ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

UNIVERSIDAD

Computer Networks Laboratory

Laboratory No. 5
Databases and Network Protocols

Database Management Systems Installation and Network Protocol Analysis

Students:

Cristian Santiago Pedraza Rodríguez

Andersson David Sánchez Méndez

Instructor: Professor Fabian Eduardo Sierra Sánchez
Course: Computer Networks

Institution: Escuela Colombiana de Ingeniería Julio Garavito

October 11, 2025

Contents

1 Objectives

- Install and configure database management systems (PostgreSQL, SQL Server, Azure SQL Database)
- Create users and databases with proper access control
- Design relational database schemas with multiple tables
- Analyze network protocols using Wireshark
- Configure remote database access and auto-start services
- Understand cloud computing concepts and Azure services

2 Tools and Equipment

2.1 Required Software

- Virtualization software (VMware, VirtualBox)
- Linux Slackware virtual machine
- Windows Server virtual machine
- Wireshark network protocol analyzer
- DBeaver database client
- Microsoft Azure account (free tier)

2.2 Operating Systems

- Linux Slackware 15.0
- Windows Server 2019/2022
- Local Windows machine for testing

3 Introduction

Modern enterprise IT infrastructure relies heavily on robust database management systems to store, manage, and retrieve organizational data. This laboratory explores the installation, configuration, and management of various database platforms across different operating systems, while also examining the network protocols that enable communication between database clients and servers.

Database Management Systems (DBMS) can be deployed in multiple environments:

1. **On-Premise Infrastructure** - Databases hosted in company datacenters with full administrative control
2. **Cloud Infrastructure** - Managed database services (e.g., Azure SQL Database) with scalability and high availability
3. **Hybrid Solutions** - Combination of on-premise and cloud deployments

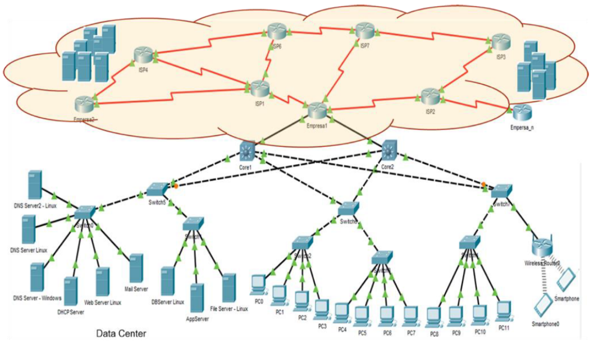


Figure 1: Enterprise network infrastructure with database servers

This laboratory focuses on three major components:

- Database installation and configuration on Linux and Windows platforms
- Network protocol analysis for understanding data transmission
- Cloud database deployment and management in Microsoft Azure

4 Part 1: Network Protocol Analysis

Before deploying database systems, understanding the underlying network protocols is essential for troubleshooting and optimization.

Exercise 1: DNS Message Analysis

Objective: Analyze DNS name resolution process using Wireshark.
Procedure:

1. Start Wireshark packet capture
2. Navigate to `http://www.google.com`
3. Apply filter: `dns`
4. Examine DNS query and response messages

DNS Message Fields Identified:

- **Transaction ID**: Unique identifier matching query with response
- **Flags**: QR (Query/Response), Opcode, RD (Recursion Desired), RA (Recursion Available)
- **Questions**: Number of queries (typically 1)
- **Answer RRs**: Resource records in response
- **Query Name**: Domain being resolved (www.google.com)
- **Query Type**: A (IPv4) or AAAA (IPv6)
- **TTL**: Time to live for caching

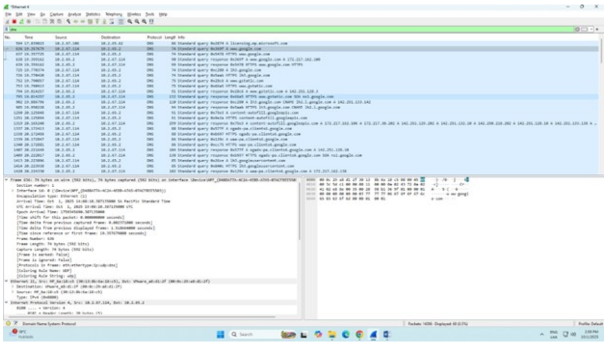


Figure 2: DNS query and response packets in Wireshark

Exercise 2: HTTP Message Analysis

Objective: Examine HTTP protocol headers and request/response cycle.
Target URL: http://profesores.is.escuelaing.edu.co/~csantiago/
Filter: `http`

HTTP Request Header Fields:

- **Method**: GET (requesting resource)
- **URI**: Requested resource path
- **Version**: HTTP/1.1
- **Host**: Destination server
- **User-Agent**: Browser information
- **Accept**: MIME types client can handle
- **Accept-Language**: Preferred languages
- **Accept-Encoding**: Compression methods supported

HTTP Response Header Fields:

- **Status Code**: 200 OK, 404 Not Found, etc.
- **Server**: Web server software
- **Content-Type**: MIME type of response
- **Content-Length**: Response body size
- **Date**: Response timestamp
- **Last-Modified**: Resource modification date

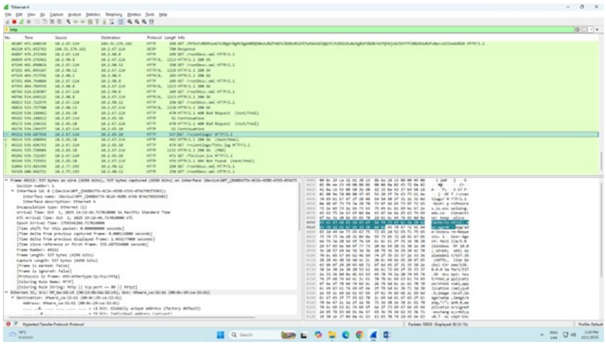


Figure 3: HTTP request and response packets in Wireshark

Exercise 3: Ethernet Frame Analysis

Objective: Analyze layer 2 Ethernet frame structure.
Filter: `http.request.method == "GET"`

