Database name: jmrozek

General description

This code represents a PostgreSQL database dump that contains data related to a business operation. The database includes tables for products, categories, suppliers, shippers, employees, customers, orders, and territories, among others. Each table is set up with a specific schema and constraints for data integrity. The database also includes various indexes and foreign key constraints for optimal data retrieval and relationships between different tables.

Schemas

1. public

Indexes

```
INDEX contact_idx ON public.customers USING btree (contact_name);
```

```
INDEX products_idx ON public.products USING btree (product_name);
```

Tables

1. us_states
2. customers
3. orders
4. employees
5. shippers
6. products
7. order_details
8. categories
9. suppliers
10. region
11. territories
12. employee_territories
13. customer_demographics
14. customer_customer_demo

---

us_states

```
table name = 'us_states'

Column('state_id', SmallInteger, primary_key=True),
Column('state_name', String(100)),
Column('state_abbr', String(2)),
```

```
Column('state_region', String(50))
```

Description

The table is named 'us_states'. It seems to be used to store information about different states in the United States.

The table has four columns: 1. 'state_id' with a data type of SmallInteger. This is the primary key of the table, which means that each row in the table will have a unique 'state_id'. This field is likely used to uniquely identify each state in the database. 2. 'state_name' with a data type of String of maximum length 100. This field is likely used to store the full name of the state. 3. 'state_abbr' with a data type of String of length 2. This field likely stores the two-letter abbreviation for each state, such as 'NY' for New York. 4. 'state_region' with a data type of String of maximum length 50. This field is used to store the region in which the state is located. This could be a geographical region like 'Midwest', or perhaps a political or census-defined region.

The table does not appear to have any defined relationships with other tables, constraints or comments. As such, the information provided is fairly straightforward and doesn't offer additional context or restrictions regarding the data stored in this table.

---

customers

```
table name = 'customers'

Column('customer_id', String(5), primary_key=True),
Column('company_name', String(40), nullable=False),
Column('contact_name', String(30), index=True),
Column('contact_title', String(30)),
Column('address', String(60)),
Column('city', String(15)),
Column('region', String(15)),
Column('postal_code', String(10)),
Column('country', String(15)),
Column('phone', String(24)),
Column('fax', String(24))
```

Description

The 'customers' table appears to store information about customers, likely for a business or organization.

The primary key for this table is 'customer_id', which is a string of up to 5 characters, implying that each customer has a unique identifier that isn't necessarily a numerical value.

'company_name' is a required field that holds a string of up to 40 characters, suggesting that the customers in this table are companies.

The 'contact_name' and 'contact_title' fields likely store the name and job title of the person at each company who should be contacted for business matters. 'contact_name' is indexed, which means that searches and queries using the contact's name will be more efficient.

The 'address', 'city', 'region', 'postal_code', 'country' fields seem to store the full mailing address of the company.

Finally, the 'phone' and 'fax' fields, as their names suggest, probably store the company's phone and fax numbers. Both can store strings of up to 24 characters.

There are no explicit relationships, constraints or comments provided in the provided code, so we cannot comment on any existing relationships with other tables, any additional constraints or any additional information provided in comments.

---

orders

```
table name = 'orders'

Column('order_id', SmallInteger, primary_key=True),
Column('customer_id', ForeignKey('customers.customer_id')),
Column('employee_id', ForeignKey('employees.employee_id')),
Column('order_date', Date),
Column('required_date', Date),
Column('shipped_date', Date),
Column('ship_via', ForeignKey('shippers.shipper_id')),
Column('freight', Float),
Column('ship_name', String(40)),
Column('ship_address', String(60)),
Column('ship_city', String(15)),
Column('ship_region', String(15)),
Column('ship_postal_code', String(10)),
Column('ship_country', String(15))
```

Description

The 'orders' table appears to be used for storing information regarding orders made by customers.

The primary key for the table is 'order_id', a small integer which uniquely identifies each order.

