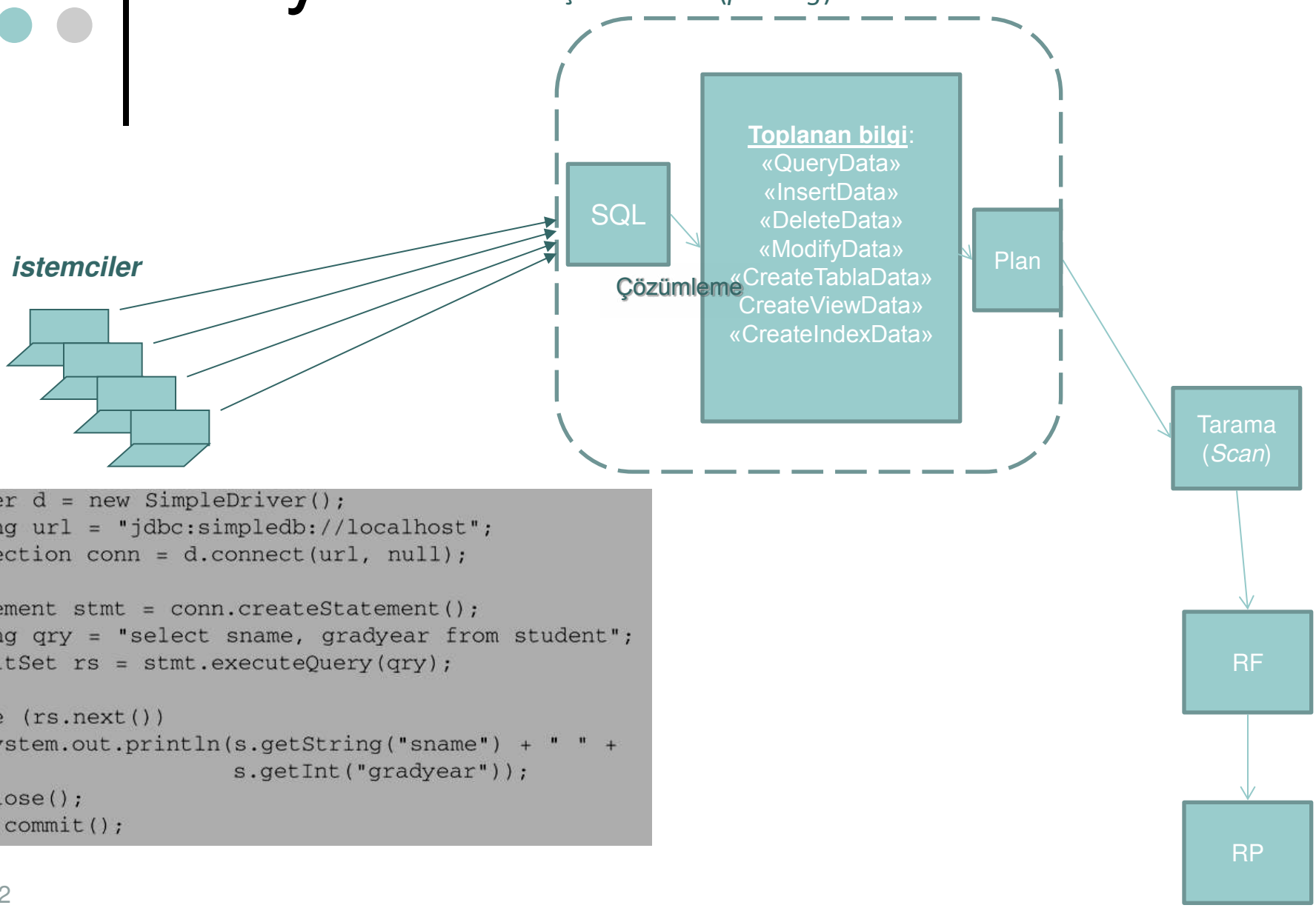
VT Sistem Gerçeklemesi Ders Notları- #10

