**Namal University Mianwali**

**Department of Computer Science**

[**CS-233L - Computer Networks (Lab)**](https://student.qualityobe.com/uniclassroom/view?id=64837)

|  |  |
| --- | --- |
| **Name** | **Reg no** |
| **Muhammad Usman**  **Zunaira Akbar**  **Saeeda Farnaz** | **NUM-BSCS-2022-08**  **NUM-BSCS-2022-34**  **NUM-BSCS-2022-54** |
| **Instructor** | **Date** |
| **Dr. Muzamil Ahmad** | **15\01\2025** |

# Lab 4: FTP, Email and DHCP server settings

**Distributed Querying over Network**

**Project Overview**

The **Distributed Querying over Network** project implements a **Client-Server Model** where a client application communicates with multiple server databases to retrieve and manage data. This system is designed to provide seamless access to distributed datasets hosted across different servers. The primary use case of this project is in scenarios where data is decentralized, and real-time querying and management across multiple systems are required.

**Project Objectives**

1. Develop a distributed querying system that connects multiple databases.
2. Provide a user-friendly GUI for managing database connections and executing queries.
3. Ensure secure and efficient communication between clients and servers.
4. Facilitate filtering and searching across distributed datasets.
5. Provide modular and extensible code for scalability.

**System Architecture**

The project is structured around three main components:

1. **Server Databases:** Each server hosts a MySQL database containing relevant datasets.
2. **Client Application:** A Python-based GUI that manages connections and executes queries across servers.
3. **Communication Protocols:** Use of MySQL connectors to enable secure and reliable database interactions.

**Database Schema**

The database consists of the following tables:

1. **Student:**
   * Columns: AdmissionNumber, FirstName, LastName, Address, ContactNo.
   * Stores information about students.
2. **Course:**
   * Columns: CourseID, Name, Description.
   * Contains details of courses offered.
3. **Enrolled:**
   * Columns: EnrollmentID, AdmissionNumber, CourseID, EnrollmentDate, CompletionDate.
   * Represents the enrollment of students in courses.

**Installation and Setup**

**Setting up Server Databases**

**Steps:**

1. **Database Schema Creation:**
   * Run database\_schema.sql to create the database structure.
2. **Populating Databases:**
   * Populate dummy data using pc1\_data.sql, pc2\_data.sql, and pc3\_data.sql.
3. **Verification:**
   * Execute pc1\_test.sql to validate the database setup.

**Hosting MySQL Server on Device A**

1. **Install MySQL Server:**
   * Download and install from the [MySQL website](https://dev.mysql.com/downloads/).
2. **Configure Remote Connections:**
   * Update bind-address to 0.0.0.0 in the MySQL configuration file.
   * Restart the MySQL service.
3. **Create Remote User:**
4. CREATE USER 'remote\_user'@'%' IDENTIFIED BY 'password';
5. GRANT ALL PRIVILEGES ON \*.\* TO 'remote\_user'@'%' WITH GRANT OPTION;
6. FLUSH PRIVILEGES;
7. **Allow Firewall Access:**
   * Open port 3306 for inbound connections.
8. **Retrieve IP Address:**
   * Use ipconfig to identify the IPv4 address of the server.

**Configuring Client Application**

1. **Install MySQL Connector:**
2. pip install mysql-connector-python
3. **Update Configuration:**
   * Update GUI.py with the server IP addresses.

**Code Implementation**

**main.py**

This module handles database interactions, including connecting to servers and querying data.

**Key Functions:**

1. **connect\_db:** Connects to a database with a timeout mechanism.
2. **fetch\_students:** Retrieves all student records.
3. **fetch\_courses:** Retrieves all course records.
4. **fetch\_enrollments:** Retrieves enrollment records, including student and course details.
5. **search\_students, search\_courses, search\_enrollments:** Filters records based on specific fields and values.

**GUI.py**

This module provides a user interface for managing database connections and displaying query results.

**Features:**

1. **Connection Management:**
   * Establish and close connections to multiple servers using checkboxes.
   * Display connection status messages.
2. **Data Display:**
   * Show query results in a table with adjustable columns.
3. **Filtering and Searching:**
   * Filter records using dropdown menus and search boxes.
4. **Error Handling:**
   * Display error messages for connection or query failures.
5. **Clean Exit:**
   * Close all active connections on application exit.

**GUI Design**

The graphical user interface is built using Tkinter and provides:

1. **Title Section:**
   * Displays the application title.
2. **Connection Controls:**
   * Checkboxes for managing server connections.
3. **Query Controls:**
   * Dropdown menus for selecting datasets and filters.
   * Search boxes for entering query parameters.
4. **Results Table:**
   * Displays query results in a tabular format.

**Testing and Validation**

1. **Database Testing:**
   * Verified schema creation using test.sql files.
   * Ensured correct data population with sample SQL scripts.
2. **Connection Testing:**
   * Tested remote MySQL connections using varying IP configurations.
3. **Functional Testing:**
   * Executed all query types via the GUI to confirm accuracy.
4. **Error Handling:**
   * Simulated connection failures to ensure robust error messages.

**Results and Discussion**

**Key Achievements:**

1. Successfully implemented a distributed querying system.
2. Enabled seamless access to multiple server databases.
3. Provided a user-friendly GUI for efficient data management.

**Challenges:**

1. Configuring MySQL for remote access required additional firewall adjustments.
2. Ensuring query accuracy across distributed datasets required careful synchronization.

**Future Enhancements:**

1. Add authentication mechanisms for secure client-server communication.
2. Implement caching for faster query results.
3. Extend the GUI to support advanced reporting and data visualization.

**Limitations**

1. The client and server devices must be on the same local LAN. While they can reside in different subnets, both devices must be part of the same network infrastructure, such as a university network with a router IP of 172.16.0.3. This constraint means that the system cannot operate if either the client or server is located outside this local network.

**Conclusion**

The **Distributed Querying over Network** project demonstrates an efficient and scalable approach to managing distributed datasets. By combining robust database management with an intuitive graphical interface, this project provides a practical solution for real-world data querying challenges.

**References**

1. [MySQL Community Server Documentation](https://dev.mysql.com/doc/)
2. Tkinter Python Library: <https://docs.python.org/3/library/tkinter.html>
3. MySQL Connector for Python: <https://pypi.org/project/mysql-connector-python/>