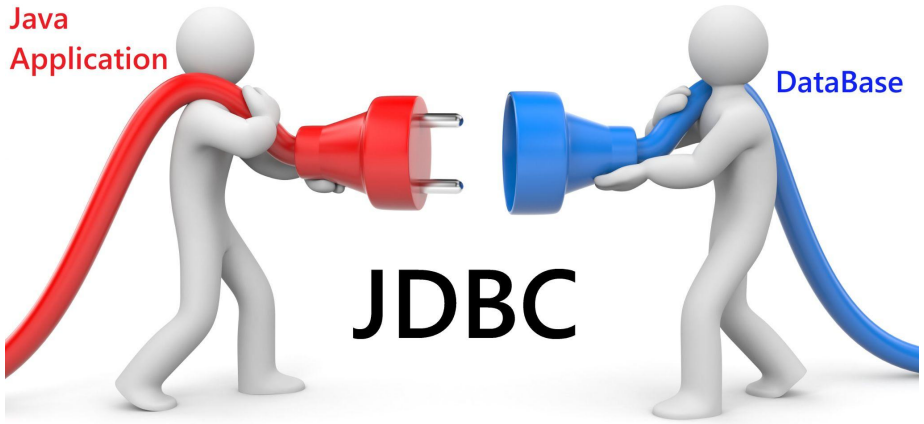
JDBC

Java DataBase Connectivity

12/03/2023

JDBC (Java DataBase Connectivity)

Java Database Connectivity (JDBC), Java programlama dilinde yazılmış uygulamaların veritabanı ile etkileşime girmesini sağlayan bir uygulama programlama arayüzüdür. JDBC ile hemen hemen tüm ilişkisel veri tabanı yönetim sistemlerine SQL sorgusu gönderilebilmektedir.



PostgreSQL

MySQL™

ORACLE®
DATABASE

Microsoft®
SQL Server®

DataBase Drivers

```
Class.forName( className: "org.postgresql.Driver");

Class.forName( className: "com.mysql.cj.jdbc.Driver");

Class.forName( className: "com.microsoft.sqlserver.jdbc.SQLServerDriver");

Class.forName( className: "oracle.cj.jdbc.driver.OracleDriver");
```

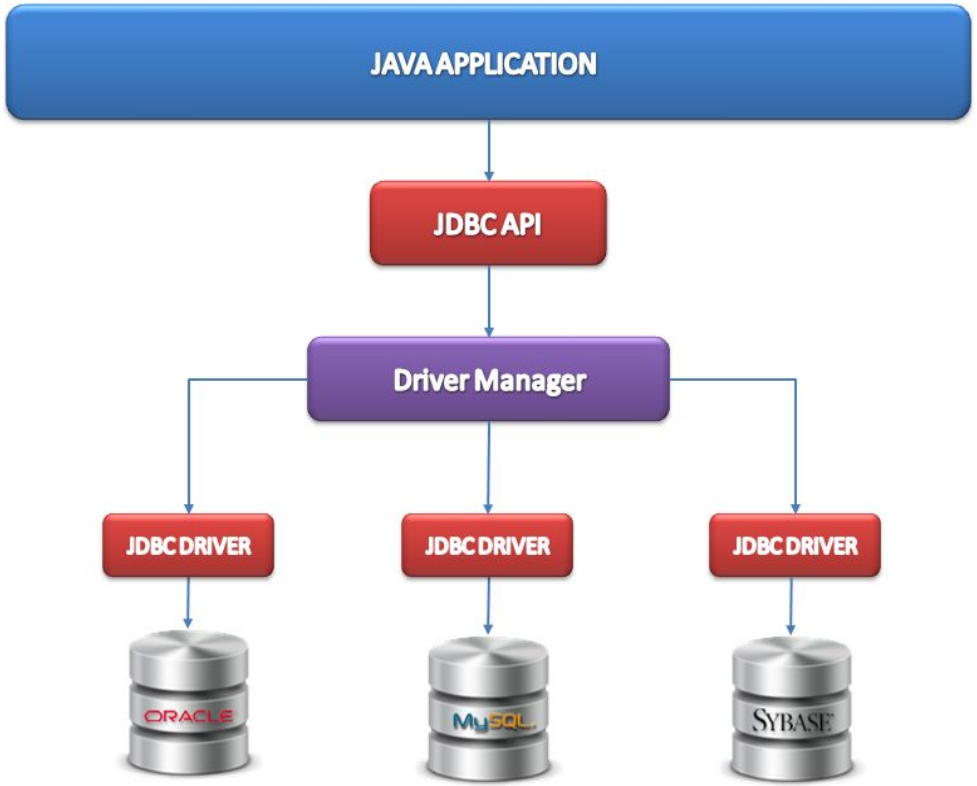
JDBC (Java DataBase Connectivity)

