Lab-5

JDBC ile veritabanına bağlanıp sorgu yapma uygulamaları.

Geçen labı hatırlayalım..

- SORU : Ortalama 30 saatten fazla çalışılan projelerin isimlerini alfabetik olarak listeleyen sorguyu yazınız.

Geçen labı hatırlayalım..

- SORU : Ortalama 30 saatten fazla çalışılan projelerin isimlerini alfabetik olarak listeleyen sorguyu yazınız.

```
select pname
from project, works_on
where pnumber=pno
group by pname
having avg(hours)>30
order by pname
```

	pname character varying (25)
1	DatabaseSystems
2	InkjetPrinters
3	LaserPrinters
4	Middleware
5	OperatingSystems

JDBC
Java Database Connectivity

- JDBC, Java programımız ile veritabanımız arasındaki bağlantıyı sağlamak için kullandığımız bir Java API'dır.
- Bir java client'ı, bir veritabanı server ile JDBC library'si kullanarak iletişim kurar. JDBC class'ları bir java client'ı ile server arasındaki veri transferini yönetir.

PostgreSQL veritabanına uygun JDBC driver indirmek için:

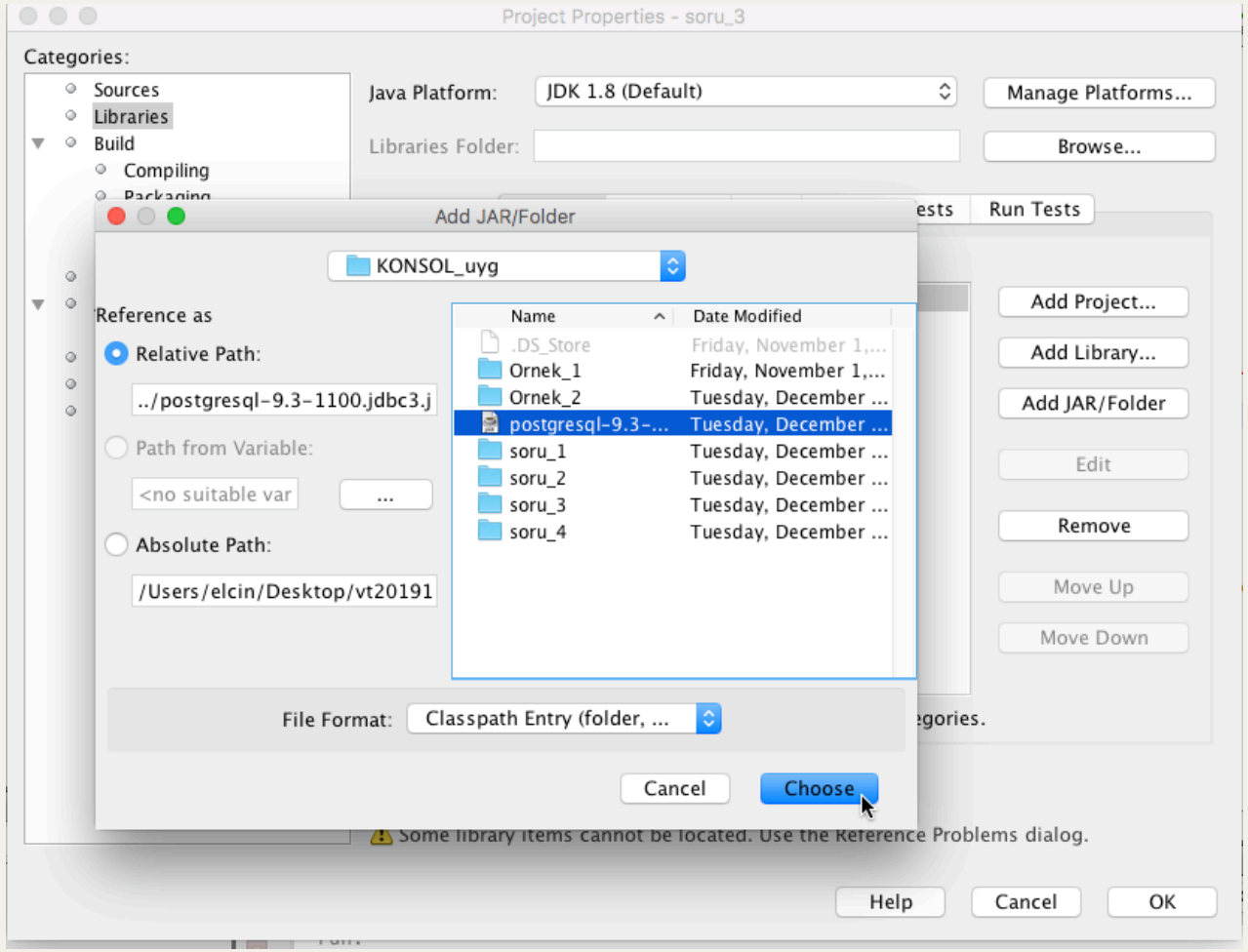
https://jdbc.postgresql.org/download.html

