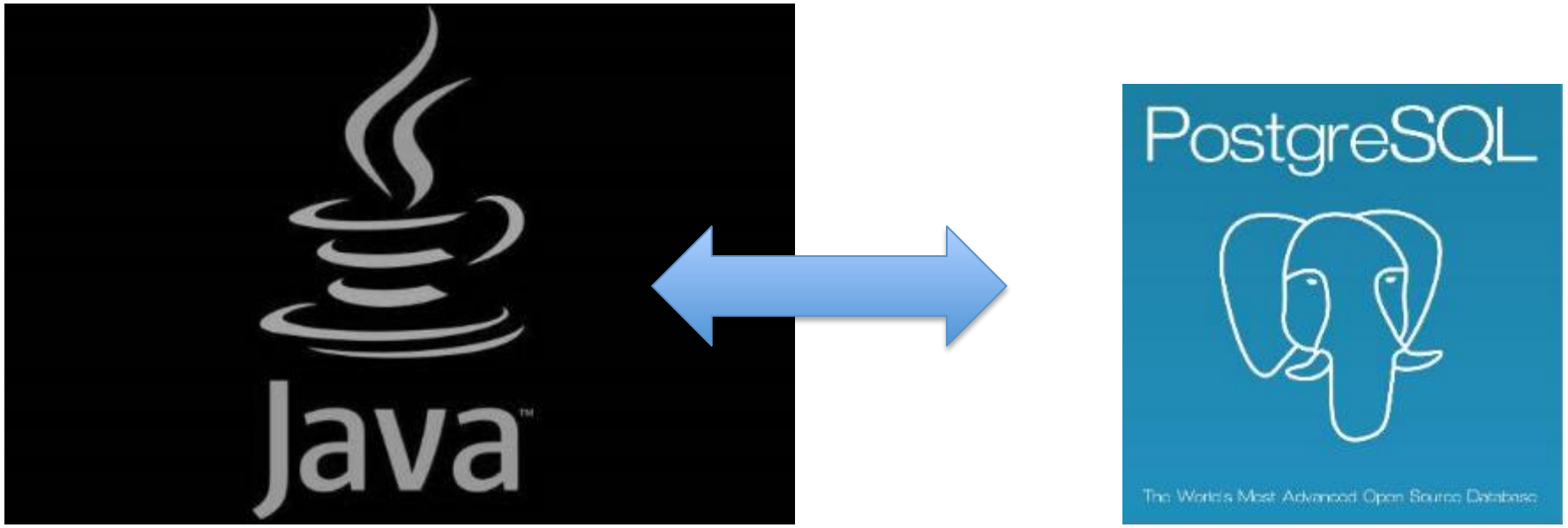
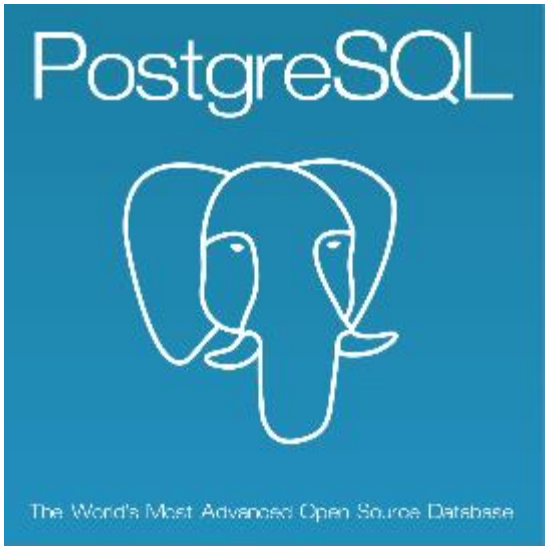
Bruke SQL fra Java

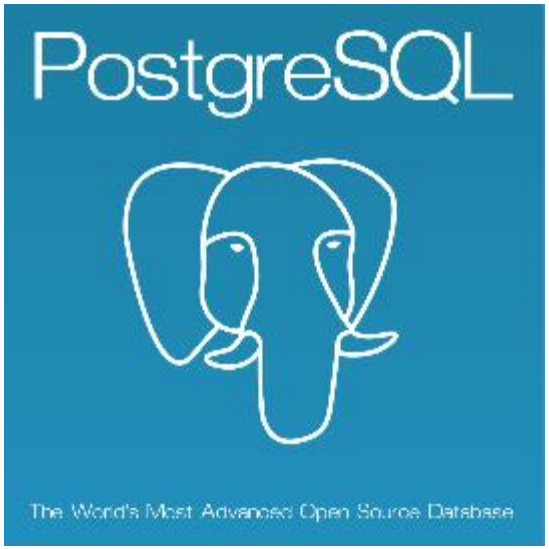
Med JDBC

(Java Database Connectivity)


Java
PostgreSQL
The World's Most Advanced Open Source Database

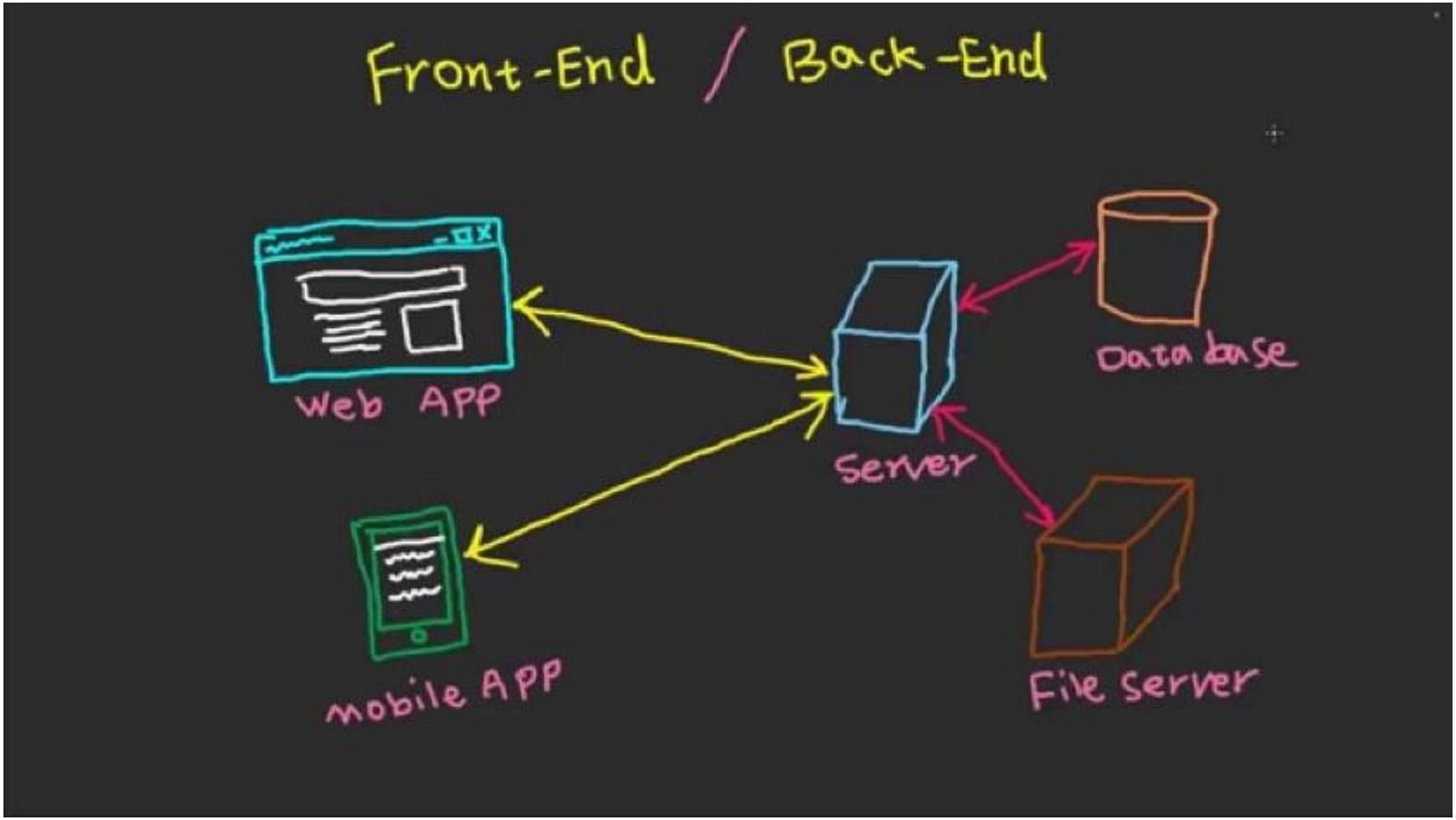

Java


PostgreSQL
The World's Most Advanced Open Source Database



Front-end

Back-end


Front-End / Back-End
Web App
Server
Database
mobile App
File Server

Package java.sql

Package java.sql (SE 7)

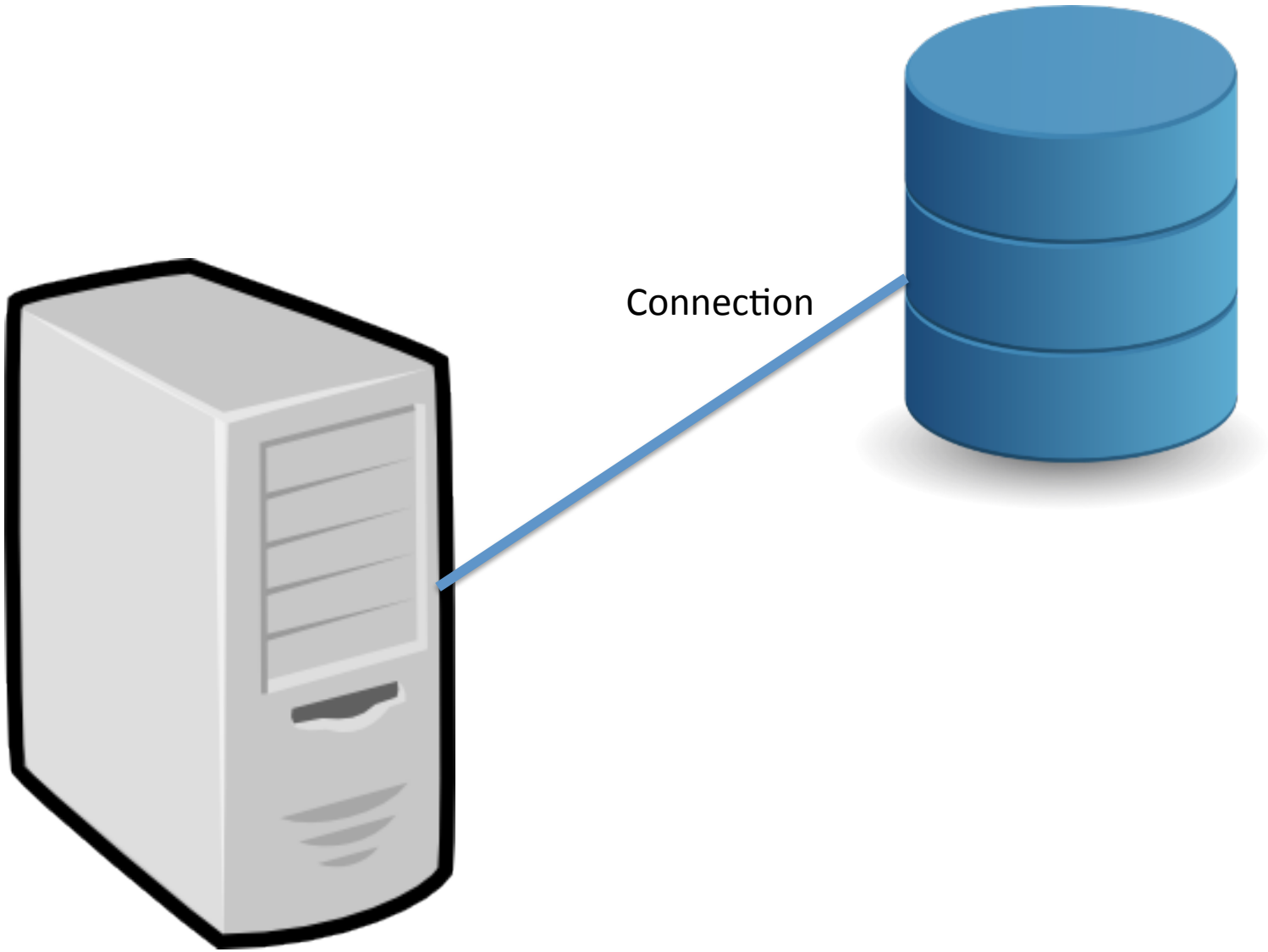
Array
Blob
CallableStatement
Clob
Connection
DatabaseMetaData
Driver
NClob
ParameterMetaData
PreparedStatement
Ref
ResultSet
ResultSetMetaData
RowId
Savepoint
SQLData
SQLInput
SQLOutput
SQLXML
Statement
Struct
Wrapper