Driver: Bu arayüz veritabanı sunucusu ile olan iletişimi idare eder. Genellikle bu nesneyi yöneten DriverManager nesnesi üzerinden erişim yapılır.

```
Class.forName( className: "org.postgresql.Driver");
```

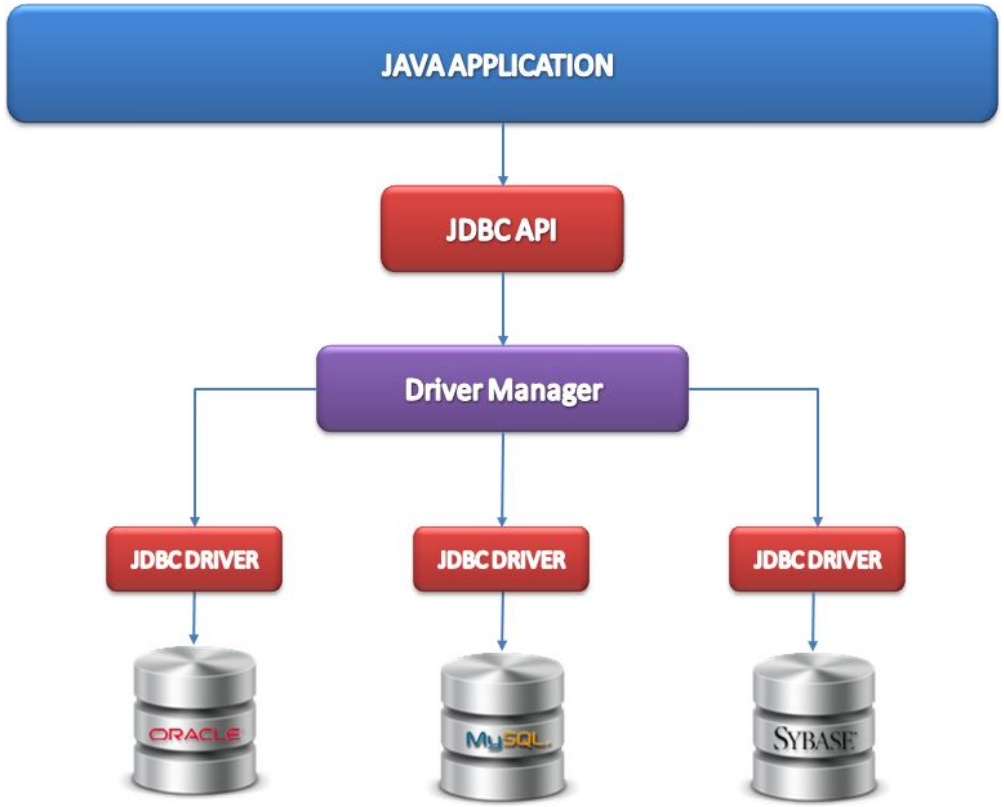
Connection: Bu arayüz, bir veritabanı ile iletişim kurmak için tüm yöntemleri içerir. Connection nesnesi iletişim bağlamını temsil eder, yani veritabanıyla yapılan tüm iletişim yalnızca bağlantı nesnesi aracılığıyla yapılır.

```
Connection con = DriverManager.getConnection(
        url: "jdbc:postgresql://localhost:5432/jdbc",
        user: "postgres",
        password: "12345");
```



DriverManager: Bu sınıf, veritabanı sürücülerinin bir listesini yönetir. Java uygulamasından gelen bağlantı istekleri ile uygun veritabanı sürücüsünü eşleştirir ve bağlantı oluşturur.

```
Connection con = DriverManager.getConnection(
        url: "jdbc:postgresql://localhost:5432/jdbc",
        user: "postgres",
        password: "12345");
```



Statement: Bu arayüz ile SQL ifadeleri veritabanına iletilir ve çalıştırılır.