The ‘customer_id’ column references the ‘customers’ table, indicating the customer responsible for each order. Similarly, the ‘employee_id’ column references the ‘employees’ table, likely indicating the employee who processed the order.

The ‘order_date’, ‘required_date’, and ‘shipped_date’ columns are all dates, presumably indicating when the order was made, when it is required to be delivered, and when it was actually shipped, respectively.

The ‘ship_via’ column is a foreign key referencing the ‘shippers’ table, suggesting the method of shipping used for each order.

The ‘freight’ column is a float, and likely represents the cost of shipping for the order.

The ‘ship_name’, ‘ship_address’, ‘ship_city’, ‘ship_region’, ‘ship_postal_code’, and ‘ship_country’ columns are all strings of varying lengths. These presumably represent the name and address to which the order is to be shipped. The ‘ship_name’ might be the name of the recipient, while the remaining fields represent different components of the shipping address.

---

employees

```
table name = 'employees'

Column('employee_id', SmallInteger, primary_key=True),
Column('last_name', String(20), nullable=False),
Column('first_name', String(10), nullable=False),
Column('title', String(30)),
Column('title_of_courtesy', String(25)),
Column('birth_date', Date),
Column('hire_date', Date),
Column('address', String(60)),
Column('city', String(15)),
Column('region', String(15)),
Column('postal_code', String(10)),
Column('country', String(15)),
Column('home_phone', String(24)),
Column('extension', String(4)),
Column('photo', LargeBinary),
Column('notes', Text),
Column('reports_to', ForeignKey('employees.employee_id')),
Column('photo_path', String(255))
```

Description

The table named ‘employees’ appears to be a record of company employees.

The ‘employee_id’ column is the primary key, suggesting each employee has a unique identification number that is small in size.

The ‘last_name’ and ‘first_name’ columns store the employee’s surname and given name respectively, with both fields required to have a value (nullable=False).

The ‘title’ and ‘title_of_courtesy’ provide information regarding the employee’s job title and their formal salutation, respectively.

The ‘birth_date’ and ‘hire_date’ columns record the date of birth and the date when the employee was hired.

The ‘address’, ‘city’, ‘region’, ‘postal_code’, and ‘country’ fields are likely to represent the employee’s home address.

The ‘home_phone’ and ‘extension’ columns presumably store details for contacting the employee at home and their extension number at work.

The ‘photo’ field may store a binary representation of the employee’s photograph, while ‘photo_path’ holds the path to the employee’s photo file.

The ‘notes’ field might be used for any additional information about the employee.

The ‘reports_to’ column appears to establish a hierarchical relationship within the company, with the foreign key referring back to the ‘employee_id’ in the same table, indicating who each employee reports to.

---

shippers

```
table name = 'shippers'

Column('shipper_id', SmallInteger, primary_key=True),
Column('company_name', String(40), nullable=False),
Column('phone', String(24))
```

Description

The ‘shippers’ table seems to be storing information about various shipping companies. This table has three columns: ‘shipper_id’, ‘company_name’, and ‘phone’.

The ‘shipper_id’ column is the primary key for this table, which means that it uniquely identifies each record in the table. The data type of ‘shipper_id’ is SmallInteger, suggesting it’s a numerical value, and it cannot be null, meaning each shipper must have an ID.

The ‘company_name’ column presumably stores the names of the shipping companies. The data type is String(40) implying that the name of the company

cannot exceed 40 characters. This field is also not nullable, demonstrating that each shipper must have a company name.

The 'phone' column might be used to store the phone contact details of the shipping companies. Its data type is String(24), indicating that the phone number can't exceed 24 characters. There's no nullability constraint specified for this column, so it's possible that some shippers may not have a phone number listed.

---

products

```
table name = 'products'

Column('product_id', SmallInteger, primary_key=True),
Column('product_name', String(40), nullable=False, index=True),
Column('supplier_id', ForeignKey('suppliers.supplier_id')),
Column('category_id', ForeignKey('categories.category_id')),
Column('quantity_per_unit', String(20)),
Column('unit_price', Float),
Column('units_in_stock', SmallInteger),
Column('units_on_order', SmallInteger),
Column('reorder_level', SmallInteger),
Column('discontinued', Integer, nullable=False)
```

Description

The 'products' table seems to be a part of an inventory or product management system.