Ethernet II Frame Fields:

- **Destination MAC** (6 bytes): Physical address of destination
- **Source MAC** (6 bytes): Physical address of sender
- **EtherType** (2 bytes): Protocol type (0x0800 for IPv4)
- **Payload**: Upper layer protocols (IP, TCP, HTTP)
- **FCS**: Frame Check Sequence for error detection

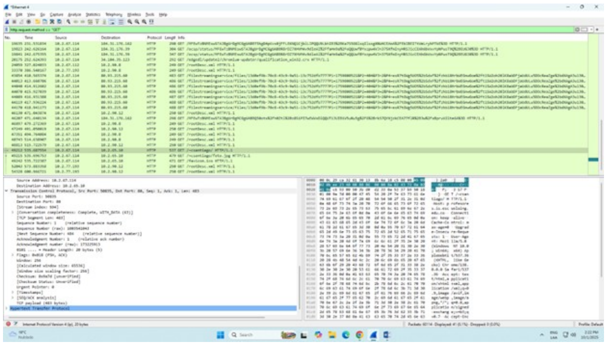


Figure 4: Ethernet frame structure in Wireshark

5 Part 2: PostgreSQL on Linux Slackware

Exercise 4: PostgreSQL Installation

Platform: Linux Slackware 15.0 VM
Team Members: Cristian Santiago Pedraza Rodríguez, Andersson David Sánchez Méndez

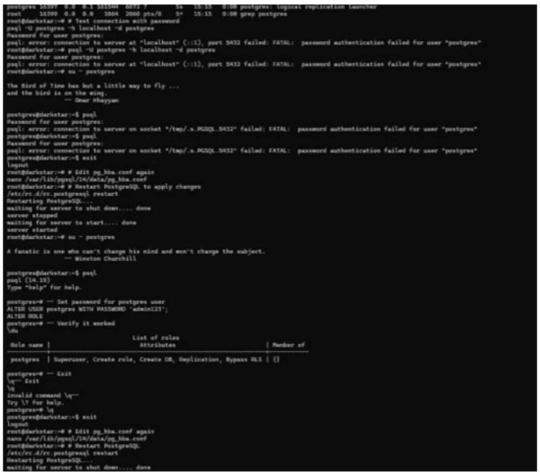
5.1 4.1 Installation Process

PostgreSQL was not available in standard Slackware repositories, requiring installation from SlackBuilds.org:

```
# Create build directory
mkdir -p /root/slackbuilds
cd /root/slackbuilds

# Download PostgreSQL SlackBuild
wget
    https://slackbuilds.org/slackbuilds/15.0/system/postgresql.tar.gz
tar xvf postgresql.tar.gz
cd postgresql

# Download PostgreSQL 14.19 source
wget
    https://ftp.postgresql.org/pub/source/v14.19/postgresql-14.19.tar.gz

# Create postgres user and group
groupadd -g 209 postgres
useradd -u 209 -g 209 -d /var/lib/pgsql -s /bin/bash postgres

# Build and install
chmod +x postgresql.SlackBuild
./postgresql.SlackBuild
installpkg /tmp/postgresql-14.19-i586-1_SBo.tgz
```

5.2 4.2 Database Initialization

```
# Initialize database cluster
mkdir -p /var/lib/pgsql/14/data
chown -R postgres:postgres /var/lib/pgsql
chmod 700 /var/lib/pgsql/14/data

su - postgres
initdb -D /var/lib/pgsql/14/data
exit

# Configure authentication
nano /var/lib/pgsql/14/data/pg_hba.conf
# Change 'trust' to 'md5' for password authentication

# Start PostgreSQL
/etc/rc.d/rc.postgresql start
```



```
postgres 16397  0.0  0.1 161544  6072 ?        Ss   15:15   0:00 postgres: logical replication launcher
root     16399  0.0  0.0   3864  2060 pts/0    S+   15:15   0:00 grep postgres
root@darkstar:~# # Test connection with password
psql -U postgres -h localhost -d postgres
Password for user postgres:
psql: error: connection to server at "localhost" (::1), port 5432 failed: FATAL:  password authentication failed for user "postgres"
root@darkstar:~# psql -U postgres -h localhost -d postgres
Password for user postgres:
psql: error: connection to server at "localhost" (::1), port 5432 failed: FATAL:  password authentication failed for user "postgres"
root@darkstar:~# su - postgres

The Bird of Time has but a little way to fly ...
and the bird is on the wing.
                -- Omar Khayyam

postgres@darkstar:~$ psql
Password for user postgres:
psql: error: connection to server on socket "/tmp/.s.PGSQL.5432" failed: FATAL:  password authentication failed for user "postgres"
postgres@darkstar:~$ psql
Password for user postgres:
psql: error: connection to server on socket "/tmp/.s.PGSQL.5432" failed: FATAL:  password authentication failed for user "postgres"
postgres@darkstar:~$ exit
logout
root@darkstar:~# # Edit pg_hba.conf again
nano /var/lib/pgsql/14/data/pg_hba.conf
root@darkstar:~# # Restart PostgreSQL to apply changes
/etc/rc.d/rc.postgresql restart
Restarting PostgreSQL...
waiting for server to shut down.... done
server stopped
waiting for server to start.... done
server started
root@darkstar:~# su - postgres

A fanatic is one who can't change his mind and won't change the subject.
                -- Winston Churchill

postgres@darkstar:~$ psql
psql (14.19)
Type "help" for help.

postgres=# -- Set password for postgres user
ALTER USER postgres WITH PASSWORD 'admin123';
ALTER ROLE
postgres=# -- Verify it worked
\du
                                   List of roles
 Role name |                         Attributes                         | Member of
-----------+------------------------------------------------------------+-----------
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS | {}

postgres=# -- Exit
\q-- Exit
\q
invalid command \q--
Try \? for help.
postgres=# \q
postgres@darkstar:~$ exit
logout
root@darkstar:~# # Edit pg_hba.conf again
nano /var/lib/pgsql/14/data/pg_hba.conf
root@darkstar:~# # Restart PostgreSQL
/etc/rc.d/rc.postgresql restart
Restarting PostgreSQL...
waiting for server to shut down.... done
```