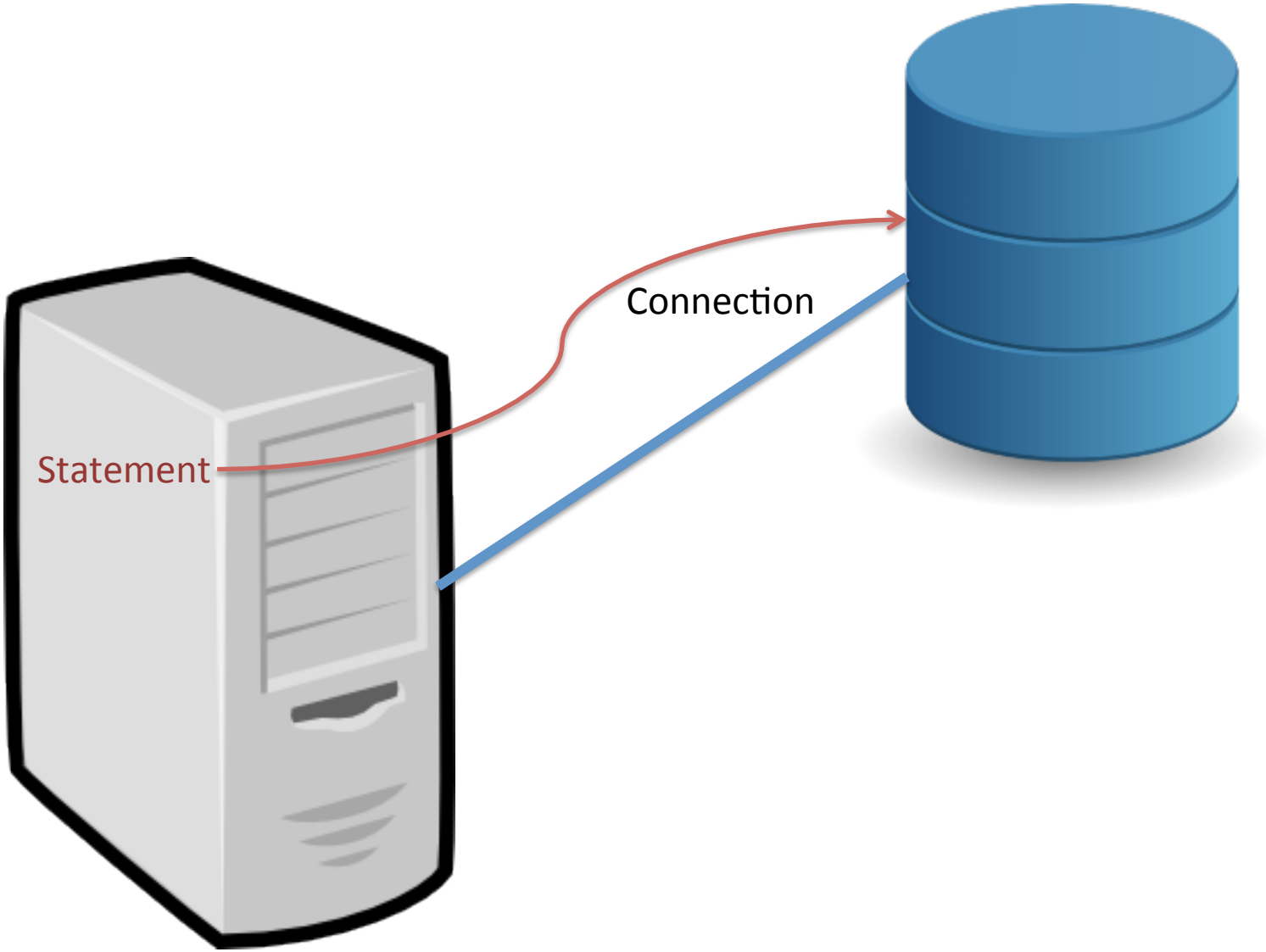
Package java.sql

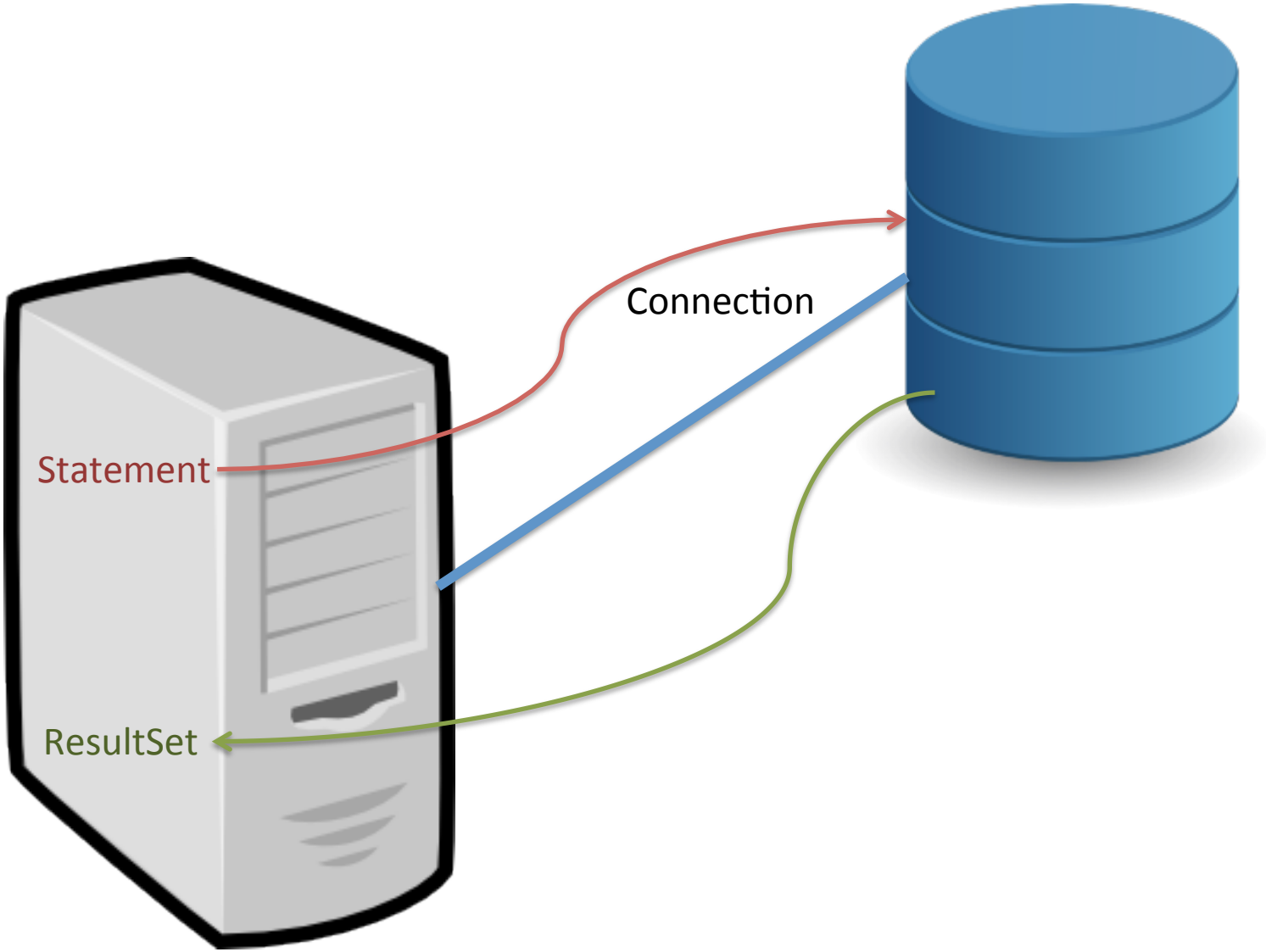
Array
Blob
CallableStatement
Clob
Connection
DatabaseMetaData
Driver
NClob
ParameterMetaData
PreparedStatement
Ref
ResultSet
ResultSetMetaData
RowId
Savepoint
SQLData
SQLInput
SQLOutput
SQLXML
Statement
Struct
Wrapper

Package java.sql

Connection
DatabaseMetaData
Driver
NClob
ParameterMetaData
PreparedStatement
Ref
ResultSet
ResultSetMetaData
RowId
Savepoint
SQLData
SQLInput
SQLOutput
SQLXML
Statement
Struct
Wrapper


Connection


Connection
Statement


Connection
Statement
ResultSet

Package java.sql

7 klasser i java.sql. Vi trenger én av dem:

DriverManager

The basic service for managing a set of JDBC drivers

Her finner vi metoder som gir oss en forbindelse. Dette skjer med metoder som returnerer (en peker til) et objekt av typen Connection.

Håndtere forbindelser mellom Java og PostgreSQL

I klassen DriverManager finner vi metoder som gir oss en forbindelse. Dette skjer med metoder som returnerer (en peker til) et objekt av typen Connection.

static Connection getConnection(String url, Properties info)

Håndtere forbindelser mellom Java og PostgreSQL

I klassen DriverManager finner vi metoder som gir oss en forbindelse. Dette skjer med metoder som returnerer (en peker til) et objekt av typen Connection.

static Connection getConnection(String url, Properties info)

DriverManager.getConnection(url, p);

```
DriverManager.getConnection(url, p);
```

DriverManager.getConnection(url, p);

```
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";
```

DriverManager.getConnection(url, p);

```
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

Properties p = new Properties();
```

java.util

Class Properties

java.lang.Object
  java.util.Dictionary<K,V>
    java.util.Hashtable<Object,Object>
      java.util.Properties

All Implemented Interfaces:

Serializable, Cloneable, Map<Object,Object>

Direct Known Subclasses:

Provider

```
public class Properties
extends Hashtable<Object,Object>
```

DriverManager.getConnection(url, p);

```
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

Properties p = new Properties();

p.setProperty("ssl", "true");
```

DriverManager.getConnection(url, p);

```
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

Properties p = new Properties();

p.setProperty("ssl", "true");

p.setProperty("sslfactory",
                    "org.postgresql.ssl.NonValidatingFactory");
```

DriverManager.getConnection(url, p);