JDBC aşağıdaki interface ve sınıfları sağlamaktadır:

- DriverManager: Bu sınıf, veritabanı sürücülerinin listesini yönetir.
- Driver: Bu interface, veritabanı sunucusu ile iletişimi ele alır. Driver nesneleri ile çok nadir etkileşim kurabilirsiniz. Bunun yerine, bu türün nesnelerini yöneten DriverManager nesnesini kullanırsınız.
- Connection: Bu interface, bütün metotları ile veritabanına irtibat kurmak için kullanılır.
- Statement: SQL ifadelerini veritabanına göndermek için bu interface'ten oluşturulan nesneleri kullanırsınız.
- ResultSet: Statements nesnelerini kullanarak SQL sorgusunu çalıştırdıktan sonra veritabanından alınan verileri tutmak için bu nesneler kullanılır.
- SQLException:Bu sınıf, bir veritabanı uygulamasında ortaya çıkan hataları ele alır.

Projeye JDBC Eklenmesi

- Proje sağ tık
- Properties
  - **Java Build Path**
- Libraries
- Add JAR/Folder
- JDBC driver seçilir ve eklenir.



import java.sql.*;

Driver’ı ekledikten sonra , artık

import java.sql.*;

diyerek, veritabanımıza bağlanıp, sql sorgularımızı java’da yazabiliriz.

Kodun içinde veritabanına bağlanıp, sorgu çalıştırma;

```
Connection conn =
 DriverManager.getConnection("jdbc:postgresql://localhost:5432/company", “user_name”, “password”);
 Statement s = conn.createStatement();
 ResultSet r = s.executeQuery(query);
```

jdbc:postgresql://servername:serverport/databasename
(port numarası default olarak: 5432)

Connection → VT’ye irtibat kurmak için

Statement → SQL ifadelerini VT’ye göndermek için

ResultSet → Sonuçları almak için

Soru : "DatabaseSystems" projesinde çalışan kadın işçilerin ad, soyad ve maaşlarını listeleyin.

```
String query = "SELECT fname, lname, salary FROM employee e, dept_locations dl WHERE e.dno =
dl.dnumber AND dlocation = 'Chicago' ";
```

```
Connection conn =
 DriverManager.getConnection("jdbc:postgresql://localhost:5432/company", “postgres”, “1234”);
 Statement s = conn.createStatement();
 ResultSet r = s.executeQuery(query);
```

NOT: Statement sınıfına ait 2 metod;
executeQuery() : select ile başlayan sorguları çalıştırır.
executeUpdate() : veriler üzerinde güncelleme, ekleme, silme ifadelerini çalıştırır.

executeUpdate()

String updateString = "UPDATE employee SET fname = ‘John’ WHERE lname = ‘Smith’");

stmt.executeUpdate(updateString);

Update

Delete

Create Table

Insert Into

Tabloya ve sütunlara ait bilgileri öğrenmek için: getMetaData()

Kolon isimleri, sayıları, tipleri vs. Bilgilere erişmek için kullanırız.

Örneğin, 3.sıradaki sütunun tipini öğrenmek için;

```
ResultSetMetaData rsmd = r.getMetaData();
System.out.println(rsmd.getColumnTypeName(3));
```

Statement vs PreparedStatement

Statement durumunda sorgu her çalıştırıldığında veri tabanının belleğinde bu sorgunun bir örneği saklanır. Bu sorgunun binlerce kere çalıştırıldığını düşünürsek bu durum zararlı olabilir. Veri tabanı performansı düşebilir veya bağlantı kopmaları yaşanabilir.

Bu durumda PreparedStatement kullanmak faydalıdır. PreparedStatement durumunda çalıştırdığımızda veri tabanında bu sorgusunun sadece bir kere örneği saklanır ve bin kere de çalıştırsak bu sorgunun veri tabanının belleğinde sadece bir örneği tutulur. Böylece PreparedStatement daha performanslı olabilir.

SON