1. 'product_id' column: This is the primary key of the table. It holds a unique identifier for each product item in the table.
2. 'product_name' column: This stores the name of each product. It is a mandatory field as indicated by nullable=False, and an index has been created on it which suggests it is frequently searched or queried.
3. 'supplier_id' column: This seems to be a foreign key that links to a 'suppliers' table. It likely denotes who supplies the particular product.
4. 'category_id' column: This is another foreign key that probably links to a 'categories' table. It can be used to categorize the products.
5. 'quantity_per_unit' column: This field likely represents the quantity of the product per unit. It is a string field perhaps to accommodate different measurement units.
6. 'unit_price' column: This is a floating number likely representing the price for a single unit of the product.

7. 'units_in_stock' column: This is likely to be the current quantity of the product in stock.
8. 'units_on_order' column: This could represent the quantity of the product currently on order.
9. 'reorder_level' column: This might be used to define a threshold to trigger a reorder of the product.
10. 'discontinued' column: This is a mandatory integer field. It's likely used as a boolean to represent whether a product is discontinued (probably 1 for 'Yes' and 0 for 'No').

---

order_details

```
    table name = 'order_details'

    Column('order_id', ForeignKey('orders.order_id'), primary_key=True,
nullable=False),
    Column('product_id', ForeignKey('products.product_id'), primary_key=True,
nullable=False),
    Column('unit_price', Float, nullable=False),
    Column('quantity', SmallInteger, nullable=False),
    Column('discount', Float, nullable=False)
```

Description

The table 'order_details' appears to be a detailing table that provides information about the individual products in each order. It has a relationship with the 'orders' table and the 'products' table through the foreign keys 'order_id' and 'product_id' respectively.

The primary key of the table is a composite key of 'order_id' and 'product_id', meaning each record in the table represents a unique combination of an order and a product.

The 'order_id' column is a foreign key that links to the 'orders' table. It represents the unique identifier of an order and it cannot be null.

The 'product_id' column is also a foreign key that links to the 'products' table. It represents the unique identifier of a product included in the order and this also cannot be null.

The 'unit_price' column specifies the price per unit of the product at the time of the order. It is a floating-point number and it also cannot be null, implying every product in an order must have a price.

The ‘quantity’ column is a small integer that represents the number of units of the product ordered. This column cannot be null, which means that every product in an order must have a quantity.

The ‘discount’ column is a floating-point number that represents any discount applied to the product in the order. This column also cannot be null which indicates that a discount value must always be provided, even if it is zero.

---

categories

```
table name = 'categories'

Column('category_id', SmallInteger, primary_key=True),
Column('category_name', String(15), nullable=False),
Column('description', Text),
Column('picture', LargeBinary)
```

Description

The ‘categories’ table seems to store information related to different categories. It has four columns:

1. ‘category_id’ - This is the primary key of the table, which means it contains unique identifiers for each row of data. It seems to store the unique ID of each category. The data type is SmallInteger, so it can store small range of integer values.
2. ‘category_name’ - This column likely stores the name of each category. It has a constraint of ‘not null’, meaning it must contain a value for each row. The data type is string and can contain up to 15 characters, which implies that the names of the categories are relatively short.
3. ‘description’ - This column likely holds a detailed description of each category. The data type is Text, so it can contain long strings of text.
4. ‘picture’ - This column probably stores binary data for an image or picture related to each category. The data type is LargeBinary, which is used for storing large amounts of binary data such as images or other multimedia.

There are no comments, constraints or relationships mentioned beyond these points.