Figure 5: PostgreSQL installation on Slackware

5.3 4.3 User and Database Creation

```
-- Connect as postgres superuser
psql

-- Create users
CREATE USER cristian_pedraza WITH PASSWORD 'secure_password1';
CREATE USER andersson_sanchez WITH PASSWORD 'secure_password2';

-- Create databases
CREATE DATABASE cristian_tourism OWNER cristian_pedraza;
CREATE DATABASE andersson_tourism OWNER andersson_sanchez;

-- Grant privileges
GRANT ALL PRIVILEGES ON DATABASE cristian_tourism TO cristian_pedraza;
GRANT ALL PRIVILEGES ON DATABASE andersson_tourism TO andersson_sanchez;

-- Verify
\du
\l
```

5.4 4.4 Database Schema Design

Theme: Colombian Tourist Sites

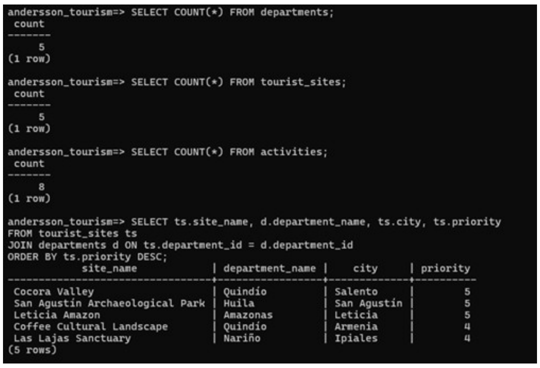
Tables:

1. `departments` - Colombian regions
2. `tourist_sites` - Tourist destinations
3. `activities` - Available activities at each site

```
-- Connect to cristian_tourism database
psql -U cristian_pedraza -d cristian_tourism

-- Table 1: Departments
CREATE TABLE departments (
    department_id SERIAL PRIMARY KEY,
    department_name VARCHAR(100) NOT NULL UNIQUE,
    region VARCHAR(50) NOT NULL,
    description TEXT
);

-- Table 2: Tourist Sites
CREATE TABLE tourist_sites (
    site_id SERIAL PRIMARY KEY,
    site_name VARCHAR(200) NOT NULL,
    department_id INTEGER REFERENCES departments(department_id),
    city VARCHAR(100) NOT NULL,
    site_type VARCHAR(50) NOT NULL,
    description TEXT,
    best_season VARCHAR(50),
    estimated_cost DECIMAL(10,2),
    priority INTEGER CHECK (priority BETWEEN 1 AND 5),
    visited BOOLEAN DEFAULT FALSE
);

-- Table 3: Activities
CREATE TABLE activities (
    activity_id SERIAL PRIMARY KEY,
    site_id INTEGER REFERENCES tourist_sites(site_id) ON DELETE CASCADE,
    activity_name VARCHAR(150) NOT NULL,
    duration_hours DECIMAL(4,1),
    difficulty_level VARCHAR(20),
    cost DECIMAL(10,2)
);
```

5.5 4.5 Data Insertion

```
-- Insert departments
INSERT INTO departments (department_name, region, description) VALUES
('Bolivar', 'Caribe', 'Coastal region with colonial history'),
('Cundinamarca', 'Andina', 'Central highlands containing Bogota'),
('Antioquia', 'Andina', 'Coffee region with vibrant culture'),
('Magdalena', 'Caribe', 'Sierra Nevada and Caribbean beaches'),
('San Andres y Providencia', 'Insular', 'Caribbean islands');

-- Insert tourist sites
INSERT INTO tourist_sites (site_name, department_id, city, site_type, best_season, estimated_cost, priority) VALUES
('Cartagena Walled City', 1, 'Cartagena', 'Historical', 'December-March', 800000.00, 5),
('Tayrona National Park', 4, 'Santa Marta', 'Natural', 'December-March', 500000.00, 5),
('Guatape and El Penol', 3, 'Guatape', 'Natural/Cultural', 'Year-round', 300000.00, 4),
('Salt Cathedral of Zipaquira', 2, 'Zipaquira', 'Religious', 'Year-round', 150000.00, 4),
('San Andres Island', 5, 'San Andres', 'Beach', 'December-May', 1500000.00, 5);

-- Insert activities
INSERT INTO activities (site_id, activity_name, duration_hours, difficulty_level, cost) VALUES
(1, 'Walking tour of old city', 3.0, 'Easy', 50000.00),
(1, 'Sunset boat tour', 2.5, 'Easy', 80000.00),
(2, 'Hiking to Cabo San Juan', 4.0, 'Moderate', 60000.00),
(2, 'Snorkeling', 2.0, 'Easy', 100000.00),
(3, 'Climb El Penol rock', 1.5, 'Moderate', 25000.00);

-- Verify data
SELECT COUNT(*) FROM departments;
SELECT COUNT(*) FROM tourist_sites;
SELECT COUNT(*) FROM activities;
```



```
andersson_tourism=> SELECT COUNT(*) FROM departments;
 count
-------
     5
(1 row)

andersson_tourism=> SELECT COUNT(*) FROM tourist_sites;
 count
-------
     5
(1 row)

andersson_tourism=> SELECT COUNT(*) FROM activities;
 count
-------
     8
(1 row)

andersson_tourism=> SELECT ts.site_name, d.department_name, ts.city, ts.priority
FROM tourist_sites ts
JOIN departments d ON ts.department_id = d.department_id
ORDER BY ts.priority DESC;
            site_name            | department_name |    city     | priority
---------------------------------+-----------------+-------------+----------
 Cocora Valley                   | Quindío         | Salento     |        5
 San Agustín Archaeological Park | Huila           | San Agustín |        5
 Leticia Amazon                  | Amazonas        | Leticia     |        5
 Coffee Cultural Landscape       | Quindío         | Armenia     |        4
 Las Lajas Sanctuary             | Nariño          | Ipiales     |        4
(5 rows)
```