Remote: Kullanıcılardan gelen JDBC isteklerini karşılar.
Planner: SQL ifadesi için işleme planı oluşturur ve karşılık gelen ilşkisel cebir ifadesini oluşturur.
Parse: SQL ifadesindeki tablo, nitelik ve ifadeleri ayrıştırır.
Query: Algebra ile ifade edilen sorguları gerçekler.
Metadata: Tablolara ait katalog bilgilerini organize eder.
Record: disk sayfalarına yazma/okumayı kayıt seviyesinde gerçekler.
Transaction&Recovery: Eşzamanlılık için gerekli olan disk sayfa erişimi kısıtlamalarını organize eder ve veri kurtarma için kayıt_defteri (*log*) dosyalarına bilgi girer.
Buffer: En sık/son erişilen disk sayfalarını ana hafıza tampon bölgede tutmak için gerekli işlemleri yapar.
Log: Kayıt_defterine bilgi yazılmasını ve taranması işlemlerini düzenler.
File: Dosya blokları ile ana hafıza sayfaları arasında bilgi transferini organize eder.

- servis sağlayan VT
- İstemci-servis sağlayıcı haberleşmesi:RMI
- RMI (*remote method invocation*) gerçekleme
- JDBC ara yüzlerinin gerçeklenmesi

Neredeyiz?




```
Driver d = new SimpleDriver();
String url = "jdbc:simpledb://localhost";
Connection conn = d.connect(url, null);

Statement stmt = conn.createStatement();
String qry = "select sname, gradyear from student";
ResultSet rs = stmt.executeQuery(qry);

while (rs.next())
   System.out.println(s.getString("sname") + " " +
                      s.getInt("gradyear"));
rs.close();
conn.commit();
```

Gömülü VT, servis sağlayıcı VT

```
SimpleDB.init("studentdb");
Transaction tx = new Transaction();

Planner planner = SimpleDB.planner();
String qry = "select sname, gradyear from student";
Plan p = planner.createQueryPlan(qry, tx);
Scan s = p.open();

while (s.next())
   System.out.println(s.getString("sname") + " " +
                      s.getInt("gradyear"));
s.close();
tx.commit();
```

