SQL Database Insertion Using Python and Oracle

Abstract

This paper demonstrates the integration of Python with Oracle Database 11g XE for executing SQL data insertion operations. By accepting user inputs, computing a result, and inserting this data into a database, the project showcases the practical implementation of backend programming principles in Python, including secure database connectivity, arithmetic computation, and safe SQL execution through parameterized queries.

1. Introduction

As modern software systems increasingly depend on reliable database interactions, backend technologies must seamlessly integrate programming languages with database management systems. This project investigates a lightweight solution using Python and Oracle Database 11g to perform CRUD operations, specifically focusing on data insertion.

2. Tools and Technologies

- Language: Python 3.x
- Database: Oracle Database 11g XE
- Driver: oracledb Python library (Thick mode)
- Client: Oracle Instant Client
- Environment: Visual Studio Code / PyCharm / IDLE on Windows OS

3. Methodology

The application flow begins by initializing the Oracle client in thick mode, enabling enhanced communication with the Oracle database. The script accepts two floating-point numbers from the user, computes their sum, and inserts all values into a table named 'sums' using parameterized SQL queries to ensure security and efficiency.

4. Implementation

The script connects to the database using credentials and DSN information, then uses a try-except block for input validation. Once the sum is calculated, the values are securely inserted into the database:

SQL Database Insertion Using Python and Oracle

```
import oracledb

oracledb.init_oracle_client(lib_dir=r"C:\oraclexe\app\oracle\product\11.2.0\server\bin")
connection = oracledb.connect(user='system', password='Sridhar@210', dsn='localhost/xe')
cursor = connection.cursor()

try:
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
    result = num1 + num2
except ValueError:
    print("Please enter valid numbers.")
    exit()

insert_query = "INSERT INTO sums (number1, number2, sum_result) VALUES (:1, :2, :3)"
cursor.execute(insert_query, (num1, num2, result))
connection.commit()

print("Data inserted successfully!")
cursor.close()
connection.close()
```

5. Results

Upon execution, the script prompts the user for input and confirms the successful insertion of the values into the Oracle database. The table 'sums' stores the original numbers and their sum, demonstrating functional database integration.

6. Discussion

This project highlights Python's potential for backend integration with enterprise databases. While functional, improvements such as GUI development or web integration using Flask or Django can extend its usability. Furthermore, the direct use of credentials in code is discouraged in production environments; instead, secure practices such as environment variables or encrypted configuration files are recommended.

7. Conclusion

The project effectively illustrates a basic yet essential technique in backend development: connecting Python applications to Oracle databases for CRUD operations. It provides a foundation for developing scalable and secure database-driven applications.

References

- Oracle Database Documentation: https://docs.oracle.com/cd/B28359_01/
- Python Oracle Driver: https://oracle.github.io/python-oracledb/
- Python Docs: https://docs.python.org/3/
- TutorialsPoint: https://www.tutorialspoint.com/python/python_database_access.htm
- Oracle Instant Client: https://www.oracle.com/database/technologies/instant-client.html
- Real Python SQL Guide: https://realpython.com/python-sql-libraries/#using-sqlite
- TechWhale: https://techwhale.in/python-oracle-database-connection/
- Stack Overflow DB Security: https://stackoverflow.com/questions/38843607/how-to-secure-database-credentials-in-python
- GeeksforGeeks: https://www.geeksforgeeks.org/python-sqlite3/
- Flask Documentation: https://flask.palletsprojects.com/en/2.0.x/