INFO 610 Fall 2020

Week 4

Finally, some Technology!

At this point, we'll be interacting with our postgresql database.

Whether you're running it in Docker or have natively installed, that's OK.

I'll start with a few 'command line' interactions with Postgres, but, primarily I will use SQL Developer.

https://www.oracle.com/database/technologies/appdev/sqldeveloper-landing.html

https://jdbc.postgresql.org/

Remove all stopped docker containers

Run and Interact

psql

In a new terminal (since I didn't run postgres docker as a daemon)

Now we can start

Let's create some tables.

CREATE TABLE

- In my scripts, I start with dropping the table so I can run it over and over.
- If the table DOESN'T exist 'DROP TABLE agents' would fail.
- IF EXISTS protects against that failure
- 'agent_id' is an integer, primary key.
- These columns use two data types
  - varchar, date

INSERT

- Page 14 of design book shows us the data we'll insert
- This statement INSERTS INTO the 'agents' table
- The values in the tuples will be placed into the columents (agent_id, agent_first_name, etc)
- the 'values' provide a list of 'tuples' (rows)

Foreign Key

- REFERENCES sets up the foreign key
- the 'agent_id' field is a foreign key to 'agents (agent_id)'

Failed Insert / Foreign Key

- We never put an agent in the 'agents' table with an 'agent_id' of 1
- When we execute this statement, we get an error:

- We have violated the foreign key constraint.

Sequence

In PostgreSQL, a sequence is a special kind of database object that generates a sequence of integers. A sequence is often used as a primary key column. Similar to the AUTO_INCREMENT concept in MySQL.

When creating a new table, the sequence is created through the SERIAL pseudo-type as follows:

By assigning the SERIAL pseudo-type to the id column, PostgreSQL will perform the following:

- Creates a sequence object and set the next value generated by the sequence as the default value for the column.
- Adds the NOT NULL constraint to the column because a sequence always generates an integer, which is a non-null value.

Serial Types

- smallserial / / bigserial
- Auto-incrementing integer values

smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing	1 to 9223372036854775807

How to create and use a serial field

It is important to note that the SERIAL does not implicitly create an index on the column or make the column as the primary key column. However, this can be done easily by specifying the PRIMARY KEY constraint for the SERIAL column.

1	apple
2	pear
3	pawpaw
4	orange

What does SERIAL really do?

Write out some more SQL

class.sql for reference after class

#

#

#

#