Проект: итерация 3

Велегурина Елена Б05-121

Пункт 8: Создать индексы для таблиц (для которых это целесообразно), аргументировав выбор поля, по которому будет создан индекс.

Primary key (первичный ключ) имеет встроенный индекс. Поэтому создавать для соответствующих столбцов ещё один будет нецелесообразно.

1) В таблице *carpet* создадим индекс по полю *producer_id*, так как это поле используется для связи с таблицей *producer*. Индекс позволит быстро находить ковры, произведенные определенным производителем.
2) Для таблицы *producer* можно создать индекс по полю *country*, так как это поле может использоваться для фильтрации производителей по стране происхождения.
3) Для таблицы *supply* можно создать индекс по полю *date_from*, так как это поле может использоваться для поиска поставок по дате начала.
4) Для таблицы *carpet_description* можно создать индекс по полю *category*, так как это поле часто будет использоваться для поиска ковров по категории.
5) Для таблицы *carpet_supply* создадим индексы по полям *carpet_id* и *supply_id*, так как это связующие поля между таблицами *carpet* и *supply*, и индексы позволят быстро находить информацию о поставках конкретного ковра.

```
CREATE INDEX idx_carpet_producer_id
   ON shop.carpet (producer_id);

CREATE INDEX idx_producer_country
   ON shop.producer (country);

CREATE INDEX idx_supply_date_from
   ON shop.supply (date_from);

CREATE INDEX idx_carpet_description_category
   ON shop.carpet_description (category);

CREATE INDEX idx_carpet_supply_carpet_id_supply_id
   ON shop.carpet_supply (carpet_id, supply_id);
```

Пункт 9: Подготовить не менее 6 представлений:

1. 2-3 получаются сокрытием полей с персональными данными клиентов из таблицы, а также сокрытием технических полей. Для сокрытия полей с персональными данными недостаточно просто целиком удалить столбец с данными. Например, для поля `CARD_NO` можно использовать маскировку вида `4276********0000`. Бонусом вместе с кодом можно приложить тесты для представлений.
2. 3-4 получаются соединением нескольких таблиц с целью получения осмысленной сводной таблицы, например, хранящей некоторую статистику продаж/частот обращения клиента и т.д. Бонусом вместе с кодом можно приложить быть тесты для представлений.

1. Сокрытие поля с персональными данными клиентов из таблицы *shop.supply* для отображения информации о доставках.
В данном представлении в поле *address* остаётся только регион, остальные же символы заменяются на 7 звездочек, чтобы скрыть персональные данные клиентов, а именно их адреса. Отсортируем по дате начала доставки.

```
CREATE VIEW shop.supply_view AS
SELECT supply_id,
      date_from,
      date_to,
      substring(address, 1, position(',' IN address) - 1) || ',
*******' AS address
FROM shop.supply
ORDER BY date_from;
```

Результат:

	supply_id	date_from	date_to	address
1	1	2022-04-03	2022-04-03	Rostovskaya oblast, **…
2	2	2022-04-04	2022-04-05	Krasnodarskiy kray, **…
3	3	2022-04-04	2022-04-05	Kostromskaya oblast, *…
4	4	2022-04-06	2022-04-07	Vladimirskaya oblast, …
5	6	2022-04-07	2022-04-07	Krasnodarskiy kray, **…
6	7	2022-04-09	2022-04-10	Moskovskaya oblast, **…
7	8	2022-04-10	2022-04-11	Novosibirskaya oblast,…
8	9	2023-01-14	2023-02-10	Moskovskaya oblast, **…
9	10	2023-03-09	2023-03-11	Vladimirskaya oblast, …
10	11	2023-04-06	2022-04-07	Moskovskaya oblast, **…

2. Сокрытие полей с технической информацией из таблицы *shop.carpet* для отображения пользовательской информации о коврах в наличии.
В данном представлении удаляются поля *amount* и *producer_id*, которое содержит техническую информацию о количестве товара на складе и идентификатор производителя, которые не представляют ценности для пользователей. Отсортируем по цене.

```
CREATE VIEW shop.carpet_info_view AS
SELECT carpet_id,
       name,
       price
FROM shop.carpet
ORDER BY price;
```