---

suppliers

```
table name = 'suppliers'
```

```
Column('supplier_id', SmallInteger, primary_key=True),
Column('company_name', String(40), nullable=False),
Column('contact_name', String(30)),
Column('contact_title', String(30)),
Column('address', String(60)),
Column('city', String(15)),
Column('region', String(15)),
Column('postal_code', String(10)),
Column('country', String(15)),
Column('phone', String(24)),
Column('fax', String(24)),
Column('homepage', Text)
```

Description

The table 'suppliers' seems to represent a list of suppliers for a company or business. The primary key for this table is 'supplier_id', which is a unique identifier for each supplier.

The 'company_name' column stores the name of the supplier's company and it is a mandatory field, as indicated by the 'nullable=False' constraint.

The 'contact_name' and 'contact_title' columns likely store the name and job title of the person to contact at the supplier's company.

The 'address', 'city', 'region', 'postal_code', and 'country' columns hold the physical location of the supplier's company.

The 'phone' and 'fax' columns are likely used for contact information, storing the phone and fax numbers of the supplier.

The 'homepage' column probably stores the URL of the supplier's company website.

Overall, this table appears to be used for storing detailed contact and location information about suppliers.

---

region

```
table name = 'region'

Column('region_id', SmallInteger, primary_key=True),
Column('region_description', String(60), nullable=False)
```

Description

The table is named 'region'. It contains information about various regions.

There are two columns in the table:

1. 'region_id': This is the primary key of the table. It contains unique identification numbers for each region. The data type of this column is SmallInteger, meaning it stores small range of integer values. Since it is marked as the primary key, no two rows can have the same 'region_id' value.
2. 'region_description': This column contains descriptions of the regions. The descriptions are stored as strings and can be up to 60 characters long. This information is required, as indicated by 'nullable=False', meaning that every row must have a region description.

There are no constraints, relationships or comments provided within this code.

---

territories

```
table name = 'territories'

Column('territory_id', String(20), primary_key=True),
Column('territory_description', String(60), nullable=False),
Column('region_id', ForeignKey('region.region_id'), nullable=False)
```

Description

The 'territories' table seems to store information related to geographical territories or areas. It contains three columns:

1. 'territory_id': This is the primary key of the table, uniquely identifying each record. It's a string with a maximum length of 20 characters.
2. 'territory_description': This field holds the description of the territory. It's a string with a maximum length of 60 characters. This field cannot be null which implies that every territory should have a description.
3. 'region_id': This is a foreign key that references the 'region_id' in the 'region' table. This suggests a relationship between the 'territories' and 'region' table, where each territory is associated with a specific region. This field cannot be null which means every territory must be associated with a region.

In summary, this table seems to be used for managing geographical areas or territories and their relationship to broader regions. The 'territory_description' and the associated 'region_id' provide detailed information about each territory.

---

employee_territories

```
    table name = 'employee_territories'

    Column('employee_id', ForeignKey('employees.employee_id'), primary_key=True,
nullable=False),
    Column('territory_id', ForeignKey('territories.territory_id'),
primary_key=True, nullable=False)
```

Description

The 'employee_territories' table appears to be a junction table that creates a many-to-many relationship between the 'employees' and 'territories' tables. It consists of two columns: 'employee_id' and 'territory_id', both of which are foreign keys referencing 'employee_id' in the 'employees' table and 'territory_id' in the 'territories' table respectively.

The 'employee_id' column stores the unique identifiers for employees. The 'territory_id' column stores the unique identifiers for territories. Both of these columns form the composite primary key for the 'employee_territories' table, which means that each combination of 'employee_id' and 'territory_id' must be unique.

This table indicates which employee is assigned to which territory. The structure of this table implies that an employee can be associated with multiple territories and a territory can also have multiple employees assigned to it.

The 'nullable=False' constraint suggests that both 'employee_id' and 'territory_id' columns cannot contain NULL values, implying that every record must explicitly assign an employee to a territory.

---

customer_demographics

```
    table name = 'customer_demographics'

    Column('customer_type_id', String(5), primary_key=True),
    Column('customer_desc', Text)
```

Description

The 'customer_demographics' table appears to be a simple table containing information about different types of customers. The table consists of two columns.

