username: kz2437
password: 5539

This Part 4 expands the existing database schema from Part 2 by adding advanced PostgreSQL features. The new schema includes full-text search, array attributes, and composite types.

1.I Added product_review table with full-text search support for the review_text attribute. A tsvector column (tsvector_review) has been added to facilitate text search indexing. An index was also created on the tsvector_review column to improve search performance.
2.I Added an array attribute categories to the product table to store multiple categories for each product.
3.I Created a new composite type employee_type and a table employees of that type.

When inserting data into the product_review table, the tsvector_review column will be automatically generated based on the review_text content.
The review_id in the product_review table and the employee_id in the employees table are of SERIAL type, and their values will be automatically generated when inserting new rows.

Schema:

add a text attribute for full-text search
The addition of the product_review table with a review_text column of type TEXT was made to store detailed customer reviews about products. It is important to enable efficient searching through these reviews based on keywords. The full-text search capabilities in PostgreSQL make it easier for users to search for specific words or phrases within the reviews, improving the overall user experience.

```
CREATE TABLE product_review (
  review_id SERIAL PRIMARY KEY,
  product_id VARCHAR(10) REFERENCES product (productID),
  review_text TEXT NOT NULL,
  review_date DATE NOT NULL
);

ALTER TABLE product_review
  ADD COLUMN tsvector_review tsvector
    GENERATED ALWAYS AS (to_tsvector('english', coalesce(review_text, ''))) STORED;

CREATE INDEX textsearch_idx ON product_review USING GIN (tsvector_review);

INSERT INTO product_review (product_id, review_text, review_date) VALUES
('B01N90RZ4M', 'The universal remote works perfectly with my Tata Sky setup.',
'2023-01-15'),
('B096MSW6CT', 'The charging cable is really fast and sturdy. Great purchase!',
'2023-02-10'),
('B08DDRGWTJ', 'The MI USB Type-C cable is durable and provides a secure
connection.', '2023-02-28'),
('B09V17S2BG', 'I love my boAt Wave Lite smartwatch. It has so many useful
features.', '2023-03-02'),
('B0B23LW7NV', 'The Spigen tempered glass screen protector fits my iPhone 14 Pro
Max perfectly.', '2023-03-20'),
('B07M69276N', 'The TP-Link AC1300 USB WiFi adapter improved my internet speed
significantly.', '2023-03-22'),
```

```
('B09NNGHG22', 'The picture quality on my Sansui 4K Ultra HD Android LED TV is
amazing.', '2023-04-01'),
('B006LW0WDQ', 'The Amazon Basics 16-Gauge speaker wire is reliable and easy to
use.', '2023-04-10'),
('B0758F7KK7', 'The Caprigo heavy duty TV wall mount bracket is sturdy and well-
built.', '2023-04-12'),
('B08NCKT9FG', 'The Boat A 350 Type C cable is a great value for the price.',
'2023-04-14');
```

```
# define a new composite type and create a table of that type
# In a real-world scenario, employees often have several attributes such as name,
position, and salary. Using a composite type allows us to group these attributes
logically within a single column in the employees table. This not only simplifies
the schema but also makes it easier to manage and query employee-related data.


CREATE TYPE employee_type AS (
  first_name VARCHAR(20),
  last_name VARCHAR(20),
  position VARCHAR(30),
  salary NUMERIC(10, 2)
);

CREATE TABLE employees (
  employee_id SERIAL PRIMARY KEY,
  employee_info employee_type
);


INSERT INTO employees (employee_info) VALUES
(ROW('John', 'Doe', 'Software Engineer', 180000)::employee_type),
(ROW('Jane', 'Smith', 'Product Manager', 195000)::employee_type),
(ROW('Alice', 'Johnson', 'Data Analyst', 120000)::employee_type),
(ROW('Bob', 'Brown', 'Web Developer', 105000)::employee_type),
(ROW('Charlie', 'Davis', 'UX Designer', 120000)::employee_type),
(ROW('Eve', 'Garcia', 'Technical Writer', 90000)::employee_type),
(ROW('Mallory', 'Harris', 'QA Engineer', 120000)::employee_type),
(ROW('Trent', 'Martinez', 'DevOps Engineer', 185000)::employee_type),
(ROW('Wendy', 'Lee', 'HR Manager', 150000)::employee_type),
(ROW('Trudy', 'Lewis', 'Marketing Specialist', 90000)::employee_type);



# add an array attribute
# Products can belong to multiple categories, and it's important to have a flexible
way to store and query these categories. Using an array attribute allows us to
store multiple category values for each product in a single column, simplifying the
schema and making queries more efficient.

ALTER TABLE product ADD COLUMN categories VARCHAR(25)[];

UPDATE product SET categories = ARRAY['Remote Controls'] WHERE productID =
'B01N90RZ4M';
```

```
UPDATE product SET categories = ARRAY['Cables & Accessories'] WHERE productID =
'B096MSW6CT';
UPDATE product SET categories = ARRAY['Accessories & Peripherals'] WHERE productID
= 'B08DDRGWTJ';
UPDATE product SET categories = ARRAY['Wearable Technology'] WHERE productID =
'B09V17S2BG';
UPDATE product SET categories = ARRAY['Screen Protectors'] WHERE productID =
'B0B23LW7NV';
UPDATE product SET categories = ARRAY['Network Adapters'] WHERE productID =
'B07M69276N';
UPDATE product SET categories = ARRAY['Home Theater', 'TV & Video'] WHERE productID
= 'B09NNGHG22';
UPDATE product SET categories = ARRAY['Speaker Cables'] WHERE productID =
'B006LW0WDQ';
UPDATE product SET categories = ARRAY['TV Mounts', 'Stands & Turntables'] WHERE
productID = 'B0758F7KK7';
UPDATE product SET categories = ARRAY['Cables & Accessories'] WHERE productID =
'B08NCKT9FG';
```

Queries:

1. Full-text search query for product_review:

```
SELECT product_id, review_text, review_date
FROM product_review
WHERE tsvector_review @@ to_tsquery('english', 'battery | fast');
# This query searches for reviews containing the words "battery" or "fast" in the
product_review table.
# The @@ operator is used to compare the tsvector_review column with the result of
to_tsquery function.
```

2. Query to access elements in the categories array in the product table:

```
SELECT productID, product_name, categories[1] AS primary_category
FROM product
WHERE categories[1] = 'Cables & Accessories';
#This query selects products with 'Cables & Accessories' as the primary category.
#The categories[1] notation is used to access the first element in the categories
array.
```

3. Query to retrieve all employees with their respective employee information from the employees table:

```
SELECT employee_id, (employee_info).first_name, (employee_info).last_name,
(employee_info).position, (employee_info).salary
FROM employees;
# This query uses the (employee_info) notation to access the individual attributes
of the employee_type composite type.
```