Figure 6: Sample query results from PostgreSQL database

6 Part 3: SQL Server on Windows Server

Exercise 5: SQL Server Installation

Platform: Windows Server 2019 VM (accessed via SSH and Remote Desktop)
Database Theme: Monthly Activity Schedule

6.1 5.1 Remote Access Configuration

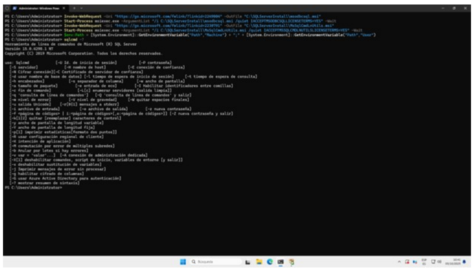
SSH was configured for remote command-line access:

```
# Enable SSH (PowerShell as Administrator)
Add-WindowsCapability -Online -Name OpenSSH.Server~~~~0.0.1.0
Start-Service sshd
Set-Service -Name sshd -StartupType 'Automatic'

# Configure firewall
New-NetFirewallRule -Name sshd -DisplayName 'OpenSSH Server'
      -Enabled True -Direction Inbound -Protocol TCP -Action
      Allow -LocalPort 22

# Connect from local machine
ssh Administrator@10.2.77.180
```

6.2 5.2 SQL Server Installation

Due to language compatibility issues with command-line installation, SQL Server was installed via Remote Desktop GUI:

1. Downloaded SQL Server 2019 Express
2. Extracted installation media
3. Launched GUI installer (Básico installation)
4. Configured mixed-mode authentication
5. Set `sa` password: Admin123!



```
PS C:\Users\Administrator> Invoke-WebRequest -Uri "https://go.microsoft.com/fwlink/?linkid=2249004" -OutFile "C:\SQLServerInstall\msodbcsql.msi"
PS C:\Users\Administrator> Start-Process msiexec.exe -ArgumentList "/i C:\SQLServerInstall\msodbcsql.msi /quiet IACCEPTMSODBCSQLLICENSETERMS=YES" -Wait
PS C:\Users\Administrator> Invoke-WebRequest -Uri "https://go.microsoft.com/fwlink/?linkid=2230791" -OutFile "C:\SQLServerInstall\MsSqlCmdLnUtils.msi"
PS C:\Users\Administrator> Start-Process msiexec.exe -ArgumentList "/i C:\SQLServerInstall\MsSqlCmdLnUtils.msi /quiet IACCEPTMSSQLCMDLNUTILSLICENSETERMS=YES" -Wait
PS C:\Users\Administrator> $env:Path = [System.Environment]::GetEnvironmentVariable("Path","Machine") + ";" + [System.Environment]::GetEnvironmentVariable("Path","User")
PS C:\Users\Administrator> sqlcmd -?
Herramienta de línea de comandos de Microsoft (R) SQL Server
Versión 15.0.4298.1 NT
Copyright (C) 2019 Microsoft Corporation. Todos los derechos reservados.

uso: Sqlcmd          [-U Id. de inicio de sesión]         [-P contraseña]
  [-S servidor]           [-H nombre de host]          [-E conexión de confianza]
  [-N Cifrar conexión][-C Certificado de servidor de confianza]
  [-d usar nombre de base de datos] [-l tiempo de espera de inicio de sesión]     [-t tiempo de espera de consulta]
  [-h encabezados]           [-s separador de columna]      [-w ancho de pantalla]
  [-a tamaño de paquete]        [-e entrada de eco]        [-I Habilitar identificadores entre comillas]
  [-c fin de comando]              [-L[c] enumerar servidores [salida limpia]]
  [-q "consulta de línea de comandos"]   [-Q "consulta de línea de comandos" y salir]
  [-m nivel de error]        [-V nivel de gravedad]     [-W quitar espacios finales]
  [-u salida Unicode]    [-r[0|1] mensajes a stderr]
  [-i archivo de entrada]         [-o archivo de salida]        [-z nueva contraseña]
  [-f <página de códigos> | i:<página de códigos>[,o:<página de códigos>]] [-Z nueva contraseña y salir]
  [-k[1|2] quitar [reemplazar] caracteres de control]
  [-y ancho de pantalla de longitud variable]
  [-Y ancho de pantalla de longitud fija]
  [-p[1] imprimir estadísticas[formato dos puntos]]
  [-R usar configuración regional de cliente]
  [-K intención de aplicación]
  [-M conmutación por error de múltiples subredes]
  [-b Anular por lotes si hay errores]
  [-v var = "valor"...]  [-A conexión de administración dedicada]
  [-X[1] deshabilitar comandos, script de inicio, variables de entorno [y salir]]
  [-x deshabilitar sustitución de variables]
  [-j Imprimir mensajes de error sin procesar]
  [-g habilitar cifrado de columnas]
  [-G usar Azure Active Directory para autenticación]
  [-? mostrar resumen de sintaxis]
PS C:\Users\Administrator>
```