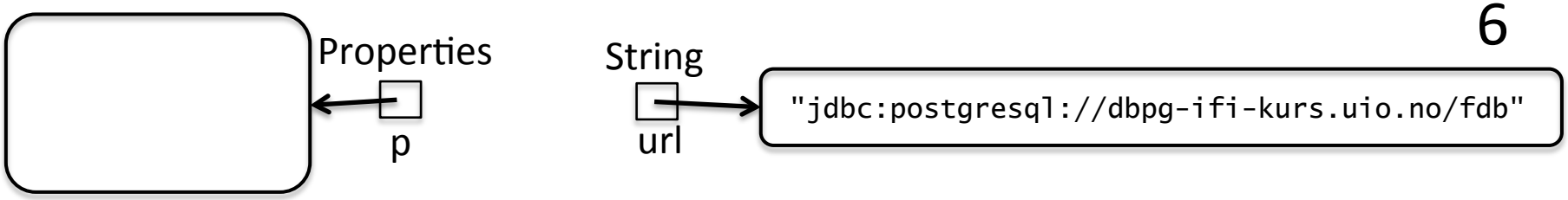
```
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

Properties p = new Properties();

p.setProperty("ssl", "true");

p.setProperty("sslfactory",
                  "org.postgresql.ssl.NonValidatingFactory");

p.put("user", "michael");
```

DriverManager.getConnection(url, p);

```
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

Properties p = new Properties();

p.setProperty("ssl", "true");

p.setProperty("sslfactory",
                  "org.postgresql.ssl.NonValidatingFactory");

p.put("user", "michael");

p.put("password", passord);
```

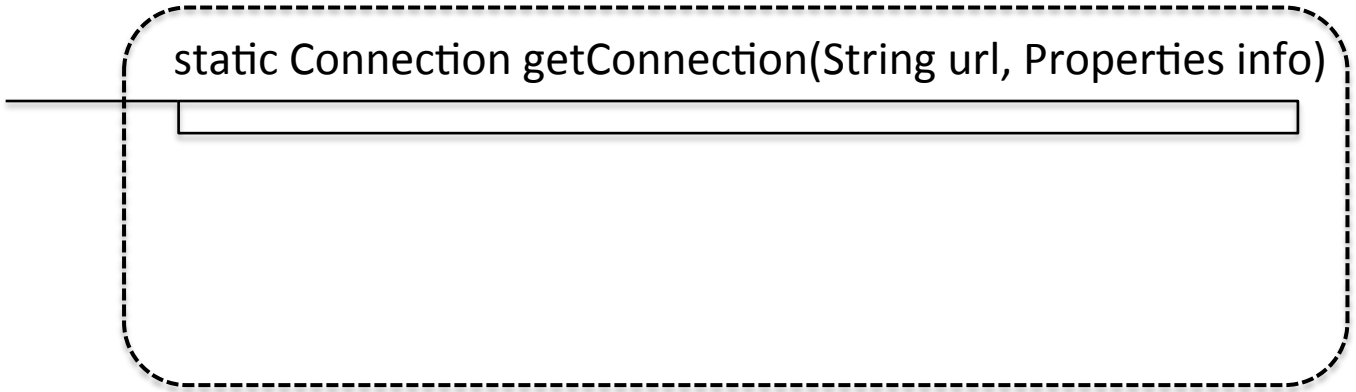
DriverManager.getConnection(url, p);

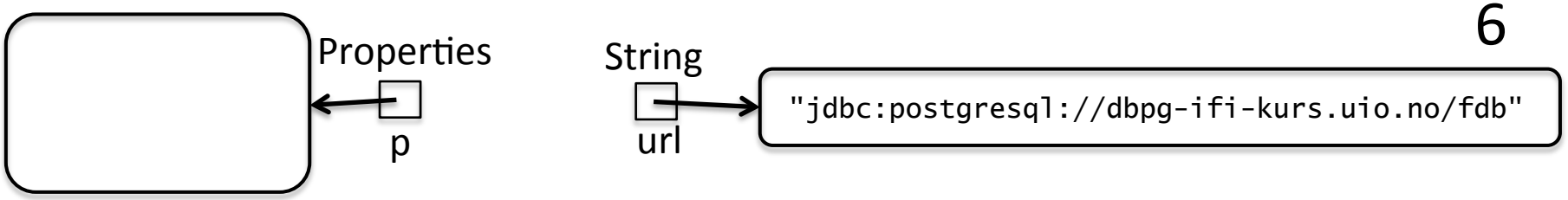
```
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

Properties p = new Properties();

p.setProperty("ssl", "true");

p.setProperty("sslfactory",
                  "org.postgresql.ssl.NonValidatingFactory");

p.put("user", "michael");

p.put("password", passord);
```



DriverManager.getConnection(url, p);

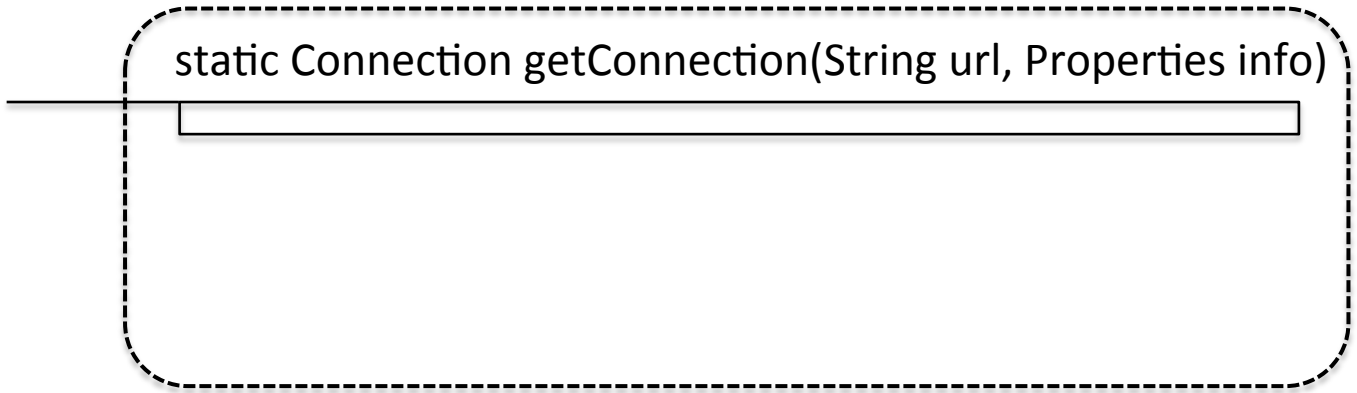
Klassedatastruktur for klassen DriverManager

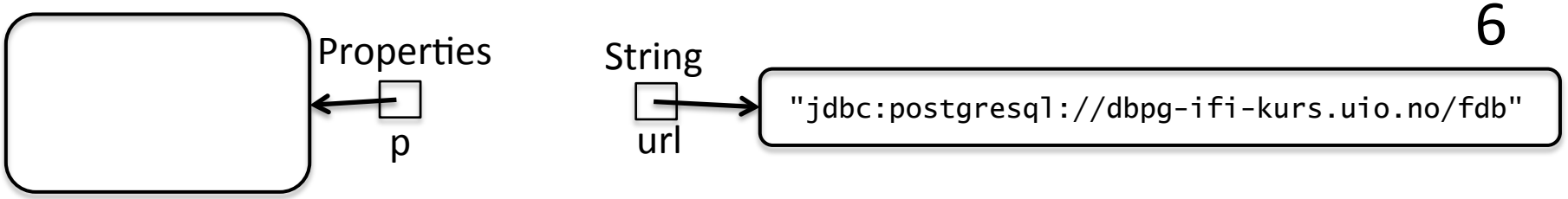




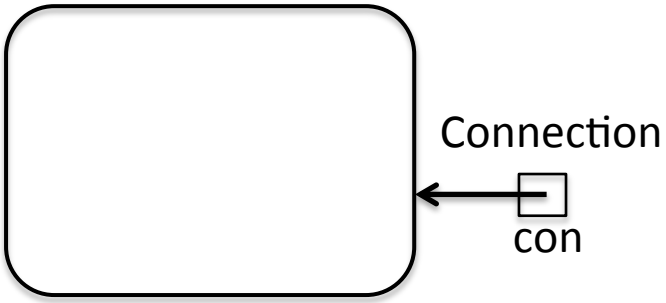
Connection con = DriverManager.getConnection(url, p);

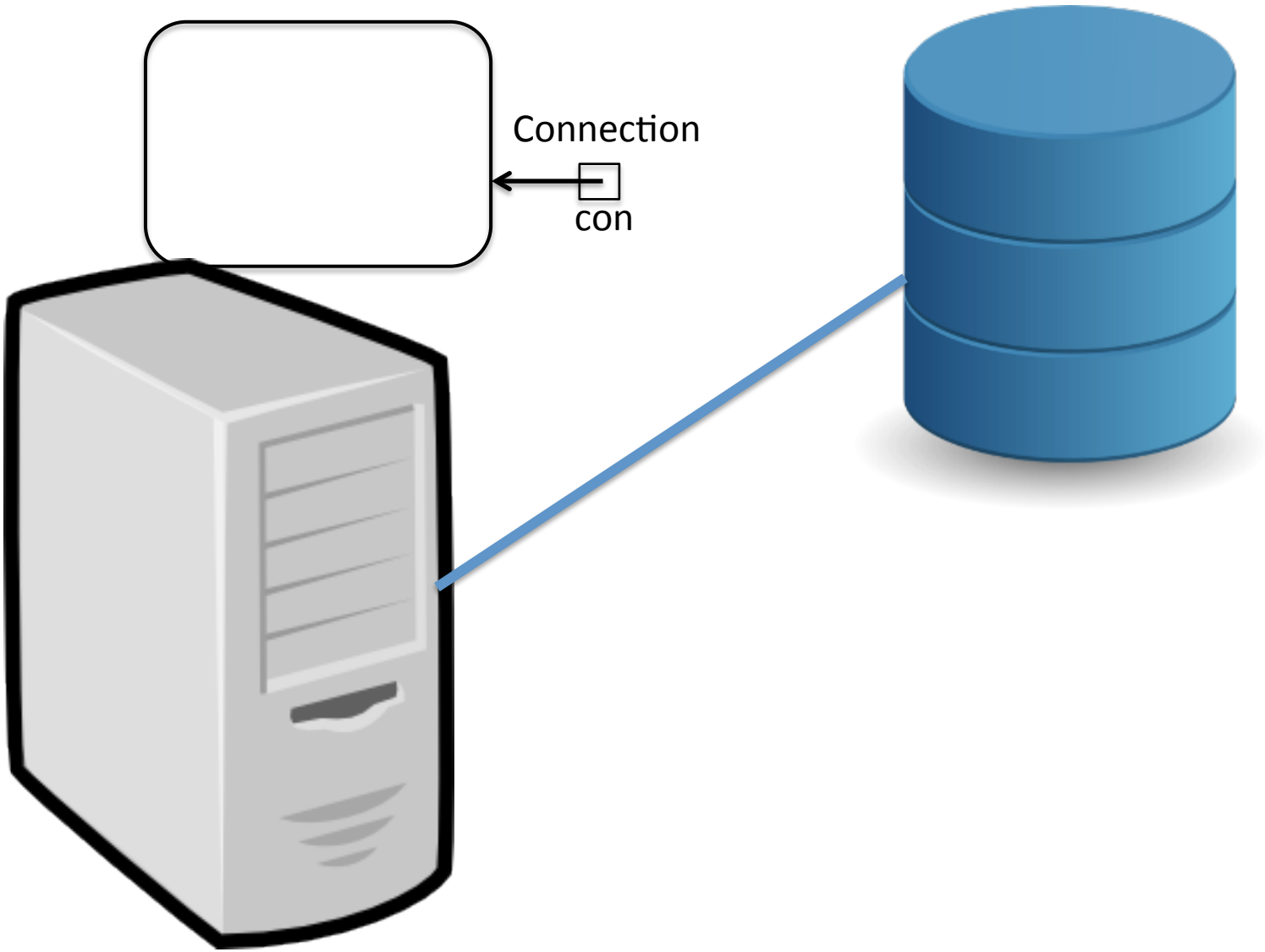
Klassedatastruktur for klassen DriverManager

