**Using ODBC (Open Database Connectivity) for Database Operations**

ODBC allows applications to interact with databases like Oracle, MySQL, SQL Server, etc., using a standardized API. Below, I'll show how to **connect to a database** and perform **basic operations** using **ODBC in Python**.

**By using JAVA**

### Download Java JDK

1. Go to the official Oracle JDK page:  
   <https://www.oracle.com/java/technologies/javase-downloads.html>
2. Scroll down to **Java SE Development Kit 21 LTS**

### Set Environment Variables

 Press **Win + S** and search for **"Environment Variables"** → Open “Edit the system environment variables”.

 In the **System Properties** window → Click **Environment Variables...**

 Under **System Variables**:

* Click **New**:
  + **Variable name**: JAVA\_HOME
  + **Variable value**: C:\Program Files\Java\jdk-21 *(or the actual path)*

### Verify Installation

java -version

**Using ODBC in Java for Database Operations (CRUD)**

Java uses **JDBC (Java Database Connectivity)** with **ODBC drivers** to connect to databases. Below is a Java program to perform **CRUD (Create, Read, Update, Delete)** operations using ODBC.

**Install Required Drivers**

Ensure you have the **ODBC Data Source** configured in your system:

* **Windows:** Go to *ODBC Data Sources (64-bit)* → Add → Select your database driver.
* **Linux/Mac:** Use odbc.ini for configuration.

**Java Code for ODBC Connection and Database Operations**

**Java Program**

import java.sql.\*;

public class OracleJDBCExample {

public static void main(String[] args) {

String url = "jdbc:oracle:thin:@localhost:1521:orcl"; // Oracle JDBC connection string

String user = "sit2"; // Database username

String password = "sit"; // Database password

try {

// Load the Oracle JDBC driver

Class.forName("oracle.jdbc.OracleDriver");

try (Connection conn = DriverManager.getConnection(url, user, password)) {

System.out.println(" Connected to Oracle database!");

// 1 Create Table

createTable(conn);

readData(conn); // Show table after creation

// 2 Insert Data

insertData(conn, 101, "Alice", 5000);

readData(conn); // Show table after insertion

insertData(conn, 102, "Bob", 3000);

readData(conn); // Show table after insertion

// 3 Read Data (already being shown after each action)

// 4 Update Data

updateData(conn, 101, 1000);

readData(conn); // Show table after update

// 5 Delete Data

deleteData(conn, 102);

readData(conn); // Show table after deletion

} catch (SQLException e) {

e.printStackTrace();

}

} catch (ClassNotFoundException e) {

System.out.println("Oracle JDBC Driver not found! Make sure ojdbc17.jar is in the classpath.");

e.printStackTrace();

}

}

// Method to create a table

public static void createTable(Connection conn) throws SQLException {

String sql = "CREATE TABLE Accounts (" +

"Account\_No INT PRIMARY KEY, " +

"Holder\_Name VARCHAR2(100), " + // VARCHAR2 is recommended for Oracle

"Balance NUMBER(10,2))"; // NUMBER(10,2) for better precision

try (Statement stmt = conn.createStatement()) {

stmt.executeUpdate(sql);

System.out.println(" Table 'Accounts' created successfully.");

} catch (SQLException e) {

System.out.println(" Table might already exist.");

}

}

// Method to insert data

public static void insertData(Connection conn, int accountNo, String holderName, double balance) throws SQLException {

String sql = "INSERT INTO Accounts (Account\_No, Holder\_Name, Balance) VALUES (?, ?, ?)";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setInt(1, accountNo);

pstmt.setString(2, holderName);

pstmt.setDouble(3, balance);

pstmt.executeUpdate();

System.out.println(" Inserted: " + holderName + " with Balance: " + balance);

}

}

// Method to read and display data

public static void readData(Connection conn) throws SQLException {

String sql = "SELECT \* FROM Accounts";

try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {

System.out.println("\n📋 Current Accounts Table:");

System.out.println("--------------------------------------");

System.out.printf("| %-10s | %-15s | %-10s |\n", "Account\_No", "Holder\_Name", "Balance");

System.out.println("--------------------------------------");

while (rs.next()) {

System.out.printf("| %-10d | %-15s | %-10.2f |\n",

rs.getInt("Account\_No"),

rs.getString("Holder\_Name"),

rs.getDouble("Balance"));

}

System.out.println("--------------------------------------\n");

}

}

// Method to update balance

public static void updateData(Connection conn, int accountNo, double amount) throws SQLException {

String sql = "UPDATE Accounts SET Balance = Balance + ? WHERE Account\_No = ?";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setDouble(1, amount);

pstmt.setInt(2, accountNo);

pstmt.executeUpdate();

System.out.println(" Updated balance for Account\_No: " + accountNo);

}

}

// Method to delete an account

public static void deleteData(Connection conn, int accountNo) throws SQLException {

String sql = "DELETE FROM Accounts WHERE Account\_No = ?";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setInt(1, accountNo);

pstmt.executeUpdate();

System.out.println(" Deleted Account\_No: " + accountNo);

}

}

}**How to Run the Program**

**1️ Set Up ODBC Data Source**

* Open **ODBC Data Source Administrator** (Windows).
* Click **Add** → Choose **SQL Server / MySQL / Oracle ODBC Driver**.
* Configure **DSN Name, Server, Username, Password**.
* Use this DSN in the **url = "jdbc:odbc:your\_dsn"** in the Java code.

**2️ Compile and Run in Terminal**

**javac -cp .;ojdbc17.jar OracleJDBCExample.java**

**java -cp .;ojdbc17.jar OracleJDBCExample**

**Expected Output**

Connected to database!

Table 'Accounts' created successfully.

Inserted: Alice with Balance: 5000.0

Inserted: Bob with Balance: 3000.0

Account: 101, Name: Alice, Balance: 5000.0

Account: 102, Name: Bob, Balance: 3000.0

Updated balance for Account\_No: 101

Deleted Account\_No: 102

**Explanation**

1. **Connection Setup**
   * Uses DriverManager.getConnection(url, user, password) to connect via ODBC.
2. **Create Table (Accounts)**
   * If not already created.
3. **Insert Data (Alice, Bob)**
   * Uses **PreparedStatement** for security.
4. **Retrieve Data**
   * Uses **ResultSet** to fetch and display records.
5. **Update Balance (Alice +1000)**
   * Modifies balance dynamically.
6. **Delete a Record (Bob’s account removed)**
   * Ensures clean deletion.