Figure 7: SQL Server installation center

6.3 5.3 User and Database Creation

```sql
-- Create SQL logins
sqlcmd -S localhost\SQLEXPRESS -E

CREATE LOGIN cristian_pedraza WITH PASSWORD = 'Cris123!',
      CHECK_POLICY = OFF;
GO
CREATE LOGIN andersson_sanchez WITH PASSWORD = 'Ander123!',
      CHECK_POLICY = OFF;
GO

-- Create databases
CREATE DATABASE cristian_schedule;
GO
CREATE DATABASE andersson_schedule;
GO

-- Create database users
USE cristian_schedule;
CREATE USER cristian_pedraza FOR LOGIN cristian_pedraza;
ALTER ROLE db_owner ADD MEMBER cristian_pedraza;
GO

USE andersson_schedule;
CREATE USER andersson_sanchez FOR LOGIN andersson_sanchez;
ALTER ROLE db_owner ADD MEMBER andersson_sanchez;
GO
```

6.4 5.4 Database Schema Design

Tables for Activity Schedule:

1. `categories` - Activity categories
2. `priorities` - Priority levels
3. `activities` - Scheduled activities

```sql
USE cristian_schedule;
GO

-- Table 1: Categories
CREATE TABLE categories (
    category_id INT PRIMARY KEY IDENTITY(1,1),
    category_name NVARCHAR(50) NOT NULL UNIQUE,
    color_code NVARCHAR(7),
    description NVARCHAR(200)
);

-- Table 2: Priorities
CREATE TABLE priorities (
    priority_id INT PRIMARY KEY IDENTITY(1,1),
    priority_level NVARCHAR(20) NOT NULL UNIQUE,
    urgency_score INT CHECK (urgency_score BETWEEN 1 AND 5),
    description NVARCHAR(200)
);

-- Table 3: Activities
CREATE TABLE activities (
    activity_id INT PRIMARY KEY IDENTITY(1,1),
    activity_title NVARCHAR(200) NOT NULL,
    activity_description NVARCHAR(MAX),
    category_id INT FOREIGN KEY REFERENCES
          categories(category_id),
    priority_id INT FOREIGN KEY REFERENCES
          priorities(priority_id),
    activity_date DATE NOT NULL,
    start_time TIME,
    end_time TIME,
    location NVARCHAR(200),
    is_completed BIT DEFAULT 0,
    created_at DATETIME DEFAULT GETDATE()
);
```

6.5 5.5 Data Insertion

```sql
-- Insert categories
INSERT INTO categories (category_name, color_code,
      description) VALUES
('Academic', '#3498db', 'University classes'),
('Work', '#e74c3c', 'Job activities'),
('Personal', '#2ecc71', 'Personal development'),
('Health', '#f39c12', 'Exercise and medical'),
('Social', '#9b59b6', 'Social events'),
('Family', '#1abc9c', 'Family time');
```

```

-- Insert priorities
INSERT INTO priorities (priority_level, urgency_score,
      description) VALUES
('Critical', 5, 'Immediate'),
('High', 4, 'Important and urgent'),
('Medium', 3, 'Important not urgent'),
('Low', 2, 'Nice to have'),
('Optional', 1, 'Can postpone');

-- Insert activities (October 2025)
INSERT INTO activities (activity_title, category_id,
      priority_id, activity_date, start_time, end_time,
      location) VALUES
('Network Protocols Lab', 1, 2, '2025-10-02', '14:00',
      '18:00', 'Computer Lab'),
('Database Design Meeting', 1, 2, '2025-10-03', '10:00',
      '12:00', 'Virtual'),
('Morning Workout', 4, 3, '2025-10-03', '06:30', '07:30',
      'Gym'),
('SQL Server Install', 1, 2, '2025-10-04', '15:00', '17:00',
      'Home'),
('Data Structures Quiz', 1, 1, '2025-10-07', '08:00', '10:00',
      'Room 301');
```

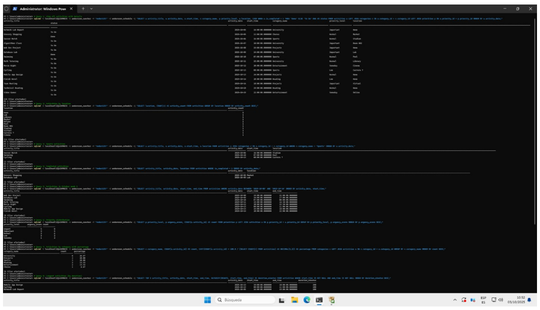


Figure 8: Activity schedule data in SQL Server

7 Part 4: Azure SQL Database

Exercise 6: Cloud Database Deployment

Platform: Microsoft Azure SQL Database
Database Theme: Library Management (Books and Scientific Articles)

7.1 6.1 Cloud Computing Concepts

What is Cloud Computing?

Cloud computing is the delivery of computing services (servers, storage, databases, networking, software) over the internet on a pay-as-you-go basis.

Advantages of Azure vs On-Premise:

- **Cost Efficiency**: No upfront hardware costs
- **Scalability**: Easy vertical and horizontal scaling
- **High Availability**: 99.99% uptime SLA
- **Global Reach**: Multiple regions worldwide
- **Maintenance**: Microsoft handles updates and patches
- **Disaster Recovery**: Built-in backup and replication

Service Models:

- **IaaS**: Virtual Machines, Storage (user manages OS and apps)
- **PaaS**: Azure SQL Database, App Service (user manages only data and apps)
- **SaaS**: Microsoft 365, Office 365 (fully managed)

Regions and Availability Zones:

Regions are geographic areas with data centers. Availability Zones are physically separate locations within regions, providing high availability and disaster recovery with 99.99% uptime SLA.

Scaling Strategies:

- **Vertical Scaling**: Increase resources (CPU, RAM) of existing instance
- **Horizontal Scaling**: Add more instances to distribute load

Transport Layer Security:

Azure SQL Database enforces TLS 1.2+ encryption by default, ensuring all data in transit is encrypted. This differs from local VM databases where TLS must be manually configured.

