A framework to securely connect to your SQL databases from anywhere via HTTP

AceQL HTTP lets you connect to your remote or Cloud SQL databases with SQL from any device (mobile, tablet, PC, etc.)
Just develop regular SQL calls with your usual C#, Java, or Python IDE, and the software will take care of all protocol, communications and security aspects.

Why AceQL?

Streamline your Desktop and Mobile App developments...

Normally, accessing remote SQL databases data from a desktop or mobile app requires writing, testing and deploying Web Services.

It also requires management of the server code, client code, and client-server dialog, including error detection.

With AceQL HTTP, you only need to code the client part to access the remote SQL data. There's no server code nor duplex communication between the client code and a Web Service server.

The framework takes care of all the complex aspects of the client-server dialog (communications, data parsing, error detection).

...**Give your users easy, secure access to your SQL DBs from their favorite database tool**

No need to set VPNs up for your users to access their SQL databases from their favorite database tool. No need to configure the firewall - just allow regular HTTPS flow. Because AceQL HTTP uses only HTTP protocol, just give your users the database address, username and password, and they're set. Of course, all this can be highly secured.

Use C#, Java or Python from client side with unmodified SQL syntax

The C# / Xamarin Client SDK allows SQL calls to be encoded with a standard C# SQL API: the SDK C# SQL syntax is identical to the Microsoft SQL Server C# API.

The Java Client JDBC Driver allows SQL calls to be encoded with standard unmodified JDBC syntax (java.sql package interfaces).

The Python Client SDK allows SQL calls to be encoded with standard unmodified DB-API 2.0 syntax (PEP 249 -- DB-API 2.0 interface).

Use any language from any OS platform

Want to use a language that isn't C#, Java or Python? Just wrap your SQL calls and parse the query results using our universal Rest APIs.

Or just tell us your need for an implementation with your favorite X / Y / Z language, and we'll integrate the request into our road map.

Use advanced SQL features directly in your desktop and mobile app code

AceQL isn't limited to simple SELECT / INSERT / UPDATE / DELETE calls.

You can use advanced SQL options:

- create transactions by passing a series of SQL commands in autocommit off mode,
- insert BLOBs or read BLOBS,
- schedule batches to quickly feed your remote databases,
- call all existing stored procedures for your company or organization.

Code your Desktop & Mobile C# and Xamarin apps with Microsoft SQL Server syntax

The C# AceQL Client SDK uses Microsoft SQL Server-like syntax. SDK class names are the equivalent of SQL Server System.Data.SqlClient namespace. They share the same suffix name for the classes, and the same method names.

Connection to a remote database:

```
string server = "https://www.acme.com:9443/aceql";
string database = "sampledb";

string connectionString = $"Server={server}; Database={database}";
string username = "MyUsername"; // User & Password are hardcoded to simplify our
sample...
char[] password = { 'M', 'y', 'S', 'e', 'c', 'r', 'e', 't' };

AceQLConnection connection = new AceQLConnection(connectionString)
{
    Credential = new AceQLCredential(username, password)
};

// Attempt to establish a connection to the remote SQL database:
await connection.OpenAsync();
Console.WriteLine("Successfully connected to database " + database + "!");
```

SELECT on the remote database:

```
string sql = "select customer_id, customer_title, lname from customer where
customer_id = 1";

using (AceQLCommand command = new AceQLCommand(sql, connection))
using (AceQLDataReader dataReader = await command.ExecuteReaderAsync())
{
    while (dataReader.Read())
    {
        Console.WriteLine();
        Console.WriteLine("customer_id   : " + dataReader.GetValue(0));
        Console.WriteLine("customer_title: " + dataReader.GetValue(1);
        Console.WriteLine("lname         : " + dataReader.GetValue(2);
    }
}
```

The C# Client SDK is packaged as a .Net Standard 2.0 Library to use in your Xamarin projects. There is no adaptation per target required. Write a unique and shared C# Xamarin code and make it run on all major desktop & mobile operating systems:

- Android
- iOS
- macOS
- Windows Desktop

Immediately integrate the JDBC Driver into your Desktop & Mobile Java apps

The AceQL Client JDBC Driver allows you to develop your SQL / JDBC code without any learning or adaptation. The main JDBC classes and methods are supported as is. It's possible to integrate the Driver into your apps without modifying their source code.