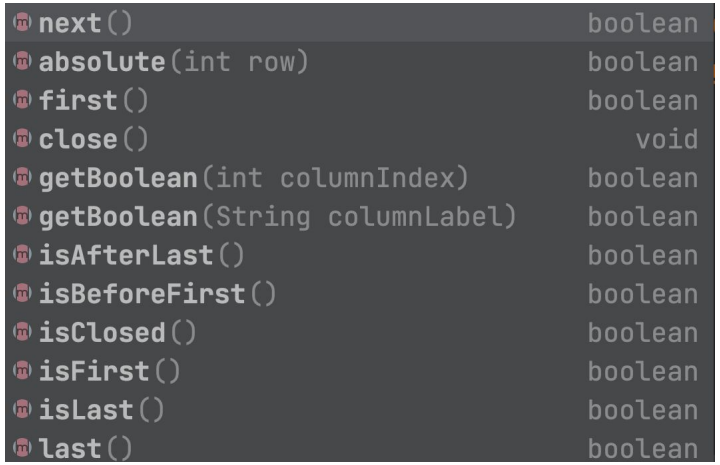
```
Statement st = con.createStatement();
```

JDBC (Java DataBase Connectivity)

ResultSet: JDBC kullanarak veri çekme işlemi sonrasında veri listelemek için ResultSet sınıfı kullanılır. SQL sorgusu çalıştırıldıktan sonra veritabanından alınan verileri saklar. Verilerin arasında gitmemizi sağlar.

```
ResultSet data = st.executeQuery( sql: "select * from students");
```

Veriler üzerinde dolaşmak için next, first, last, previous, absolute gibi metotlara sahiptir.

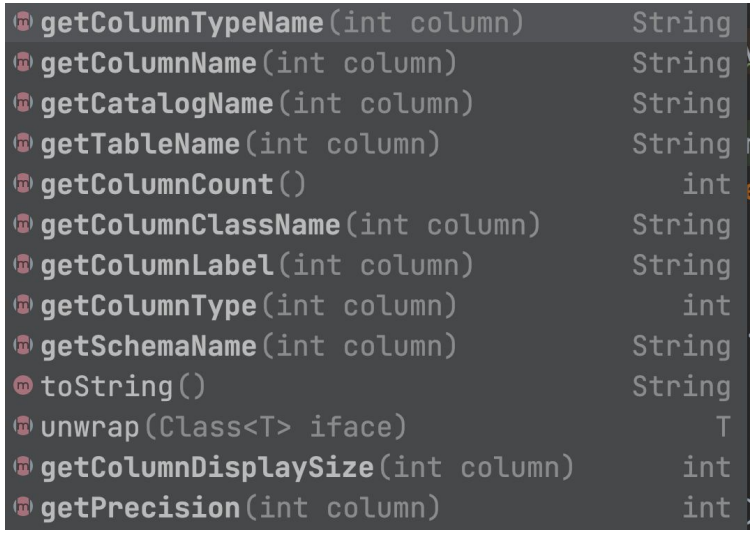


JDBC (Java DataBase Connectivity)

ResultSetMetaData: Bu arayüzü kullanarak, ResultSet hakkında daha fazla bilgi alabiliriz. Her ResultSet nesnesi, bir ResultSetMetaData nesnesiyle ilişkilendirilir. Bu nesne, sütunun veri türü, sütun adı, sütun sayısı, tablo adı, şema adı gibi sütun özelliklerinin ayrıntılarına verir. ResultSet'in, getMetaData() yöntemini kullanarak ResultSetMetaData nesnesini alabiliriz.

```
ResultSet rs = ps.executeQuery();

ResultSetMetaData rsmd = rs.getMetaData();
```



JDBC (Java DataBase Connectivity)

PreparedStatement: Yazdığımız herhangi bir SQL sorgusunu Statement ile çalıştırdığımızda; veri tabanının belleğinde bu sorgunun bir örneği saklanır. Bu sorgunun binlerce kere çalıştırıldığını düşünürsek; bu durum veritabanı performansını düşürür veya bağlantı kopmaları yaşanabilir.

Bu durumda PreparedStatement kullanmak faydalı olabilir. Herhangi bir SQL sorgusunu PreparedStatement ile 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 olur.