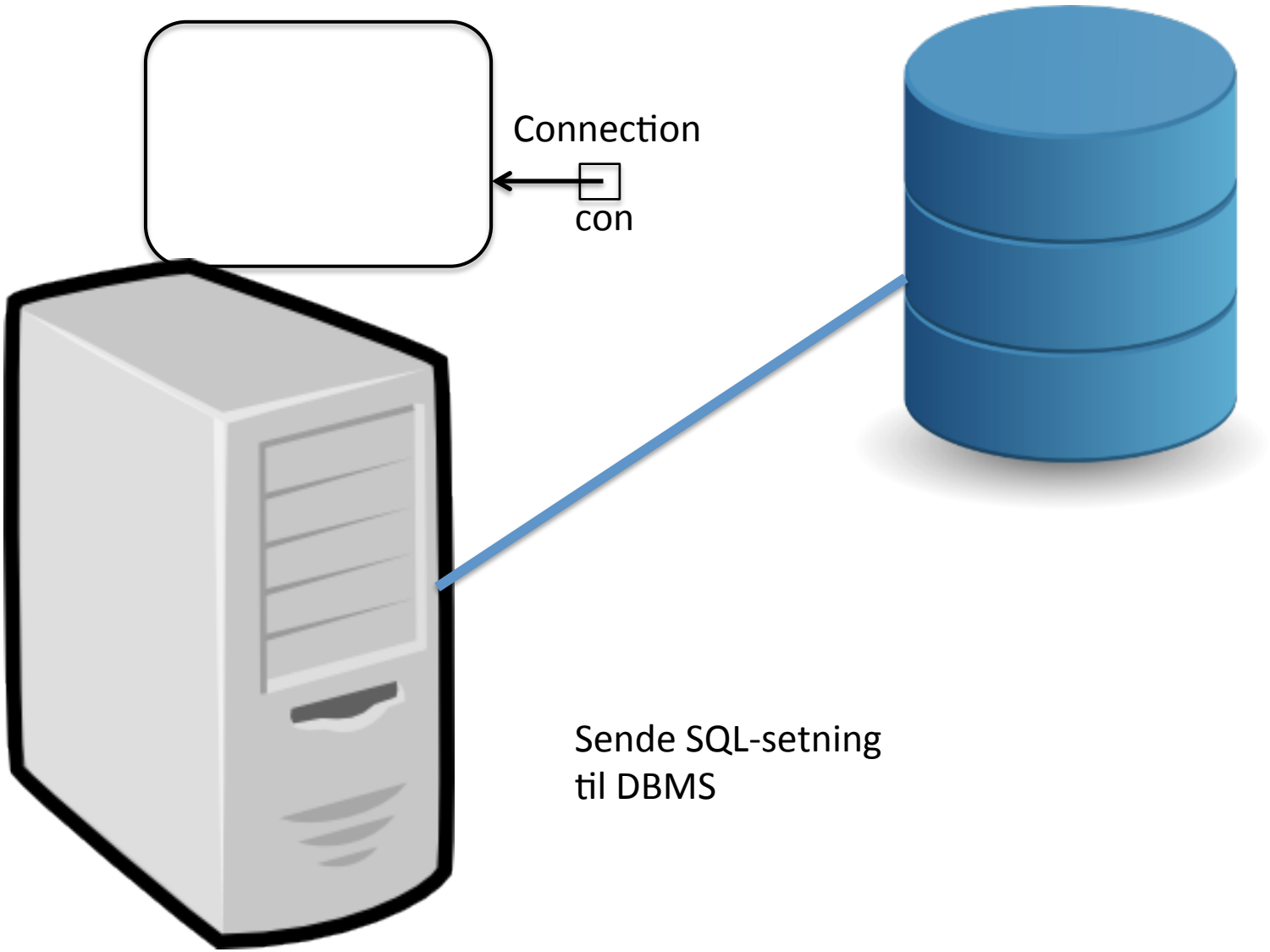




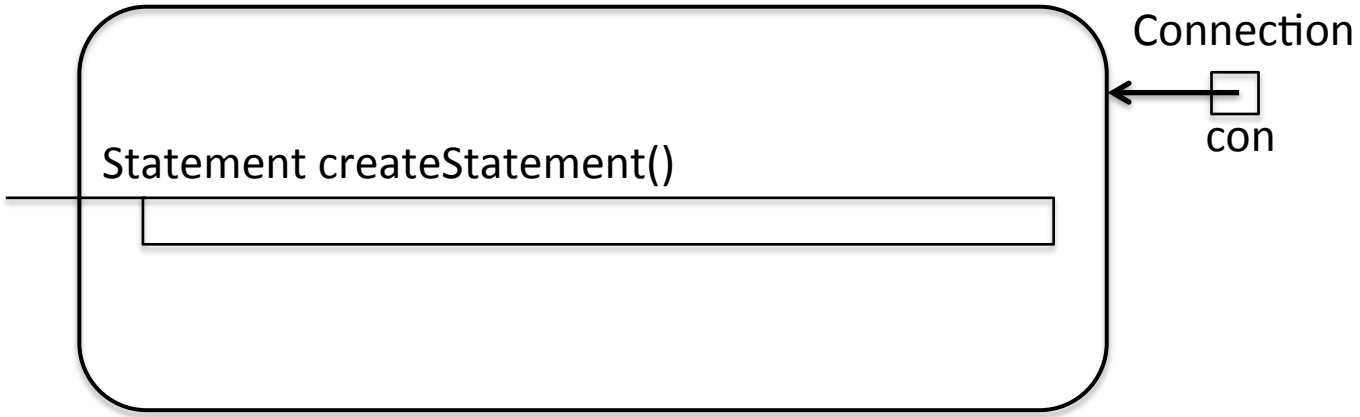
Connection con = DriverManager.getConnection(url, p);