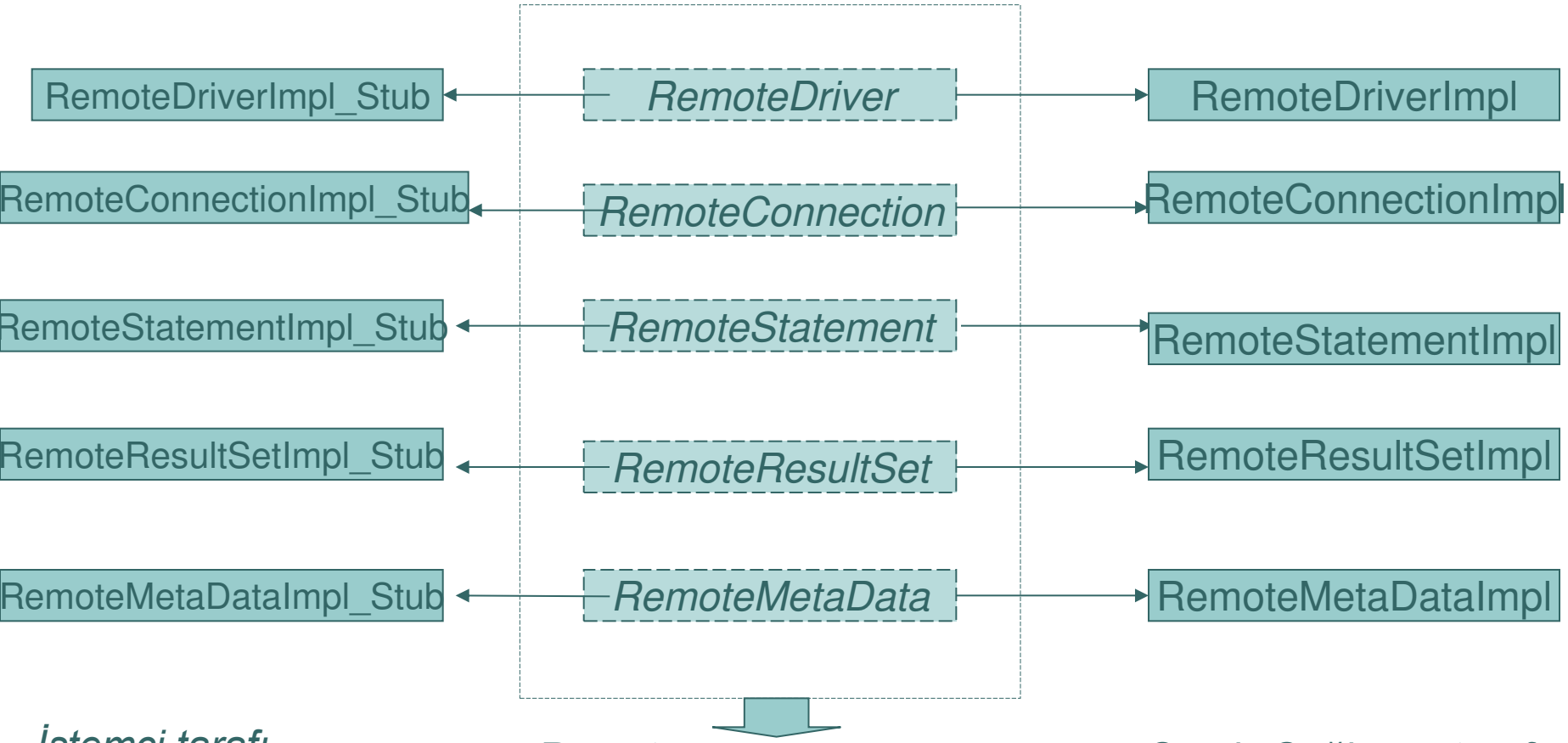
(a) Using an embedded database system

```
Driver d = new SimpleDriver();
String url = "jdbc:simpledb://localhost";
Connection conn = d.connect(url, null);

Statement stmt = conn.createStatement();
String qry = "select sname, gradyear from student";
ResultSet rs = stmt.executeQuery(qry);

while (rs.next())
   System.out.println(s.getString("sname") + " " +
                      s.getInt("gradyear"));
rs.close();
conn.commit();
```

İki farklı “gömülü vt kullanan” programın aynı veri tabanı üzerinde işlem yaptiğini düşünelim. Herhangi bir sorun var mı?

İki farklı istemcinin aynı veri tabanına, servis sağlayıcı üzerinden erişim, işlem yaptiğini düşünelim. Herhangi bir sorun var mı?

Remote Method Invocation (RMI)