Connection to a remote database:

```
// The URL of the AceQL Server servlet
String url = "https://www.acme.com/aceql";

// Load the AceQL Client JDBC Driver
DriverManager.registerDriver(new AceQLDriver());
Class.forName(AceQLDriver.class.getName());

// User & Password are hardcoded to simplify our sample...
Properties info = new Properties();
info.put("user", "MyUsername");
info.put("password", "MySecret");
info.put("database", "sampledb"); // The remote database to use

// Attempts to establish a connection to the remote database:
Connection connection = DriverManager.getConnection(url, info);
System.out.println("Successfully connected to database " + database + "!");
```

SELECT on the remote database:

```
String sql = "SELECT CUSTOMER_ID, FNAME, LNAME FROM CUSTOMER WHERE CUSTOMER_ID =
?";
PreparedStatement prepStatement = connection.prepareStatement(sql);
prepStatement.setInt(1, customerId);

ResultSet rs = prepStatement.executeQuery();
while (rs.next()) {
    System.out.println();
    System.out.println("customer_id: " + rs.getInt("customer_id"));
    System.out.println("fname      : " + rs.getString("fname"));
    System.out.println("lname      : " + rs.getString("lname"));
}
```

The AceQL Client JDBC Driver can be integrated directly into your favorite database visualizer, including metadata management:

Get easy access to all your SQL corporate databases from your Python apps

The module supports Python 3.6–3.9 and provides a SQL interface compliant with the DB-API 2.0 specification described by PEP 249.

Connection to a remote database:

```
# URL of the AceQL server, Remote SQL database name authentication info
url = "https://www.acme.com:9443/aceql"

# User & Password are hardcoded to simplify our sample...
username = "user1"
password = "password1"
database = "sampledb"

# Attempts to establish a connection to the remote database:
connection = aceql.connect(url=url, username=username, password=password,
database=database)
```

SELECT on the remote database:

```
cursor = connection.cursor()
sql = "SELECT CUSTOMER_ID, FNAME, LNAME FROM CUSTOMER WHERE CUSTOMER_ID = ?"
params = (1,)
cursor.execute(sql, params)
row = cursor.fetchone()
print(row)
```

Take complete control of your server-side configuration and deployment

The startup configuration only requires 4 standard and well-known JDBC SQL connection properties to be entered in a properties file: `driverClassName`, `url`, `username`, `password`.

Most of the configuration, including the firewalling & client authentication, can be done by just filling in the lines of the properties file:

- Default Tomcat JDBC Pool customization,
- SSL/TLS configuration to encrypt the HTTP flow between the client users and AceQL Server,
- Default and ready-to-use client users authentication against Windows, SSH, or LDAP servers,
- Configuration of the AceQL Server asynchronous servlet using a `ThreadPoolExecutor`,
- Chained SQL firewall rulesets for each database.

The advanced configuration is done using dynamic Java code injection of your own or third-party Java classes. It allows you to take complete control of JDBC pool creation, define the firewalling process exactly (including third-party software plug-ins), define client users' session creation, etc.

AceQL HTTP may be run stateless: deploy it freely using your favorite virtualization tools like Docker and Kubernetes.

Define your SQL databases security

AceQL HTTP is a 3-tier architecture and includes a comprehensive security manager, allowing you to:

- Freely choose and configure a strong authentication system to protect your SQL databases.
- Control the syntax and parameters of incoming SQL commands.
- Trigger actions in the event of unexpected or malicious SQL commands.