

ACCA7002 Database Systems

Tutorial Tasks

Tutorial 6: JOINS and types

The aim of this exercise is to help you practice Joins

Task 6.1: Create a PostgreSQL database named "LibraryDB."

```
CREATE DATABASE LibraryDB;
```

Task 6.2: Create a "Books" table to store book information.

```
CREATE TABLE Books (  
    BookID serial PRIMARY KEY,  
    Title VARCHAR(100),  
    AuthorID INT,  
    ISBN VARCHAR(13),  
    Genre VARCHAR(50),  
    PublishedYear INT  
);
```

Task 6.3: Create an "Authors" table to store author information.

```
CREATE TABLE Authors (  
    AuthorID serial PRIMARY KEY,  
    AuthorName VARCHAR(100)  
);
```

Task 6.4: Create a "Members" table to store information about library members.

```
CREATE TABLE Members (  
    MemberID serial PRIMARY KEY,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    Email VARCHAR(100),  
    JoiningDate DATE  
);
```

Task 6.5: Insert sample data into the "Books," "Authors," and "Members" tables.

```
INSERT INTO Authors (AuthorName) VALUES  
    ('J.K. Rowling'),  
    ('George Orwell'),  
    ('Jane Austen');
```

```
INSERT INTO Books (Title, AuthorID, ISBN, Genre,
PublishedYear) VALUES
    ('Harry Potter and the Sorcerer's Stone', 1, '978-
0439708180', 'Fantasy', 1997),
    ('1984', 2, '978-0451524935', 'Dystopian', 1949),
    ('Pride and Prejudice', 3, '978-1613823720', 'Romance',
1813);
```

```
INSERT INTO Members (FirstName, LastName, Email, JoiningDate)
VALUES
    ('John', 'Doe', 'john.doe@email.com', '2023-01-15'),
    ('Alice', 'Smith', 'alice.smith@email.com', '2023-02-20'),
    ('Bob', 'Johnson', 'bob.johnson@email.com', '2023-03-10');
```

Task 6.6: Performing Joins

Perform an INNER JOIN to show which books are authored by whom.

```
SELECT Books.Title, Authors.AuthorName
FROM Books
INNER JOIN Authors ON Books.AuthorID = Authors.AuthorID;
```

Perform a LEFT OUTER JOIN to show all books and their respective authors.

```
SELECT Books.Title, Authors.AuthorName
FROM Books
LEFT JOIN Authors ON Books.AuthorID = Authors.AuthorID;
```

Perform a RIGHT OUTER JOIN to show all authors and their authored books.

```
SELECT Books.Title, Authors.AuthorName
FROM Books
RIGHT JOIN Authors ON Books.AuthorID = Authors.AuthorID;
```

Perform a FULL OUTER JOIN to show all books and all authors, including books with no known author.

```
SELECT Books.Title, Authors.AuthorName
FROM Books
FULL JOIN Authors ON Books.AuthorID = Authors.AuthorID;
```

Task 6.7: Answer the following:

1. What is the purpose of SQL joins, and when would you use them?

Answer: SQL joins are used to combine rows from two or more tables based on a related column. They are employed to retrieve data from multiple tables simultaneously and are particularly useful when you need to extract information from related datasets.

2. What is the key difference between INNER JOIN and LEFT JOIN?

Answer: The key difference is in the result set. An INNER JOIN returns only the matching rows from both tables, while a LEFT JOIN returns all rows from the left table and the matching rows from the right table. In a LEFT JOIN, if there are no matches in the right table, NULL values are returned.

3. How does a RIGHT JOIN differ from a LEFT JOIN?

Answer: A RIGHT JOIN returns all rows from the right table and the matching rows from the left table. In contrast, a LEFT JOIN returns all rows from the left table and the matching rows from the right table.

4. When is it appropriate to use a FULL OUTER JOIN?

Answer: A FULL OUTER JOIN is used when you want to retrieve all rows from both tables, including those with no matches in the other table. It's suitable for situations where you need a comprehensive view of the data from multiple sources.

5. Describe a real-life scenario where a SQL join would be necessary.

Answer: In a library database, you might use a SQL join to connect the "Books" table with the "Authors" table. This would allow you to display a list of books along with their respective authors, making it easier for library staff and members to find books by a specific author.

6. How do you avoid common pitfalls when using SQL joins, such as producing incorrect or unexpected results?

Answer: To avoid pitfalls, carefully plan the join conditions, understand the relationships between tables, and use aliases to differentiate columns with the same name in multiple tables. Additionally, it's important to test your queries and check the results to ensure they match your expectations.

7. Explain the difference between a self-join and a cross join.

Answer: A self-join is when a table is joined with itself to retrieve related data within the same table. A cross join, on the other hand, returns the Cartesian product of two tables, resulting in all possible combinations of rows from both tables.

8. What are the potential performance considerations when using joins in large databases?

Answer: Joins in large databases can be resource-intensive. It's essential to index columns used in join conditions, use proper database design, and optimize queries. Additionally, limiting the size of result sets and using appropriate filtering criteria can help improve performance.

9. Can you think of an example where you would use multiple joins in a single SQL query?

Answer: In an e-commerce database, you might use multiple joins to retrieve information about customers, their orders, order items, and product details in a single query. This would provide a comprehensive view of customer orders, including the products purchased.