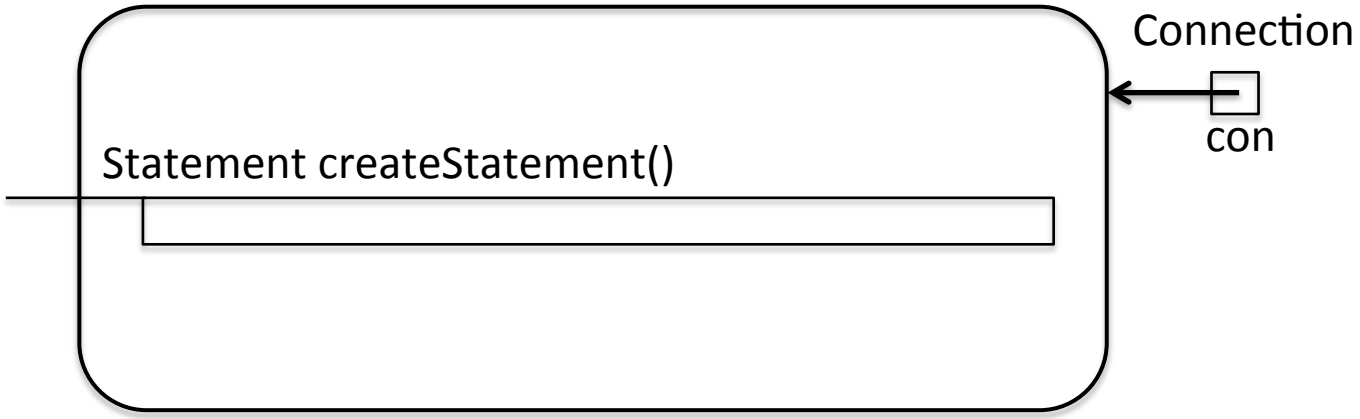

Connection
con


Connection
con
Sende SQL-setning
til DBMS

Kjøre SQL-kommandoer: Statement-klassen

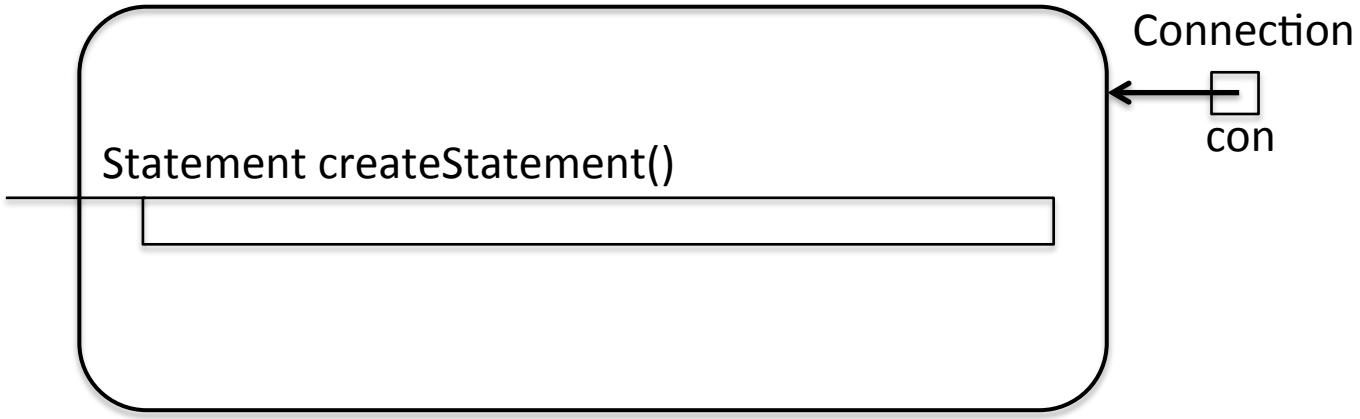


Kjøre SQL-kommandoer: Statement-klassen



/* Creates a Statement object for sending
SQL statements to the database. */

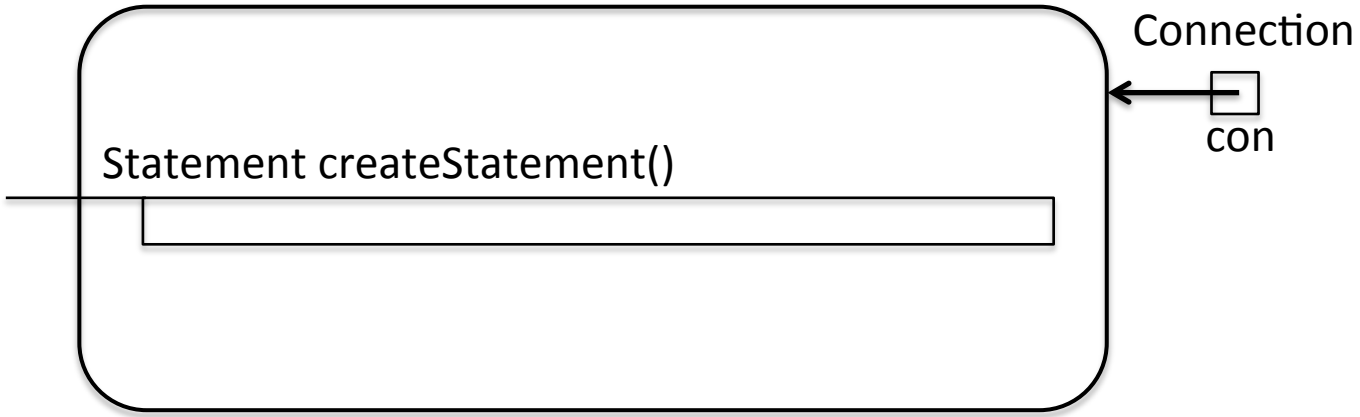
Kjøre SQL-kommandoer: Statement-klassen



/* Creates a Statement object for sending
SQL statements to the database. */

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

Kjøre SQL-kommandoer: Statement-klassen

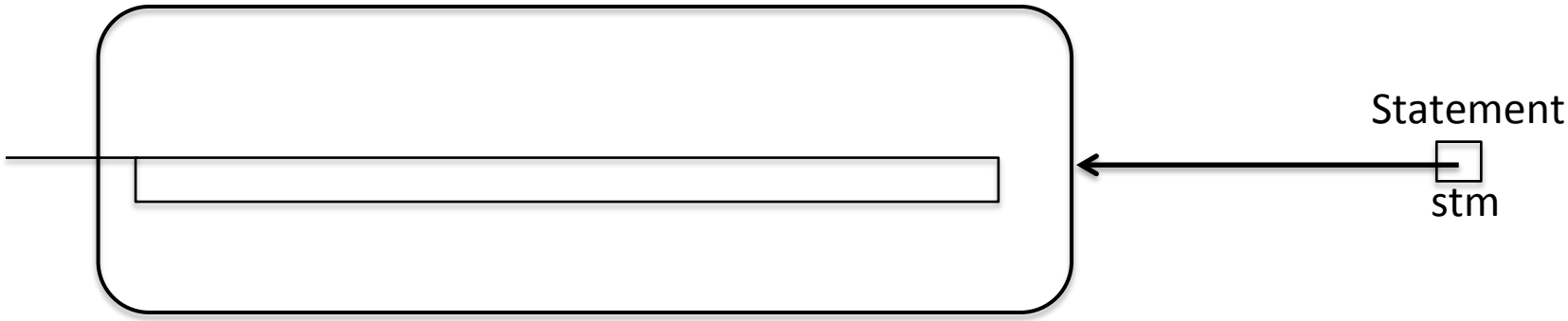


/* Creates a Statement object for sending
SQL statements to the database. */

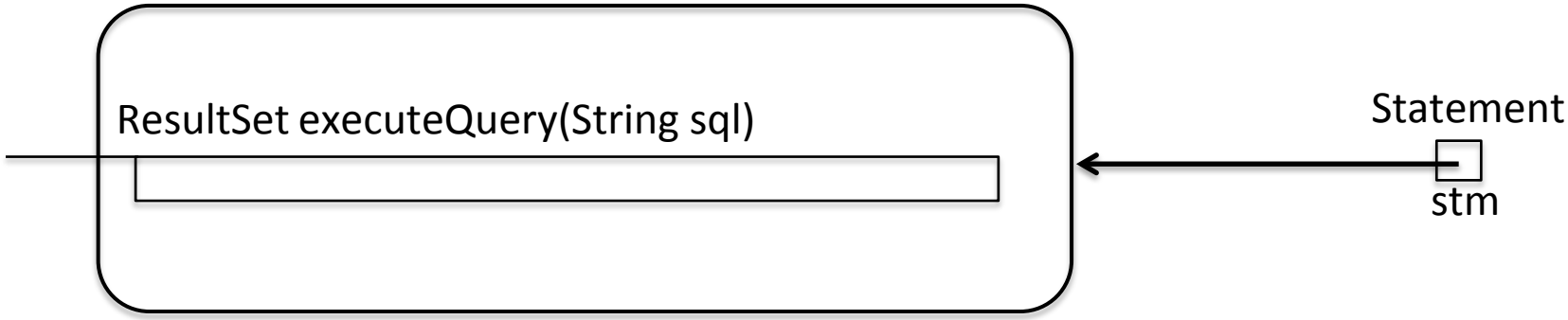
```
Statement stm = con.createStatement();
```

Kjøre SQL-kommandoer: Statement-objektet

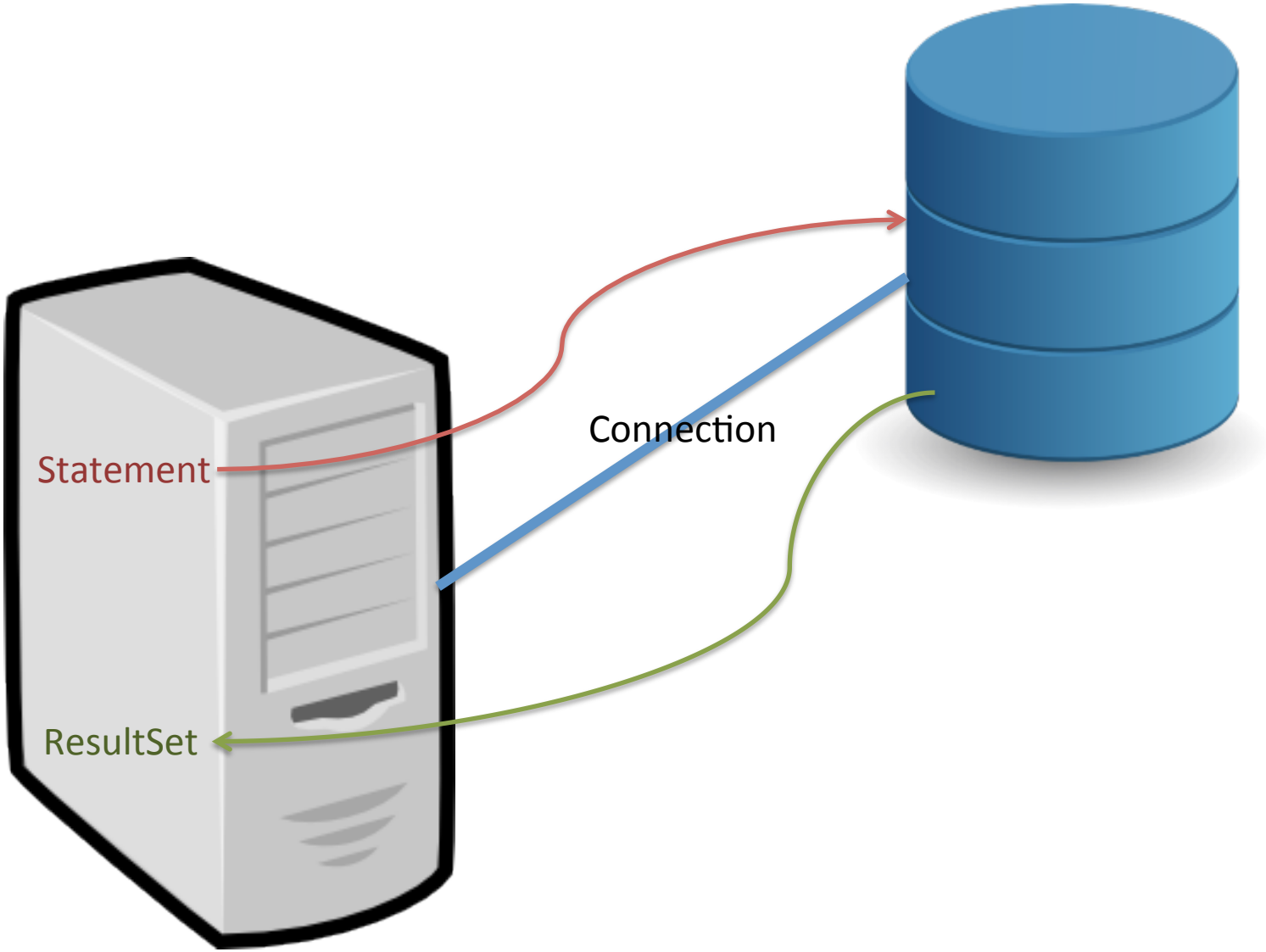
Kjøre SQL-kommandoer: Statement-objektet



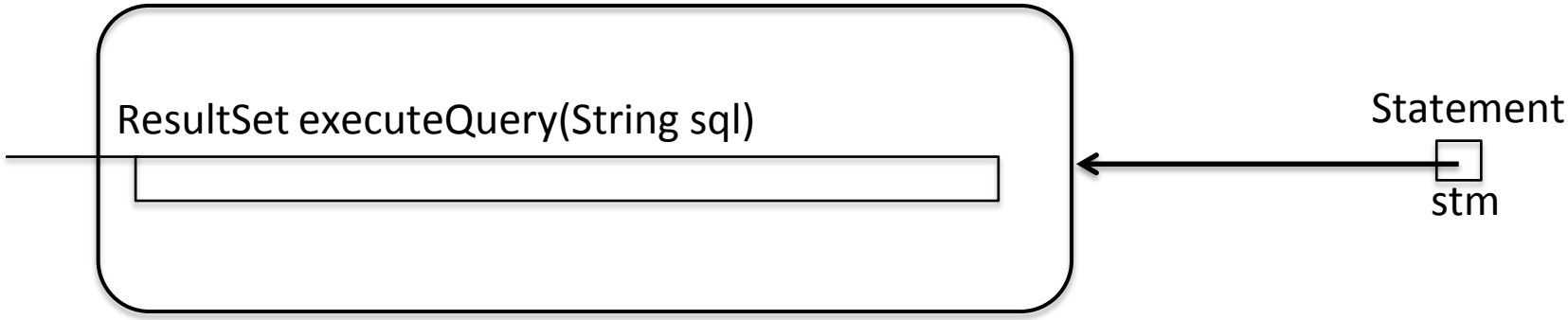
Kjøre SQL-kommandoer: Statement-objektet



Executes the given SQL statement,
which returns a single ResultSet object.

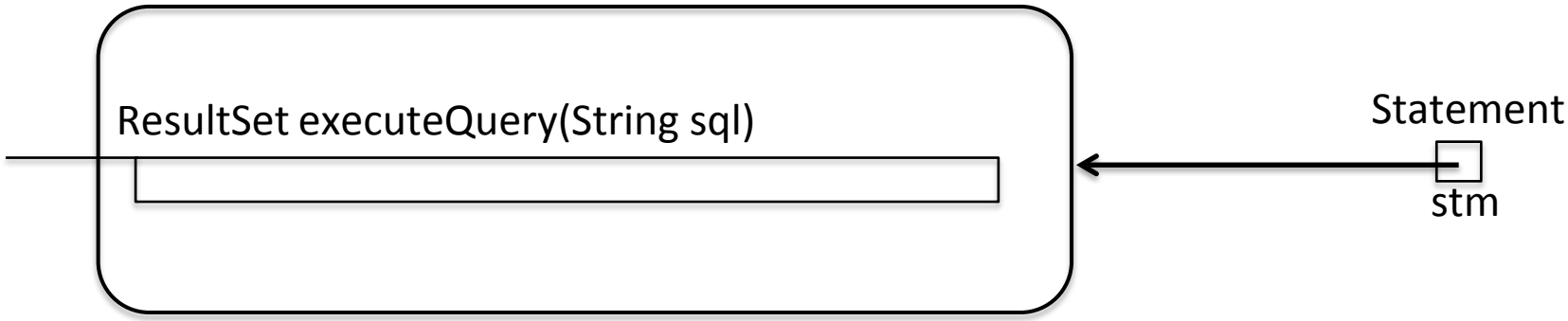

Statement
Connection
ResultSet

Utføre spørringer: Statement-objektet



String spørring = " select F.filmid as fid, F.title as tittel, F.prodyear as paar
from film F natural join Filmcountry C where C.country='Norway' " ;