TCP Connection Handling:

Azure SQL uses a gateway architecture that proxies connections, enabling automatic failover and load balancing. Local VM databases use direct TCP connections with static IP addresses.

7.2 6.2 Azure SQL Database Creation

```
# Azure Portal steps:
1. Create Resource Group: lab05-database-rg
2. Create SQL Server: lab05-sql-server-[unique]
   - Location: East US
   - Admin: sqladmin
   - Password: Admin123!@#
3. Create SQL Database: library_db
   - Pricing tier: Basic (5 DTU)
   - Backup storage: Locally-redundant
4. Configure Firewall:
   - Allow Azure services: Yes
   - Add client IP: 192.168.1.2
```

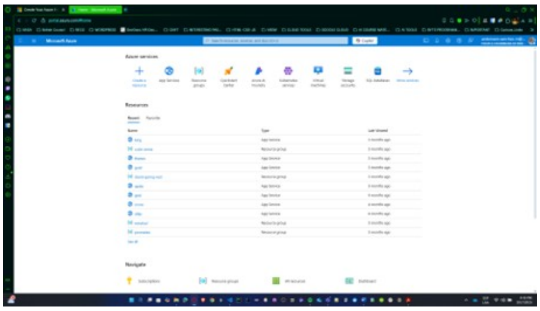


Figure 9: Azure SQL Database in portal

7.3 6.3 Database Schema

Tables:

1. `authors` - Author information
2. `books` - Book records
3. `scientific_articles` - Article records

```sql
-- Table 1: Authors
CREATE TABLE authors (
    author_id INT PRIMARY KEY IDENTITY(1,1),
    first_name NVARCHAR(50) NOT NULL,
    last_name NVARCHAR(50) NOT NULL,
    country NVARCHAR(50),
    birth_year INT,
    specialty NVARCHAR(100)
);

-- Table 2: Books
CREATE TABLE books (
    book_id INT PRIMARY KEY IDENTITY(1,1),
    title NVARCHAR(200) NOT NULL,
    author_id INT FOREIGN KEY REFERENCES authors(author_id),
    isbn NVARCHAR(20) UNIQUE,
    publication_year INT,
    genre NVARCHAR(50),
    pages INT,
    language NVARCHAR(30),
    available_copies INT DEFAULT 1
);

-- Table 3: Scientific Articles
CREATE TABLE scientific_articles (
    article_id INT PRIMARY KEY IDENTITY(1,1),
    title NVARCHAR(300) NOT NULL,
    author_id INT FOREIGN KEY REFERENCES authors(author_id),
    journal_name NVARCHAR(150),
    publication_date DATE,
    doi NVARCHAR(100) UNIQUE,
    field_of_study NVARCHAR(100),
    citation_count INT DEFAULT 0,
    open_access BIT DEFAULT 0
);
```

7.4 6.4 Sample Data

```sql
-- Insert authors
INSERT INTO authors (first_name, last_name, country,
      birth_year, specialty) VALUES
(’Gabriel’, ’Garcia Marquez’, ’Colombia’, 1927, ’Magical
      Realism’),
(’Stephen’, ’Hawking’, ’UK’,1942, ’Theoretical Physics’),
(’Marie’, ’Curie’, ’Poland’, 1867, ’Physics and Chemistry’),
(’Isaac’, ’Asimov’, ’Russia’, 1920, ’Science Fiction’);

-- Insert books
INSERT INTO books (title, author_id, isbn, publication_year,
      genre, pages, language) VALUES
(’Cien Anos de Soledad’, 1, ’978-0307474728’, 1967, ’Magical
      Realism’, 417, ’Spanish’),
(’A Brief History of Time’, 2, ’978-0553380163’, 1988,
      ’Science’, 256, ’English’),
(’Foundation’, 4, ’978-0553293357’, 1951, ’Science Fiction’,
      255, ’English’);

-- Insert scientific articles
INSERT INTO scientific_articles (title, author_id,
      journal_name, publication_date, doi, field_of_study,
      citation_count, open_access) VALUES
(’Radioactive Substances’, 3, ’Nature’, ’1898-07-15’,
      ’10.1038/radioactive.1898’, ’Physics’, 15420, 0),
(’Black Holes and Thermodynamics’, 2, ’Physical Review D’,
      ’1974-04-01’, ’10.1103/PhysRevD.blackholes’, ’Theoretical
      Physics’, 8934, 1),
(’Foundation of Robotics’, 4, ’IEEE Robotics’, ’1950-12-01’,
      ’10.1109/robotics.foundation’, ’Computer Science’, 7821,
      1);

-- Query examples
SELECT b.title, a.first_name + ’ ’ + a.last_name AS author
FROM books b
JOIN authors a ON b.author_id = a.author_id;

SELECT field_of_study, COUNT(*) AS article_count,
      AVG(citation_count) AS avg_citations
FROM scientific_articles
GROUP BY field_of_study;
```

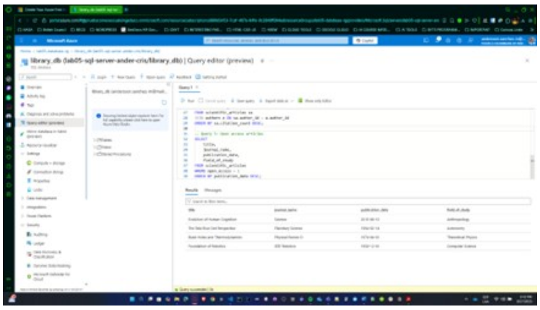

library_db (lab05-sql-server-ander-cris/library_db) | Query editor (preview)

Figure 10: Query results in Azure Query Editor

7.5 6.5 Video Demonstration

A comprehensive video demonstration of the Azure SQL Database implementation, including connection setup, table structure, and data queries, is available at:

Azure SQL Database Demo Video
https://youtu.be/Ye6jw5p2mKk

The video showcases:

- Azure Portal navigation and database creation
- Connection to library_db database
- Table structure and relationships (authors, books, scientific_articles)
- Sample queries demonstrating joins and aggregations
- Data retrieval and analysis

7.6 6.6 Resource Cleanup

After completing the demonstration and documentation, all Azure resources were deleted to avoid additional costs:

```
# Delete resource group (removes all resources)
1. Navigate to Resource Groups in Azure Portal
2. Select lab05-database-rg
3. Click "Delete resource group"
4. Type resource group name to confirm
5. Click Delete
```

This ensures no charges are incurred and free trial credits are preserved.