The first column is 'customer_type_id', which is likely to be a unique identifier for each customer type. This column is marked as the primary key, which means that it should contain unique values to identify each row in the table. The 'customer_type_id' is of String type and has a maximum length of 5 characters.

The second column is 'customer_desc', which is a Text type and might contain detailed descriptions of each customer type. It seems that this column can contain any form of text, and there is no defined limit on its length.

There are no additional constraints or relationships mentioned for this table. It seems to be a standalone table, and its data might be used in combination with other tables through its primary key.

---

customer_customer_demo

```
    table name = 'customer_customer_demo'

    Column('customer_id', ForeignKey('customers.customer_id'), primary_key=True,
nullable=False),
    Column('customer_type_id',
ForeignKey('customer_demographics.customer_type_id'), primary_key=True,
nullable=False)
```

Description

The table 'customer_customer_demo' appears to be a junction table that creates a many-to-many relationship between the 'customers' table and the 'customer_demographics' table.

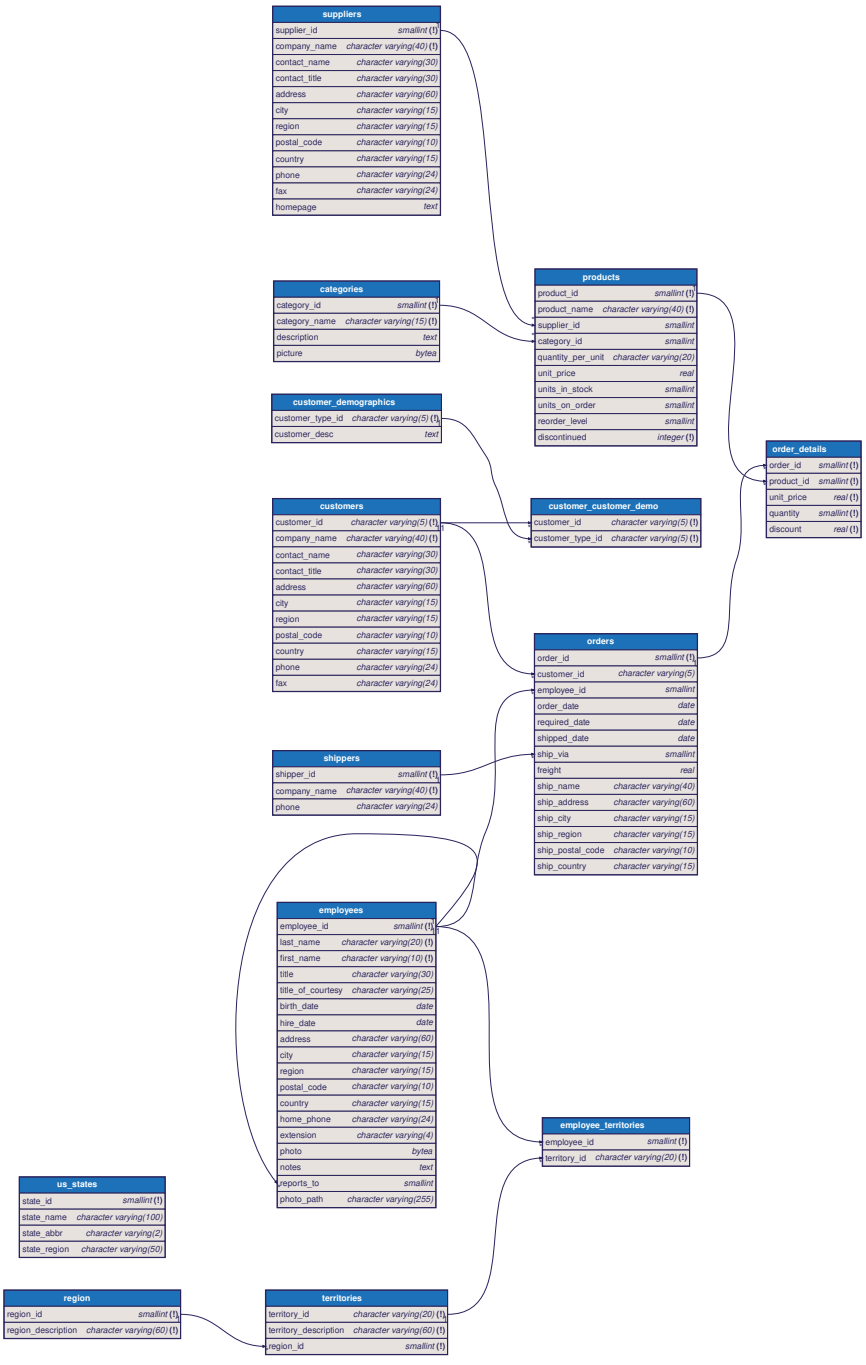
It has two columns: 'customer_id' and 'customer_type_id'. Both of these fields are foreign keys, referencing 'customer_id' from the 'customers' table and 'customer_type_id' from the 'customer_demographics' table respectively.