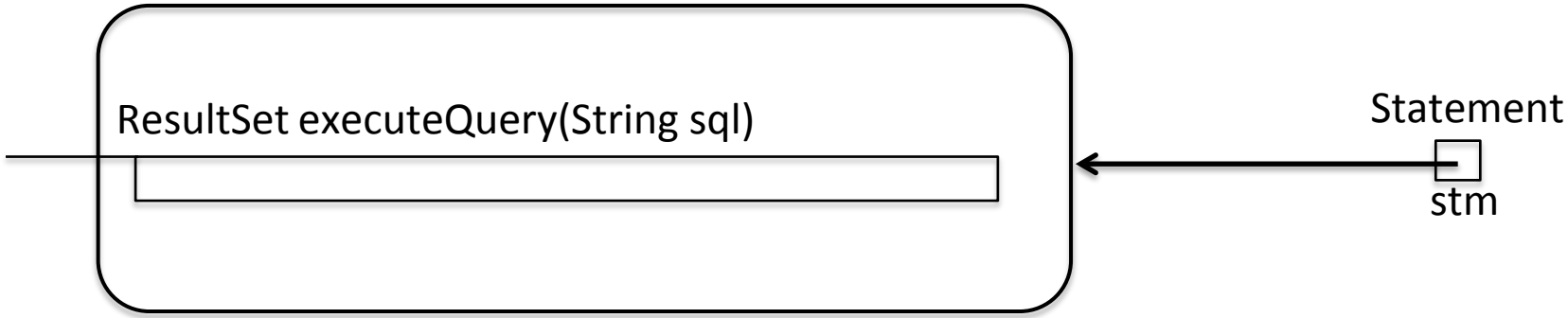
Utføre spørringer: Statement-objektet



String spørring = " select F.filmid as fid, F.title as tittel, F.prodyear as paar
from film F natural join Filmcountry C where C.country='Norway' " ;

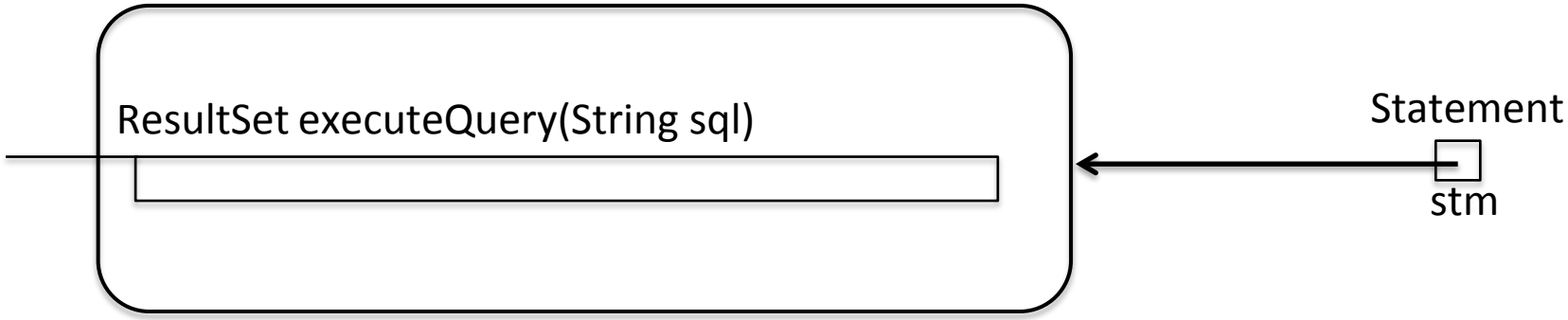
```
String spørring = "select F.filmid as fid, "
                  +"       F.title as tittel, "
                  +"       F.prodyear as paar "
                  +"from film F natural join Filmcountry C "
                  +"where C.country='Norway' " ;
```

Utføre spørringer: Statement-objektet

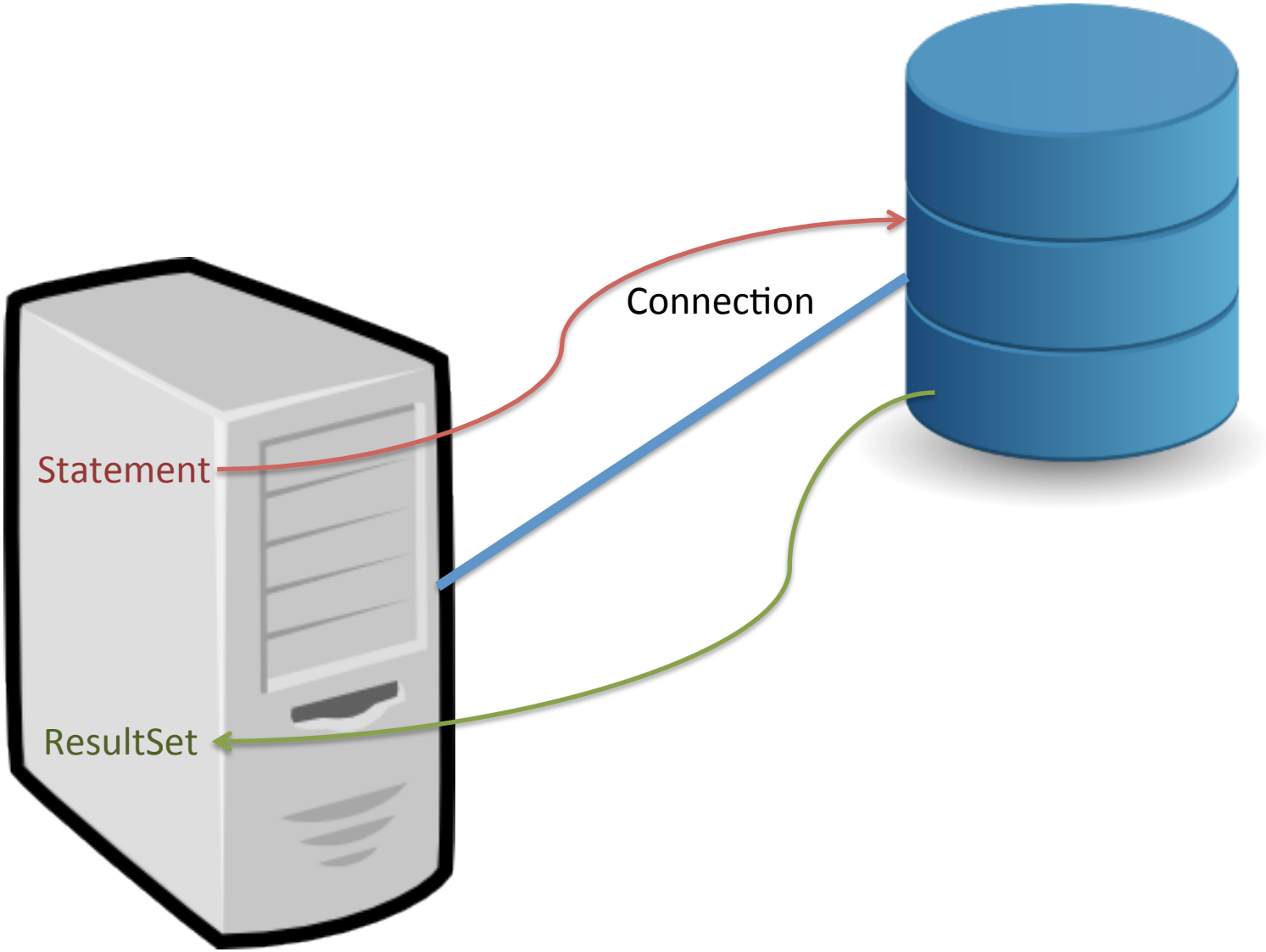



```
String spørring = "select F.filmid as fid, "
                  +"       F.title as tittel, "
                  +"       F.prodyear as paar "
                  +"from film F natural join Filmcountry C "
                  +"where C.country='Norway' " ;
```

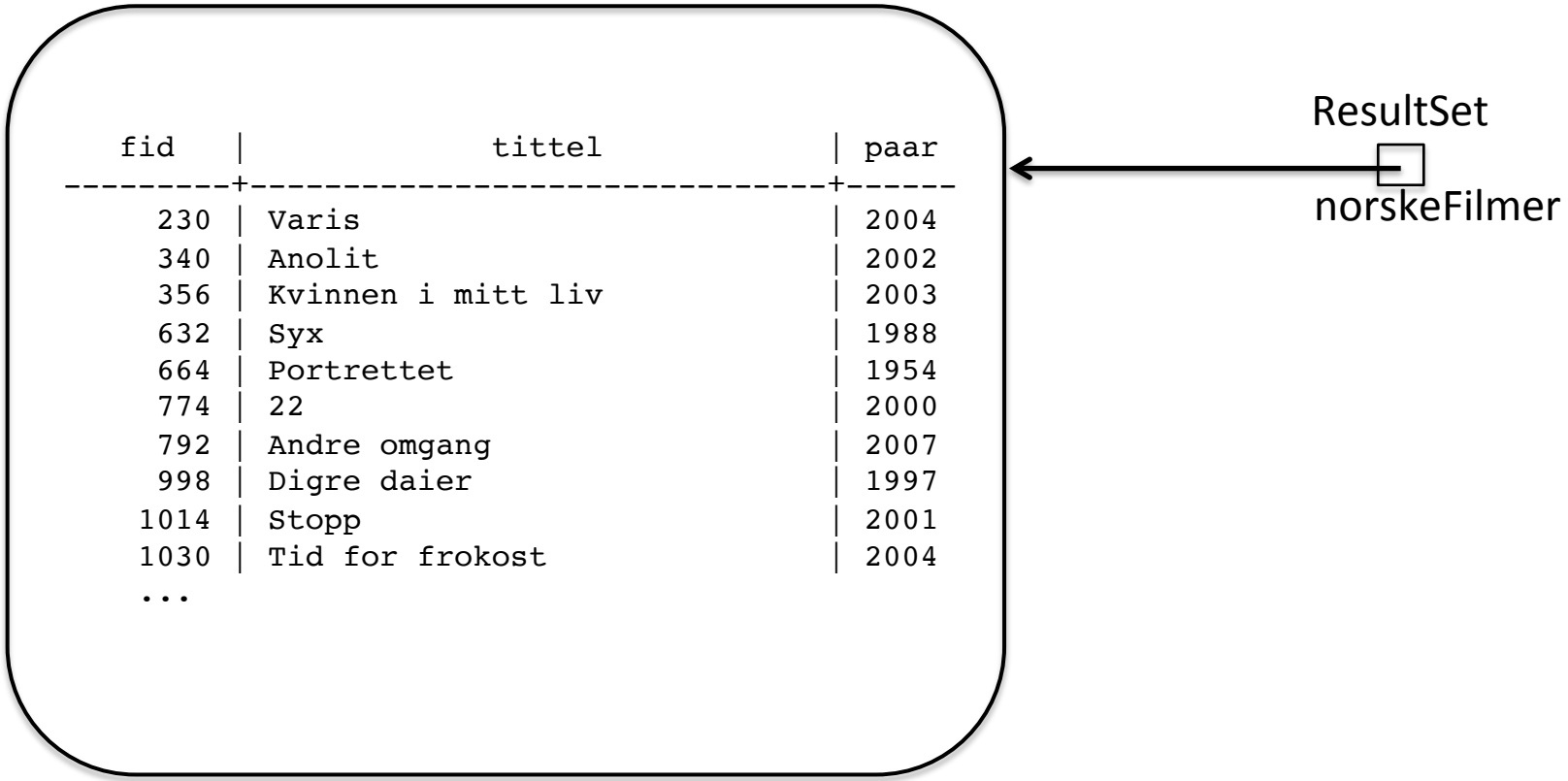
Utføre spørringer: Statement-objektet




```
String spørring = "select F.filmid as fid, "
                  +"       F.title as tittel, "
                  +"       F.prodyear as paar "
                  +"from film F natural join Filmcountry C "
                  +"where C.country='Norway' " ;


ResultSet norskeFilmer = stm.executeQuery (spørring);
```

