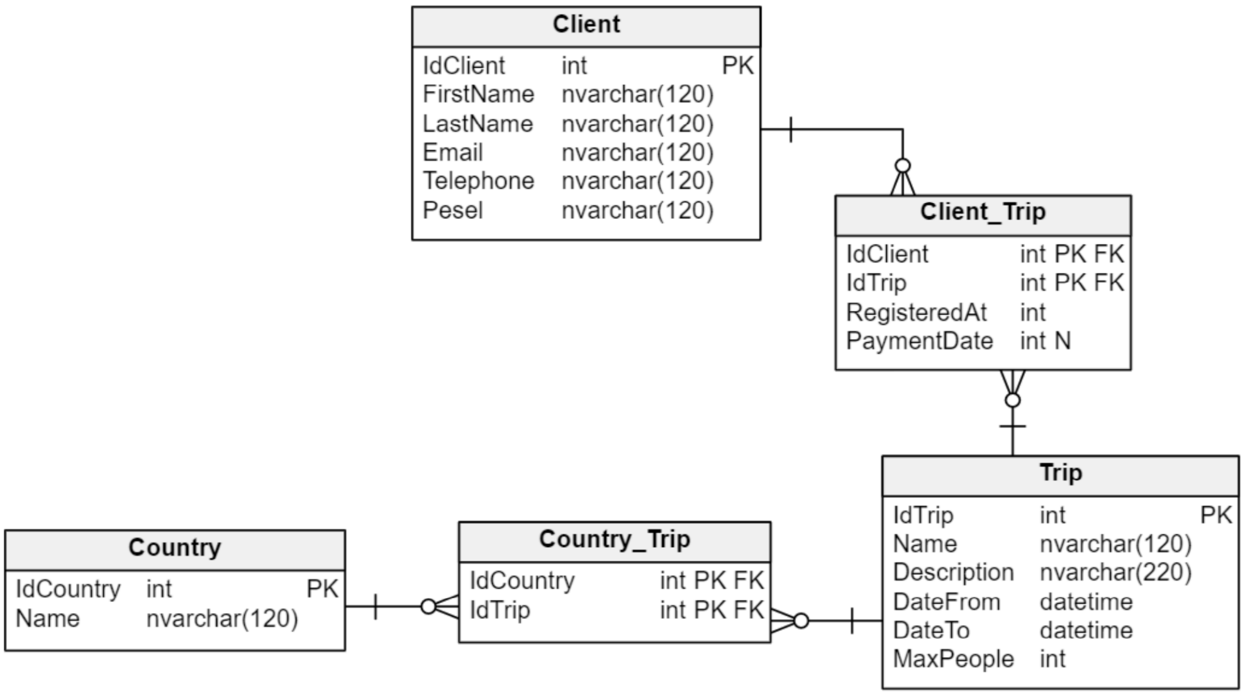

In this assignment, you will implement a REST API for a travel agency database. You'll create 5 endpoints that perform various operations using SQL connections and SqlCommand objects to interact with the database. Your API should handle data selection, insertion, updating, and deletion operations.

Required Endpoints

1. GET /api/trips

This endpoint will retrieve all available trips with their basic information.

- Implement using SqlConnection and SqlCommand
- Return trip details including ID, name, description, date range, and maximum number of participants
- Include country information for each trip

2. GET /api/clients/{id}/trips

This endpoint will retrieve all trips associated with a specific client.

- Accept client ID as a path parameter
- Return all trips that the client has registered for
- Include trip details and registration/payment information
- Handle cases where client doesn't exist or has no trips

3. POST /api/clients

This endpoint will create a new client record.

- Accept client details in the request body (FirstName, LastName, Email, Telephone, Pesel)
- Validate input data (required fields, format validation)
- Insert data into the Client table

- Return appropriate status codes and the newly created client ID

4. PUT /api/clients/{id}/trips/{tripId}

This endpoint will register a client for a specific trip.

- Accept client ID and trip ID as path parameters
- Check if the client and trip exist
- Check if maximum number of participants hasn't been reached
- Insert a record into Client_Trip table with current date as RegisteredAt
- Return appropriate status codes and messages

5. DELETE /api/clients/{id}/trips/{tripId}

This endpoint will remove a client's registration from a trip.

- Accept client ID and trip ID as path parameters
- Check if the registration exists
- Delete the record from the Client_Trip table
- Return appropriate status codes and messages

Requirements

1. Use ADO.NET with SqlConnection and SqlCommand objects (no Entity Framework or ORMs)
2. Implement proper error handling with appropriate HTTP status codes
3. Follow REST API design principles
4. Use parameterized queries to prevent SQL injection
5. Implement proper connection handling (opening/closing connections, using 'using' statements)
6. Add basic input validation
7. Document your API with comments explaining the purpose of each endpoint and SQL query