Результат:

	carpet_id	name	price
1	9	Искусственная трава, г…	390
2	10	Искусственная трава га…	519
3	1	Кварц-виниловая LVT са…	883
4	2	Плитка ПВХ LVT Tarkett…	951
5	5	Линолеум напольный на …	1210
6	8	Ламинат кроношпан Loft…	1287
7	6	Ковролин на пол офисны…	1341
8	7	Коврики самоклеящиеся …	1427
9	3	Кварц-виниловый ламина…	1849
10	4	Грязезащитное противос…	3649

3. Получение сводной таблицы, содержащей информацию о количестве ковров, проданных каждым производителем. Отсортируем по результирующей колонке в обратном порядке.

```
CREATE VIEW shop.carpet_sales_by_producer AS
SELECT p.producer_id,
       p.name         AS producer_name,
       sum(c.amount) AS total_sales
FROM shop.carpet c
         JOIN shop.producer p
              ON c.producer_id = p.producer_id
GROUP BY p.producer_id,
         p.name
ORDER BY total_sales DESC;
```

Результат:

	producer_id	producer_name	total_sales
1	3	Магазин искусственных …	668
2	4	ИМ Групп	166
3	1	France carpets	146
4	5	Remontnick.ru	75
5	2	Brigadir	22

4. Получение сводной таблицы, содержащей информацию о средних ценах поставщиков на их ковры. Отсортируем по средней цене.

```
CREATE VIEW shop.average_price_by_producer AS
SELECT producer.name,
      avg(carpet.price) AS average_price
FROM shop.carpet
        JOIN shop.producer
             ON carpet.producer_id = producer.producer_id
GROUP BY producer.name
ORDER BY average_price;
```

Результат:

	name	average_price
1	Магазин искусственных цветов No.1	706.3333333333333333
2	ИМ Групп	1314
3	Remontnick.ru	1427
4	France carpets	1827.6666666666666667
5	Brigadir	1849

5. Получение сводной таблицы, содержащей информацию о частых клиентах, а именно о количестве их заказов ковров. Отсортируем по результирующей колонке в убывающем порядке.

```
CREATE VIEW shop.carpet_amount_by_address AS
SELECT supply.address,
      count(carpet_supply.carpet_id) AS total_carpet
FROM shop.carpet_supply
        JOIN shop.supply
             ON carpet_supply.supply_id = supply.supply_id
GROUP BY supply.address
ORDER BY total_carpet DESC;
```

Результат:

	address	total_carpet
1	Moskovskaya oblast, Moskva, Glagoleva Gene…	3
2	Vladimirskaya oblast, Vladimir, Suzdalskiy…	2
3	Krasnodarskiy kray, Krasnodar, Gidrostroit…	2
4	Kostromskaya oblast, Kostroma, Severnoy Pr…	1
5	Rostovskaya oblast, Rostov-na-donu, Poselk…	1
6	Novosibirskaya oblast, Novosibirsk, Vyborn…	1

6. Получение сводной таблицы, содержащей информацию о наиболее популярных категориях ковров в зависимости от количества их закупок. Отсортируем по результирующей колонке в убывающем порядке.

```
CREATE VIEW shop.carpet_sales_by_category AS
SELECT carpet_description.category,
      sum(carpet.amount) AS total_sold
FROM shop.carpet
        JOIN shop.carpet_description
             ON carpet.carpet_id = carpet_description.carpet_id
GROUP BY carpet_description.category
ORDER BY total_sold DESC;
```

Результат:

	category	total_sold
1	Искусственная трава	575
2	Ковролин на резиновой основе	103
3	Грязезащитные покрытия	98
4	Линолеум	93
5	Ковролин для гостиниц	75
6	Плитка ПВХ / LVT	70
7	Ламинат	63

Напишем тесты для полученных представлений:

1. В первом тесте проверяем, что выбраны 4 столбца, товары действительно отсортированы по дате начала отправки и она не превосходит даты окончания.
2. В втором тесте проверяем, что выбраны 3 столбца и товары действительно отсортированы по цене.
3. В третье тесте проверяем, что выбраны 3 столбца и товары действительно отсортированы по числу проданных ковров в зависимости от поставщика.
4. В четвёртом тесте проверяем, что выбраны 2 столбца и товары действительно отсортированы по средней цене в зависимости от поставщика.
5. В пятом тесте проверяем, что выбраны 2 столбца и товары действительно отсортированы по числу ковров в зависимости от адреса доставки.

