues

In practical usage, it's important to note that inserting a new Location into the database may require up to three separate INSERT statements.

One into Locations for basic information.

One into either Political_Locations or Natural_Locations, depending.

One into States, Cities, Lakes, Forests, or Mountains, depending.

This supports the entity subtype design but one wonders if there isn't a way to abstract the underlying design from the user for ease of use.

Future Enhancements

- **Ability to dynamically update industry and natural resource types through a stored procedure**
- **More types of Locations and People**
- **Economic views for economic analysts.**