İstemci tarafı gerçeklemesi

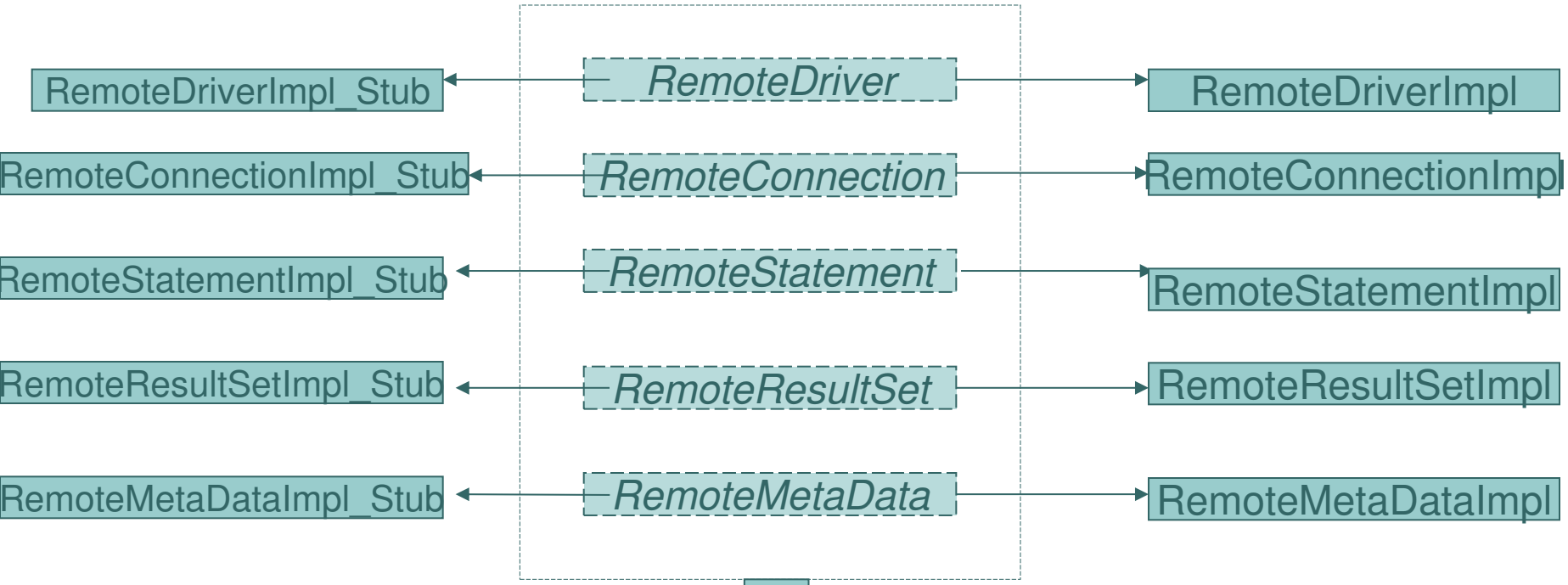
Remote arayüzünü extent eden arayüzler

Servis Sağlayıcı tarafı gerçeklemesi

Remote Method Invocation (RMI)

```
RemoteDriver rdvr = ...
RemoteConnection rconn = rdvr.connect();
RemoteStatement  rstmt = rconn.createStatement();
```

?????



İstemci tarafı gerçeklemesi

Remote arayüzünü extent eden arayüzler

Servis Sağlayıcı tarafı gerçeklemesi

```
RemoteDriver rdvr = ...
RemoteConnection rconn = rdvr.connect();
RemoteStatement  rstmt = rconn.createStatement();
```

?????

interface gerçekleme

Sistem başlatma:

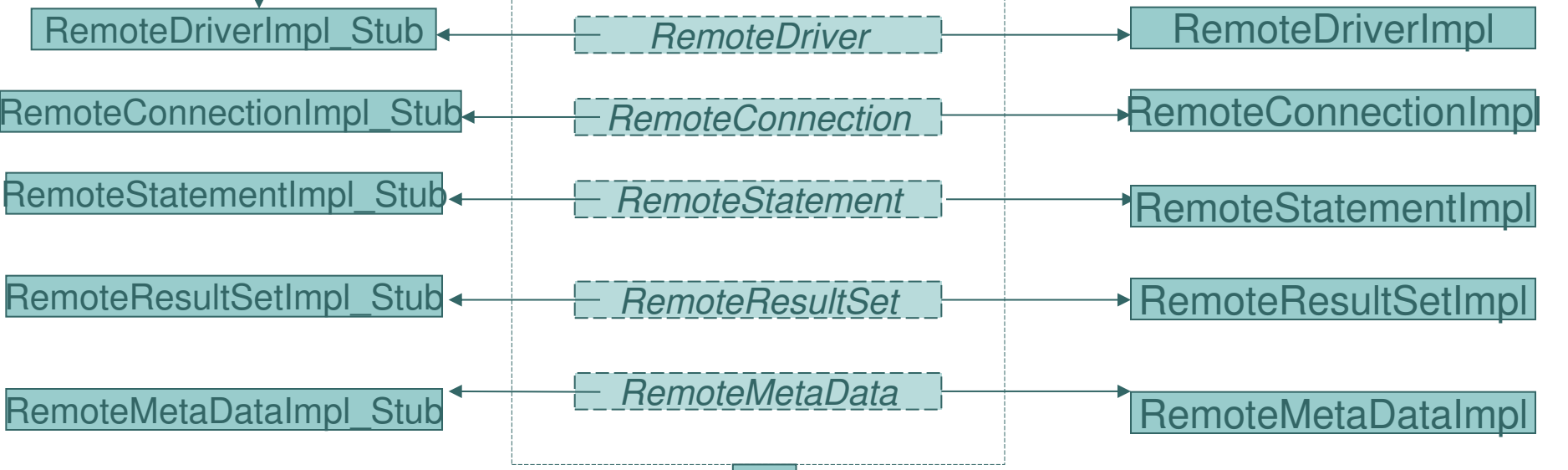
simpledb.server.Startup.java
RemoteDriver d= new RemoteDriverImpl();
Naming.rebind("simpledb",d);