6. В шестом тесте проверяем, что выбраны 2 столбца и товары действительно отсортированы по числу проданных ковров в зависимости от категории.

```
import pandas as pd
import os
import psycopg2 as pg
import unittest
from dataclasses import dataclass


@dataclass
class Credentials:
    dbname: str = "pg_db"
    host: str = "127.0.0.1"
    port: int = 5432
    user: str = "postgres"
    password: str = "postgres"


def psycopg2_conn_string():
    return f"""
        dbname='{os.getenv("DBNAME", Credentials.dbname)}'
        user='{os.getenv("DBUSER", Credentials.user)}'
        host='{os.getenv("DBHOST", Credentials.host)}'
        port='{os.getenv("DBPORT", Credentials.port)}'
        password='{os.getenv("DBPASSWORD", Credentials.password)}'
"""


def set_connection():
    return pg.connect(psycopg2_conn_string())


class TestHardQueries(unittest.TestCase):
    def __init__(self, *args, **kwargs):
        super(TestHardQueries, self).__init__(*args, **kwargs)
        self.conn = set_connection()
        self.cursor = self.conn.cursor()

    def test1(self):
        query = """
            SELECT *
            FROM shop.supply_view;
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 4
        for i in range(result.shape[0]):
            assert '*******' in result.iloc[i].loc['address']
        assert result.iloc[0].loc['date_from'] <= \
```

```
            result.iloc[0].loc['date_to']
        for i in range(1, result.shape[0]):
            assert result.iloc[i].loc['date_from'] >= \
                   result.iloc[i - 1].loc['date_from']
            assert result.iloc[i].loc['date_from'] <= \
                   result.iloc[i].loc['date_to']

    def test2(self):
        query = """
            SELECT *
            FROM shop.carpet_info_view;
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 3
        for i in range(1, result.shape[0]):
            assert result.iloc[i].loc['price'] >= \
                   result.iloc[i - 1].loc['price']

    def test3(self):
        query = """
            SELECT *
            FROM shop.carpet_sales_by_producer;
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 3
        for i in range(1, result.shape[0]):
            assert result.iloc[i].loc['total_sales'] <= \
                   result.iloc[i - 1].loc['total_sales']

    def test4(self):
        query = """
            SELECT *
            FROM shop.average_price_by_producer;
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 2
        for i in range(1, result.shape[0]):
            assert result.iloc[i].loc['average_price'] >= \
                   result.iloc[i - 1].loc['average_price']

    def test5(self):
        query = """
            SELECT *
            FROM shop.carpet_amount_by_address;
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 2
        for i in range(1, result.shape[0]):
```

```
            assert result.iloc[i].loc['total_carpet'] <= \
                   result.iloc[i - 1].loc['total_carpet']

    def test6(self):
        query = """
            SELECT *
            FROM shop.carpet_sales_by_category;
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 2
        for i in range(1, result.shape[0]):
            assert result.iloc[i].loc['total_sold'] <= \
                   result.iloc[i - 1].loc['total_sold']

    def end(self):
        self.cursor.close()
        self.conn.close()
```

Результат:

```
============================= test session starts ==============================
collecting ... collected 6 items

test_views.py::TestHardQueries::test1 PASSED                             [ 16%]
test_views.py::TestHardQueries::test2 PASSED                             [ 33%]
test_views.py::TestHardQueries::test3 PASSED                             [ 50%]
test_views.py::TestHardQueries::test4 PASSED                             [ 66%]
test_views.py::TestHardQueries::test5 PASSED                             [ 83%]
test_views.py::TestHardQueries::test6 PASSED                             [100%]

======================== 6 passed, 6 warnings in 0.27s =========================
```

Пункт 10: Создать не менее 2 хранимых процедур/функций. Логика процедур/функций согласовывается с семинаристом. Вместе с кодом можно приложить тесты для хранимого кода.