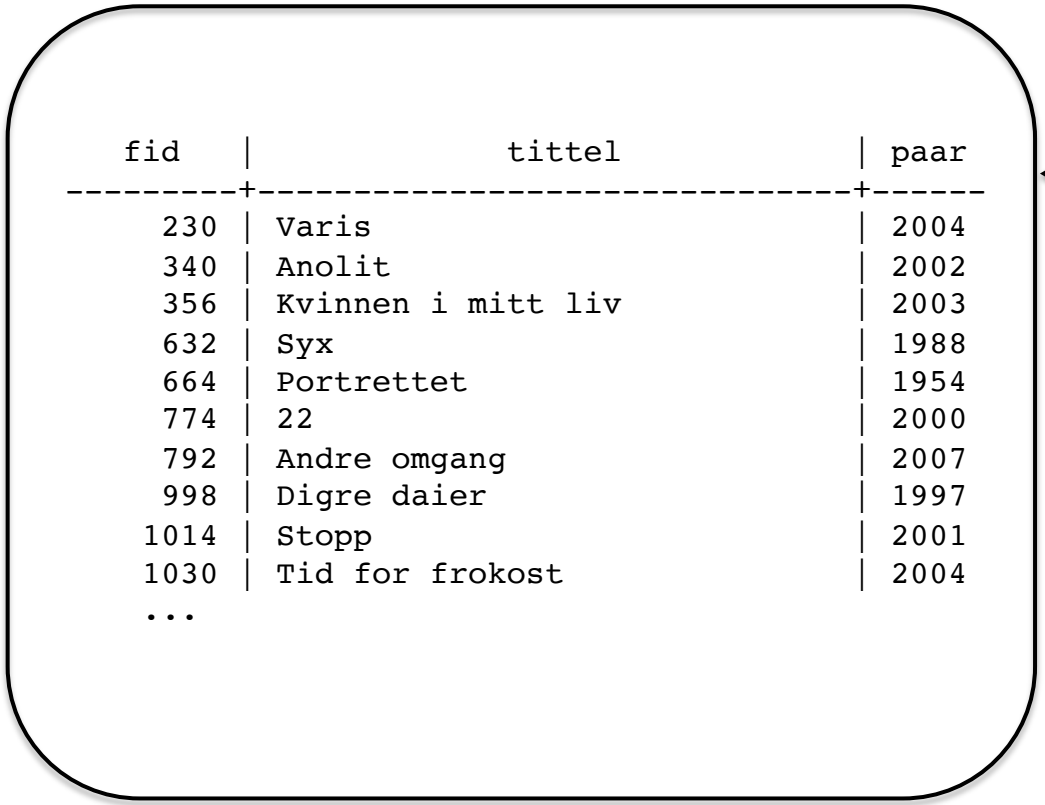

Connection
Statement
ResultSet

Hente resultatene: `ResultSet`-objektet




```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet

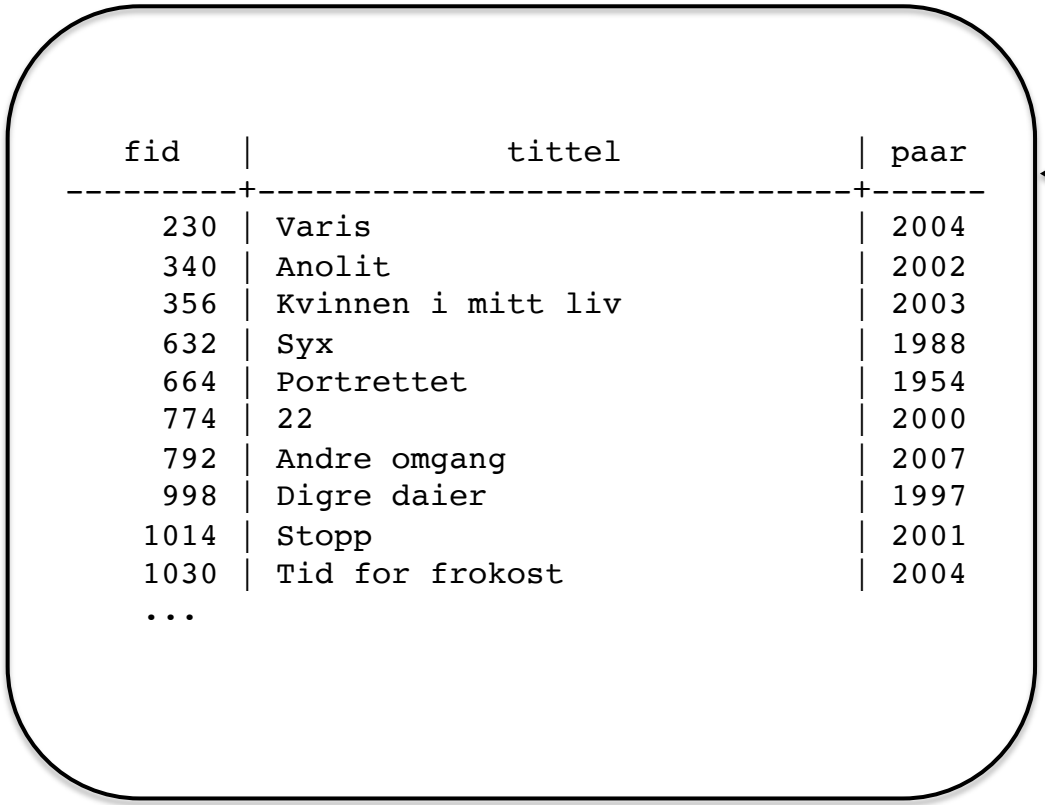


fid	tittel	paar
230	Varis	2004
340	Anolit	2002
356	Kvinnen i mitt liv	2003
632	Syx	1988
664	Portrettet	1954
774	22	2000
792	Andre omgang	2007
998	Digre daier	1997
1014	Stopp	2001
1030	Tid for frokost	2004
...		

A table of data representing a database result set, which is usually generated by executing a statement that queries the database.

```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet

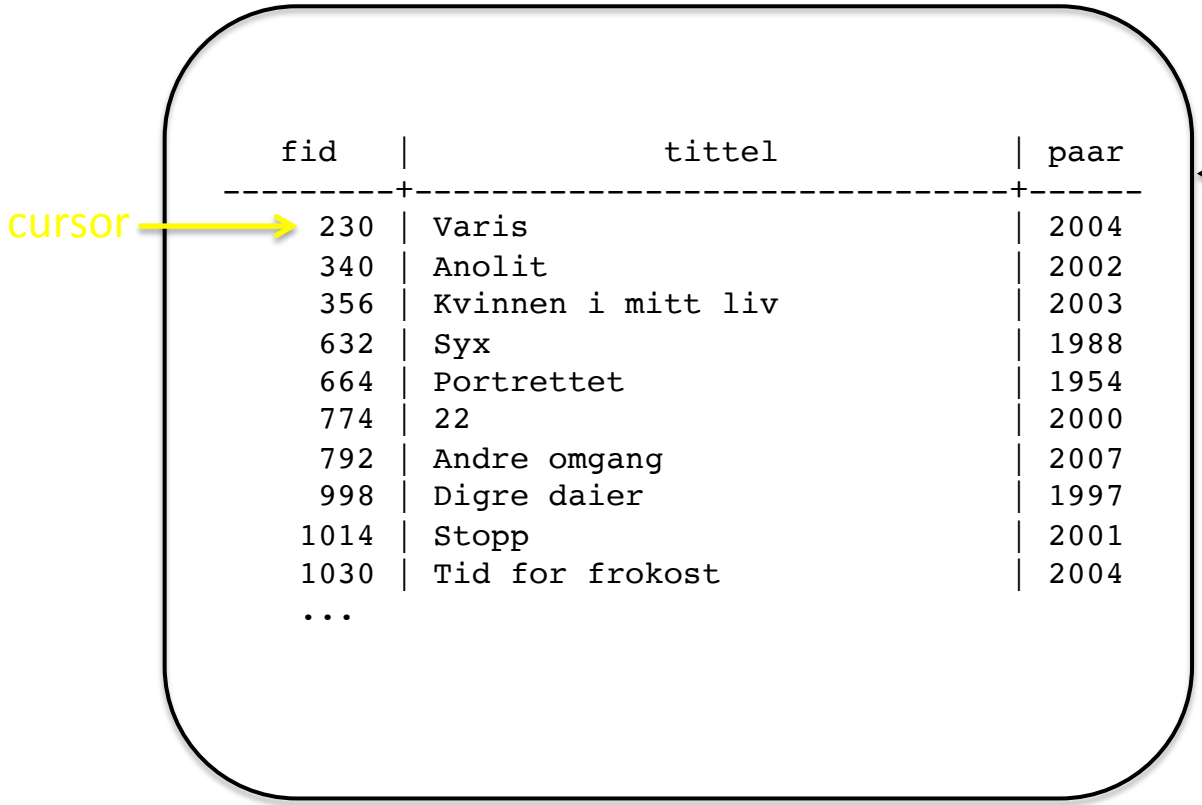


fid	tittel	paar
230	Varis	2004
340	Anolit	2002
356	Kvinnen i mitt liv	2003
632	Syx	1988
664	Portrettet	1954
774	22	2000
792	Andre omgang	2007
998	Digre daier	1997
1014	Stopp	2001
1030	Tid for frokost	2004
...		