8 Part 5: Remote Access Configuration

Exercise 7: Auto-Start and Remote Connectivity

Objective: Configure database auto-start on boot and enable remote connections via DBeaver

8.1 7.1 PostgreSQL Auto-Start (Slackware)

```bash
# Make startup script executable
chmod +x /etc/rc.d/rc.postgresql

# Add to rc.local for auto-start
echo "/etc/rc.d/rc.postgresql start" >> /etc/rc.d/rc.local
chmod +x /etc/rc.d/rc.local

# Configure for remote access
nano /var/lib/pgsql/14/data/postgresql.conf
# Change: listen_addresses = ’*’

nano /var/lib/pgsql/14/data/pg_hba.conf
# Add: host all all 0.0.0.0/0 md5

# Restart PostgreSQL
/etc/rc.d/rc.postgresql restart

# Verify listening on all interfaces
netstat -an | grep 5432
```

8.2 7.2 SQL Server Auto-Start (Windows)

```
# PowerShell - Set service to automatic
Set-Service -Name "MSSQL`$SQLEXPRESS" -StartupType Automatic

# Enable TCP/IP protocol
# Via SQL Server Configuration Manager:
1. Open SQL Server Configuration Manager
2. Expand SQL Server Network Configuration
3. Protocols for SQLEXPRESS
4. Enable TCP/IP
5. Set port 1433 in IPAll section
6. Restart SQL Server service

# Configure Windows Firewall
New-NetFirewallRule -DisplayName "SQL Server TCP 1433" -Direction Inbound -Protocol TCP -LocalPort 1433 -Action Allow

# Verify service status
Get-Service "MSSQL`$SQLEXPRESS"
netstat -an | findstr 1433
```

8.3 7.3 DBeaver Remote Connection

PostgreSQL Connection Settings:

- **Host**: 10.2.77.176 (Slackware VM IP)
- **Port**: 5432
- **Database**: cristian_tourism
- **Username**: cristian_pedraza
- **Password**: secure_password1
- **Authentication**: Database Native

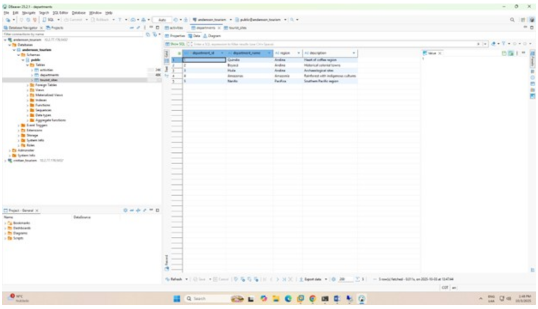


Figure 11: DBeaver connected to PostgreSQL database

SQL Server Connection Settings:

- **Host**: 10.2.77.180 (Windows Server VM IP)
- **Port**: 1433
- **Database**: cristian_schedule
- **Username**: cristian_pedraza
- **Password**: Cris123!
- **Authentication**: SQL Server Authentication

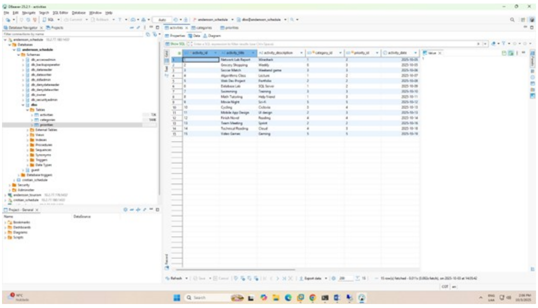


Figure 12: DBeaver connected to SQL Server database

Azure SQL Database Connection Settings:

- **Host**: lab05-sql-server-[unique].database.windows.net
- **Port**: 1433
- **Database**: library_db
- **Username**: sqladmin
- **Password**: Admin123!@#
- **Driver Properties**: encrypt=true, trustServerCertificate=true

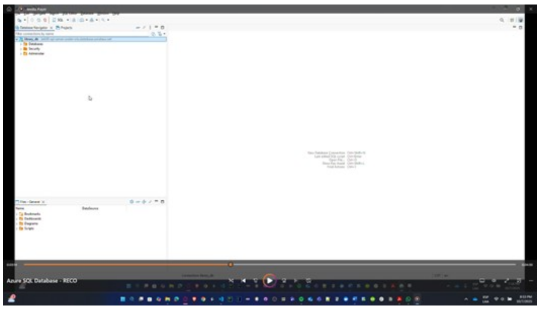


Figure 13: DBeaver connected to Azure SQL Database

9 Sample Queries and Analysis

9.1 PostgreSQL Query Examples

```sql
-- Query 1: Tourist sites by priority
SELECT ts.site_name, d.department_name, ts.priority, ts.estimated_cost
FROM tourist_sites ts
JOIN departments d ON ts.department_id = d.department_id
ORDER BY ts.priority DESC, ts.estimated_cost ASC;

-- Query 2: Activities by site with total cost
SELECT ts.site_name,
       COUNT(a.activity_id) AS activity_count,
       SUM(a.cost) AS total_activity_cost
FROM tourist_sites ts
LEFT JOIN activities a ON ts.site_id = a.site_id
GROUP BY ts.site_name
ORDER BY total_activity_cost DESC;

-- Query 3: Sites by region
SELECT d.region, COUNT(ts.site_id) AS site_count
FROM departments d
LEFT JOIN tourist_sites ts ON d.department_id = ts.department_id
GROUP BY d.region
ORDER BY site_count DESC;
```