1. Функция "get_carpet_info" принимает на вход id ковра и возвращает полную информацию о ковре, включая его имя, цену, количество, производителя, категорию, длину, ширину и цвет.

```
CREATE OR REPLACE FUNCTION shop.get_carpet_info(p_carpet_id
INTEGER)
   RETURNS TABLE
           (
               carpet_id     INTEGER,
               name          VARCHAR(256),
```

```
                price          INTEGER,
                amount         INTEGER,
                producer_name  VARCHAR(256),
                category       VARCHAR(256),
                length         INTEGER,
                width          INTEGER,
                colour         VARCHAR(256)
            )
AS
$$
BEGIN
    RETURN QUERY
        SELECT c.carpet_id,
               c.name,
               c.price,
               c.amount,
               p.name AS producer_name,
               cd.category,
               cd.length,
               cd.width,
               cd.colour
        FROM shop.carpet AS c
                 JOIN shop.producer AS p
                      ON c.producer_id = p.producer_id
                 JOIN shop.carpet_description AS cd
                      ON c.carpet_id = cd.carpet_id
        WHERE c.carpet_id = p_carpet_id;
END;
$$ LANGUAGE plpgsql;
```

Результат:

```
SELECT *
FROM shop.get_carpet_info(1);
```

	carpet_id	name	price	amount
1	10	Искусственная трава газон декоративн…	519	517

producer_name	category	length	width	colour
Магазин искусственных цветов No.1	Искусственная трава	1	1	green

2. Функция "get_total_carpet_amount" возвращает общее количество ковров на складе, чтобы узнавать его текущую загруженность.

```
CREATE OR REPLACE FUNCTION shop.get_total_carpet_amount()
    RETURNS INTEGER AS
$$
BEGIN
    RETURN (SELECT sum(amount)
            FROM shop.carpet);
```

```
END;
$$ LANGUAGE plpgsql;
```

Результат:

```
SELECT *
FROM shop.get_total_carpet_amount();
```

	get_total_carpet_amount
1	1077

Напишем тесты для полученных хранимых функции:

1. Проверим, что функция возвращает таблицу 1 на 9. Проверим, что цена, ширина, длина и идентификатор положительны и количество ковров на складе неотрицательно.
2. Проверим, что функция возвращает таблицу 1 на 1. Проверим, что количество ковров на складе неотрицательно.

```
import pandas as pd
import os
import psycopg2 as pg
import unittest
from dataclasses import dataclass


@dataclass
class Credentials:
    dbname: str = "pg_db"
    host: str = "127.0.0.1"
    port: int = 5432
    user: str = "postgres"
    password: str = "postgres"


def psycopg2_conn_string():
    return f"""
        dbname='{os.getenv("DBNAME", Credentials.dbname)}'
        user='{os.getenv("DBUSER", Credentials.user)}'
        host='{os.getenv("DBHOST", Credentials.host)}'
        port='{os.getenv("DBPORT", Credentials.port)}'
        password='{os.getenv("DBPASSWORD", Credentials.password)}'
"""


def set_connection():
    return pg.connect(psycopg2_conn_string())
```

```python
class TestHardQueries(unittest.TestCase):
    def __init__(self, *args, **kwargs):
        super(TestHardQueries, self).__init__(*args, **kwargs)
        self.conn = set_connection()
        self.cursor = self.conn.cursor()

    def test1(self):
        query = """
            SELECT *
            FROM shop.get_carpet_info(1);
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 9
        assert result.shape[0] == 1
        assert result.iloc[0].loc['carpet_id'] >= 1
        assert result.iloc[0].loc['length'] >= 1
        assert result.iloc[0].loc['width'] >= 1
        assert result.iloc[0].loc['price'] >= 1
        assert result.iloc[0].loc['amount'] >= 0

    def test2(self):
        query = """
            SELECT *
            FROM shop.get_total_carpet_amount();
        """
        result = pd.read_sql(query, con=self.conn)
        assert result.shape[1] == 1
        assert result.shape[0] == 1
        assert result.iloc[0].loc['get_total_carpet_amount'] >= 0

    def end(self):
        self.cursor.close()
        self.conn.close()
```

Результат:

```
============================= test session starts ==============================
collecting ... collected 2 items

test_functions.py::TestHardQueries::test1 PASSED                         [ 50%]
test_functions.py::TestHardQueries::test2 PASSED                         [100%]

======================== 2 passed, 2 warnings in 0.22s =========================
```