RMI registry

RemoteDriver.class
(RemoteDriver stub class'ı)

simpledb.remote.SimpleDriver.java
String newurl = url.replace("jdbc:simpledb","rmi") + "/simpledb";
RemoteDriver rdvr = (RemoteDriver) Naming.lookup(newurl);

Database server thread

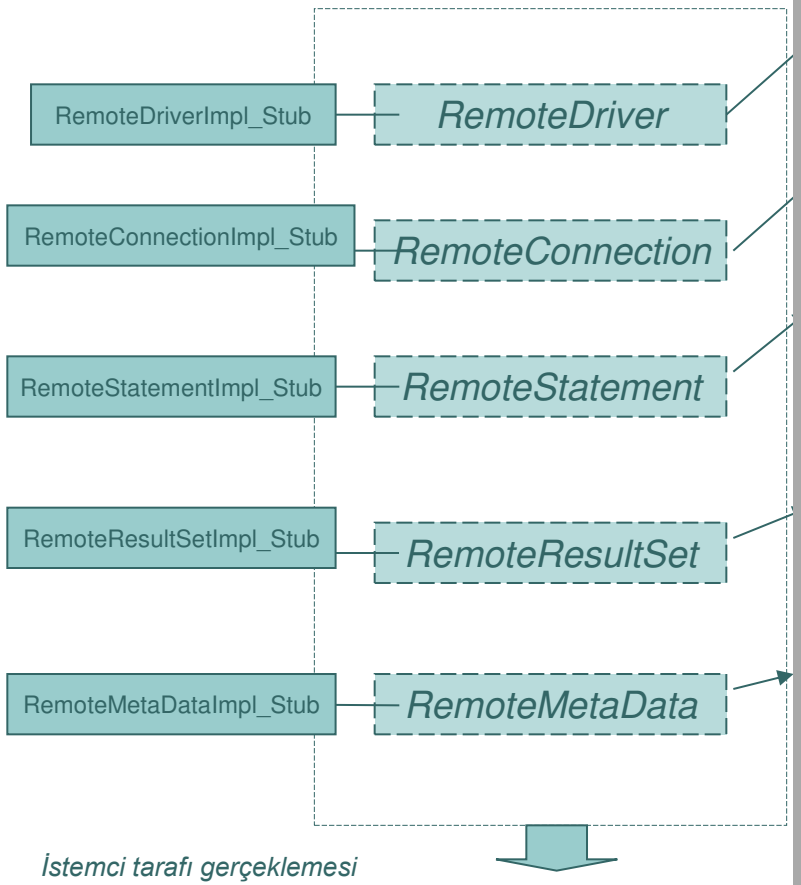


İstemci tarafı gerçeklemesi

Remote arayüzünü extent eden arayüzler

Servis Sağlayıcı tarafı gerçeklemesi

RMI (sunucu tarafı) gerçekleme



İstemci tarafı gerçeklemesi

Remote arayüzünü extent eden arayüzler

```
public class RemoteMetaDataImpl extends UnicastRemoteObject
                                implements RemoteMetaData {
   private Schema sch;
   private Object[] fieldnames;

   public RemoteMetaDataImpl(Schema sch)
                                     throws RemoteException {
      this.sch = sch;
      fieldnames = sch.fields().toArray();
   }
   public int getColumnCount() throws RemoteException {
      return fieldnames.length;
   }

   public String getColumnName(int column)
                                     throws RemoteException {
      return (String) fieldnames[column-1];
   }

   public int getColumnType(int column)
                                     throws RemoteException {
      String fldname = getColumnName(column);
      return sch.type(fldname);
   }

   public int getColumnDisplaySize(int column)
                                     throws RemoteException {
      String fldname = getColumnName(column);
      int    fldtype = sch.type(fldname);
      int  fldlength = sch.length(fldname);
      if (fldtype == INTEGER)
         return 6;   // accommodate 6-digit integers
      else
         return fldlength;
   }
}
}
```

Kullanıcı tarafı-JDBC ara yüzlerinin gerçeklenmesi

- JDBC'deki arayüzler:
  - Driver,Connection,ResultSet,ResultSetMetaData
- Bu standar arayüzlerdeki metodlar hem çok sayıda hem SQLException fırlatılıyor..
- Oysa bizim RMI gerçekleştirmelerimizde (otomatik olarak oluşan istemci tarafındaki stub sınıfları)
  - SQLException fırlatılmıyor.
  - JDBC arayüzündeki bütün metodlar gerçeklenmiyor..
- Bu ve benzeri sorunların nesne-dayalımlı programlamada çözümü: Ambalaj sınıflarıdır. (*wrapper methods)*
- İstemci tarafındaki Ambalaj sınıflarımız:
  - SimpleDriver, SimpleConnection, SimpleResultSet, SimpleResuştSetMetaData

Kullanıcı tarafı-JDBC ara yüzlerinin gerçeklenmesi

```
public class SimpleDriver extends DriverAdapter
                          implements Driver {
   public Connection connect(String host, Properties prop)
                                        throws SQLException {
      try {
         String newurl = url.replace("jdbc:simpledb","rmi")
                       + "/simpledb";
         RemoteDriver rdvr =
                   (RemoteDriver) Naming.lookup(newurl);
         RemoteConnection rconn = rdvr.connect();
         return new SimpleConnection(rconn);
      }
      catch (Exception e) {
         throw new SQLException(e);
      }
   }
}
```

```
public abstract class DriverAdapter implements Driver {

   public boolean acceptsURL(String url)
                                        throws SQLException {
      throw new SQLException("operation not implemented");
   }

   public Connection connect(String url, Properties info)
                                        throws SQLException {
      throw new SQLException("operation not implemented");
   }

   public int getMajorVersion() {
      return 0;
   }

   public int getMinorVersion() {
      return 0;
   }

   public DriverPropertyInfo[] getPropertyInfo(String url,
                                        Properties info) {
      return null;
   }

   public boolean jdbcCompliant() {
      return false;
   }
}
```

Kullanıcı tarafı-JDBC ara yüzlerinin gerçeklenmesi

```
public class SimpleConnection extends ConnectionAdapter
                              implements Connection {
   private RemoteConnection rconn;

   public SimpleConnection(RemoteConnection c) {
      rconn = c;
   }

   public Statement createStatement() throws SQLException {
      try {
         RemoteStatement rstmt = rconn.createStatement();
         return new SimpleStatement(rstmt);
      }
      catch(Exception e) {
         throw new SQLException(e);
      }
   }

   public void close() throws SQLException {
      try {
         rconn.close();
      }
      catch(Exception e) {
         throw new SQLException(e);
      }
   }
}
```