9.2 SQL Server Query Examples

```sql
-- Query 1: Activities with category and priority
SELECT a.activity_title, a.activity_date, a.start_time,
       c.category_name, p.priority_level, a.location
```

```sql
FROM activities a
LEFT JOIN categories c ON a.category_id = c.category_id
LEFT JOIN priorities p ON a.priority_id = p.priority_id
ORDER BY a.activity_date, a.start_time;

-- Query 2: Activity count by category
SELECT c.category_name, COUNT(a.activity_id) AS count
FROM categories c
LEFT JOIN activities a ON c.category_id = a.category_id
GROUP BY c.category_name
ORDER BY count DESC;

-- Query 3: High priority pending activities
SELECT a.activity_title, a.activity_date, p.priority_level
FROM activities a
JOIN priorities p ON a.priority_id = p.priority_id
WHERE a.is_completed = 0 AND p.urgency_score >= 4
ORDER BY a.activity_date;
```

9.3 Azure SQL Database Query Examples

```sql
-- Query 1: Books with author information
SELECT b.title, a.first_name + ' ' + a.last_name AS author_name,
       b.publication_year, b.genre, b.available_copies
FROM books b
JOIN authors a ON b.author_id = a.author_id
ORDER BY b.publication_year DESC;

-- Query 2: Most cited articles
SELECT TOP 5 sa.title, a.first_name + ' ' + a.last_name AS author,
       sa.journal_name, sa.citation_count
FROM scientific_articles sa
JOIN authors a ON sa.author_id = a.author_id
ORDER BY sa.citation_count DESC;

-- Query 3: Publications by author
SELECT a.first_name + ' ' + a.last_name AS author_name,
       a.specialty,
       (SELECT COUNT(*) FROM books WHERE author_id = a.author_id) AS books_count,
       (SELECT COUNT(*) FROM scientific_articles WHERE author_id = a.author_id) AS articles_count
FROM authors a
ORDER BY books_count + articles_count DESC;
```

10 Troubleshooting and Lessons Learned

10.1 PostgreSQL Installation Challenges

Challenge: PostgreSQL not available in standard Slackware repositories.

Solution: Used SlackBuilds.org to compile from source. Required creating postgres user/group and proper directory permissions.

Key Learning: Understanding manual package compilation and dependency management in Slackware.

10.2 SQL Server Installation Issues

Challenge: Spanish installer refused command-line installation on English Windows Server.

Solution: Used Remote Desktop to access GUI installer, which bypassed language check.

Key Learning: Not all enterprise software supports fully automated installation across all language configurations. GUI access is sometimes necessary.

10.3 Remote Connection Configuration

Challenge: Firewall blocking PostgreSQL port 5432 and SQL Server port 1433.

Solution:

- Configured `listen_addresses = '*'` in postgresql.conf
- Modified pg_hba.conf to allow network connections
- Added Windows Firewall rules for SQL Server
- Enabled TCP/IP protocol in SQL Server Configuration Manager

Key Learning: Database security involves multiple layers (authentication, network configuration, firewall rules).

10.4 Azure SQL Database Connection

Challenge: Initial connection timeout due to missing firewall rules.

Solution: Added client IP address (192.168.1.2) to Azure SQL Server firewall rules through portal.

Key Learning: Cloud databases require explicit firewall configuration for external access. Azure provides centralized management through portal interface.

11 Conclusions

This laboratory provided comprehensive hands-on experience with enterprise database management systems across multiple platforms. Key accomplishments include:

- Successfully installed and configured PostgreSQL on Linux Slackware and SQL Server on Windows Server, demonstrating cross-platform database administration skills.
- Deployed and managed Azure SQL Database in the cloud, understanding the differences between IaaS, PaaS, and SaaS service models and the benefits of managed database services.
- Designed and implemented normalized database schemas with proper relationships using foreign keys for three different use cases: tourism management, activity scheduling, and library management.
- Configured remote database access through authentication settings, network protocols, and firewall rules, successfully connecting via DBeaver from a local machine to all three platforms.
- Analyzed network protocols (DNS, HTTP, Ethernet) using Wireshark, understanding how data flows through network layers and how this knowledge applies to database connectivity troubleshooting.
- Implemented user-based access control and security measures, including password authentication and understanding the importance of transport layer security (TLS) in cloud environments.

Key Takeaway: Modern IT infrastructure requires proficiency across multiple database platforms and deployment models. Understanding both on-premise and cloud solutions, along with network fundamentals, is essential for effective database administration.

12 References

1. PostgreSQL Global Development Group. *PostgreSQL 14.19 Documentation.* Available: https://www.postgresql.org/docs/14/
2. Microsoft Corporation. *SQL Server 2019 Documentation.* Available: https://docs.microsoft.com/en-us/sql/
3. Microsoft Corporation. *Azure SQL Database Documentation.* Available: https://docs.microsoft.com/en-us/azure/sql-database/
4. Wireshark Foundation. *Wireshark User's Guide.* Available: https://www.wireshark.org/docs/
5. SlackBuilds.org. *PostgreSQL SlackBuild.* Available: https://slackbuilds.org/repository/15.0/system/postgresql/
6. DBeaver Corporation. *DBeaver Universal Database Tool.* Available: https://dbeaver.io/
7. RFC 1035 - Domain Names - Implementation and Specification. IETF, 1987.
8. RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1. IETF, 1999.
9. Date, C.J. *An Introduction to Database Systems*, 8th Edition. Addison-Wesley, 2003.
10. Silberschatz, A., Korth, H., Sudarshan, S. *Database System Concepts*, 7th Edition. McGraw-Hill, 2019.