The 'customer_id' column stores the unique identifiers for customers. This column is a primary key and cannot be null, indicating that each record must be associated with a customer.

Similarly, the 'customer_type_id' column stores the unique identifiers for different customer types. It's also a primary key and cannot be null, which means each record must be associated with a customer type.

The combination of these two primary keys ensures the uniqueness of each record in the 'customer_customer_demo' table, meaning that each customer can be associated with multiple customer types and each customer type can be associated with multiple customers.

Graph


suppliers
supplier_id smallint (1)
company_name character varying(40) (1)
contact_name character varying(30)
contact_title character varying(30)
address character varying(60)
city character varying(15)
region character varying(15)
postal_code character varying(10)
country character varying(15)
phone character varying(24)
fax character varying(24)
homepage text
categories
category_id smallint (1)
category_name character varying(15) (1)
description text
picture bytea
customer_demographics
customer_type_id character varying(5) (1)
customer_desc text
customers
customer_id character varying(5) (1)
company_name character varying(40) (1)
contact_name character varying(30)
contact_title character varying(30)
address character varying(60)
city character varying(15)
region character varying(15)
postal_code character varying(10)
country character varying(15)
phone character varying(24)
fax character varying(24)
shippers
shipper_id smallint (1)
company_name character varying(40) (1)
phone character varying(24)
products
product_id smallint (1)
product_name character varying(40) (1)
supplier_id smallint
category_id smallint
quantity_per_unit character varying(20)
unit_price real
units_in_stock smallint
units_on_order smallint
reorder_level smallint
discontinued integer (1)
customer_customer_demo
customer_id character varying(5) (1)
customer_type_id character varying(5) (1)
order_details
order_id smallint (1)
product_id smallint (1)
unit_price real (1)
quantity smallint (1)
discount real (1)
orders
order_id smallint (1)
customer_id character varying(5)
employee_id smallint
order_date date
required_date date
shipped_date date
ship_via smallint
freight real
ship_name character varying(40)
ship_address character varying(60)
ship_city character varying(15)
ship_region character varying(15)
ship_postal_code character varying(10)
ship_country character varying(15)
employees
employee_id smallint (1)
last_name character varying(20) (1)
first_name character varying(10) (1)
title character varying(30)
title_of_courtesy character varying(25)
birth_date date
hire_date date
address character varying(60)
city character varying(15)
region character varying(15)
postal_code character varying(10)
country character varying(15)
home_phone character varying(24)
extension character varying(4)
photo bytea
notes text
reports_to smallint
photo_path character varying(255)
employee_territories
employee_id smallint (1)
territory_id character varying(20) (1)
us_states
state_id smallint (1)
state_name character varying(100)
state_abbr character varying(2)
state_region character varying(50)
region
region_id smallint (1)
region_description character varying(60) (1)
territories
territory_id character varying(20) (1)
territory_description character varying(60) (1)
region_id smallint (1)