A ResultSet object maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The next method moves the cursor to the next row, and because it returns false when there are no more rows in the ResultSet object, it can be used in a while loop to iterate through the result set..

```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

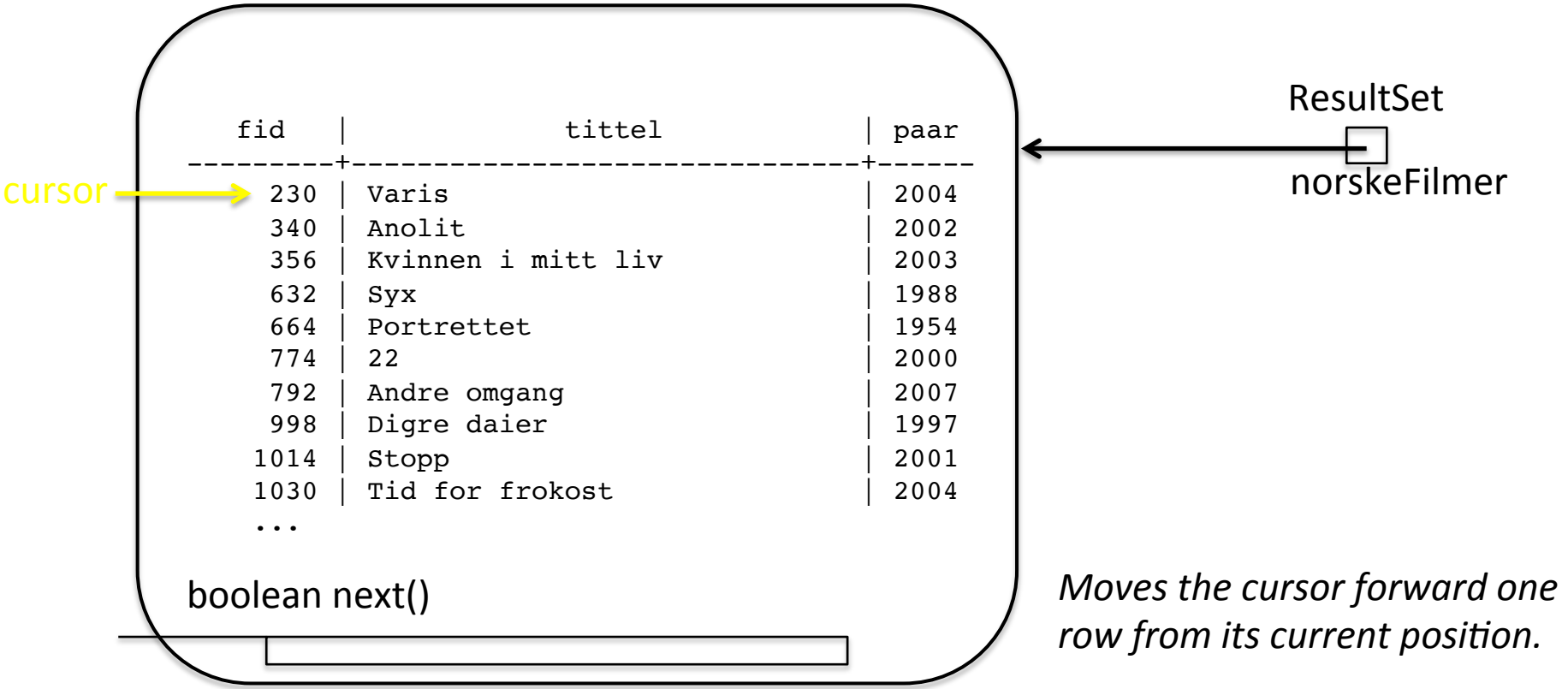
Hente resultatene: `ResultSet`-objektet



A ResultSet object maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The next method moves the cursor to the next row, and because it returns false when there are no more rows in the ResultSet object, it can be used in a while loop to iterate through the result set..

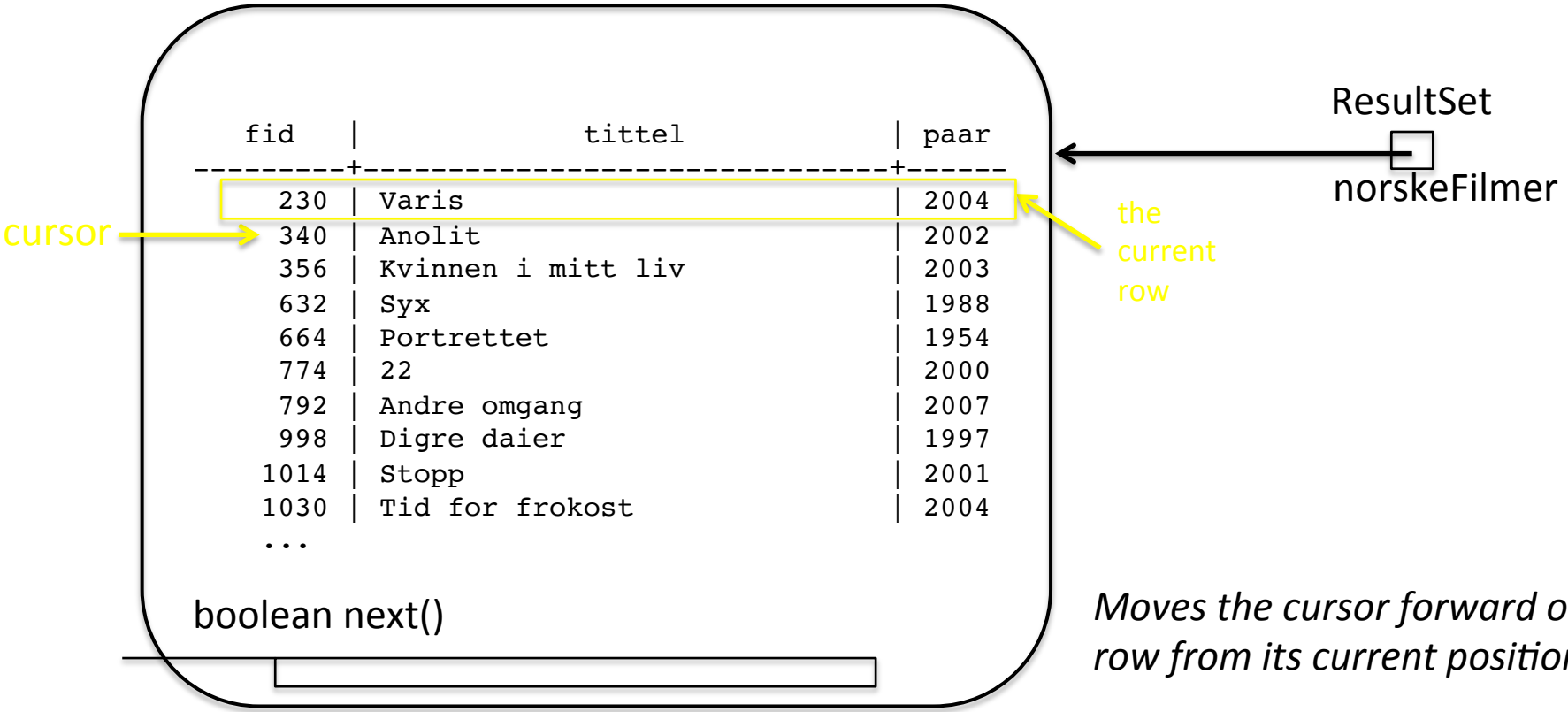
```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet




```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

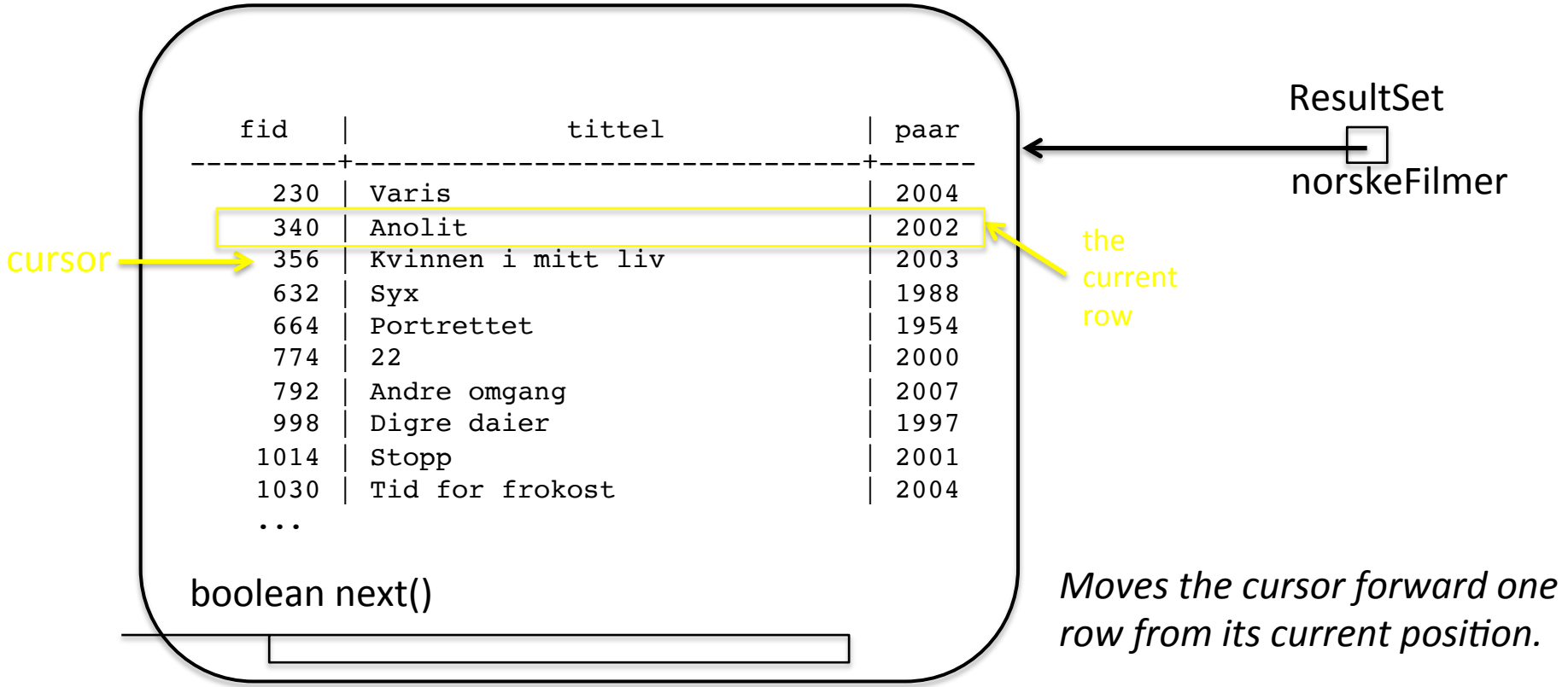
Hente resultatene: `ResultSet`-objektet



Moves the cursor forward one row from its current position.

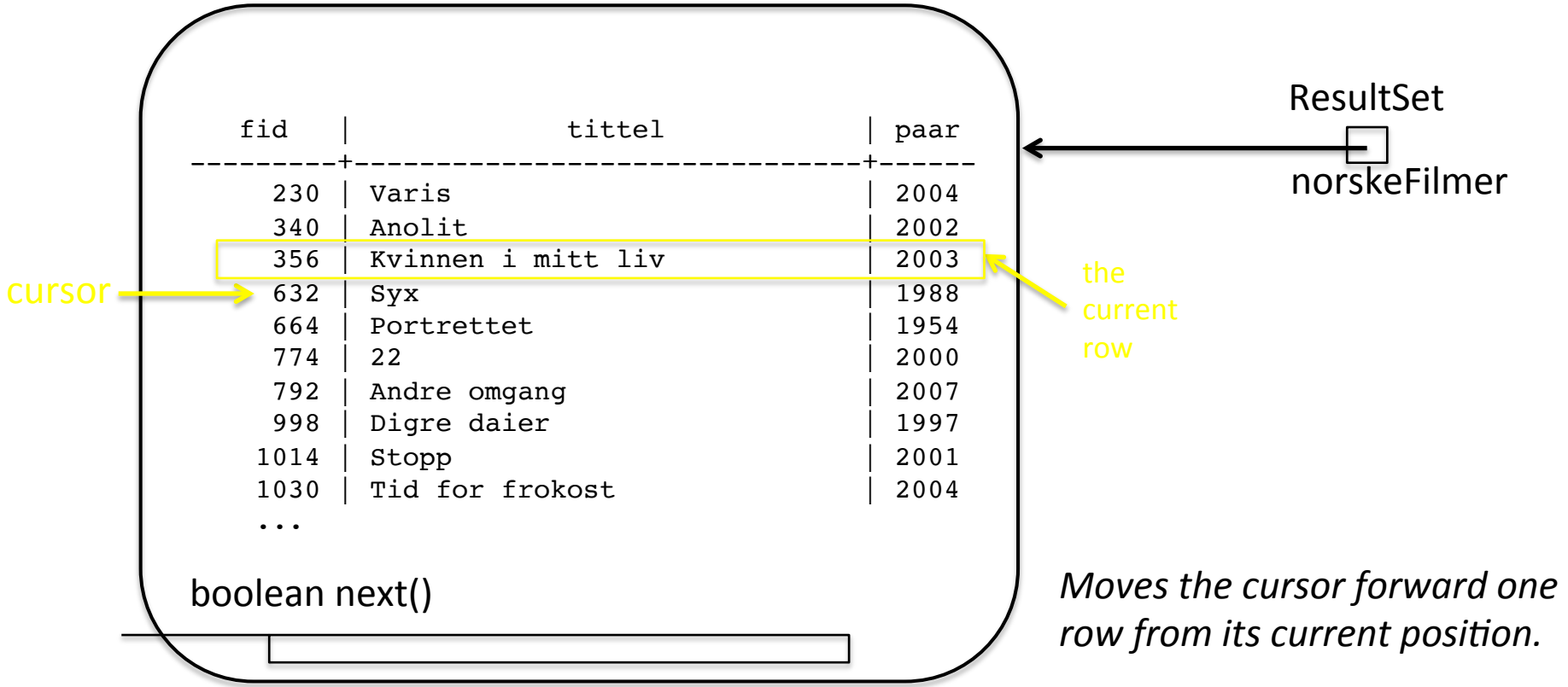
```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet



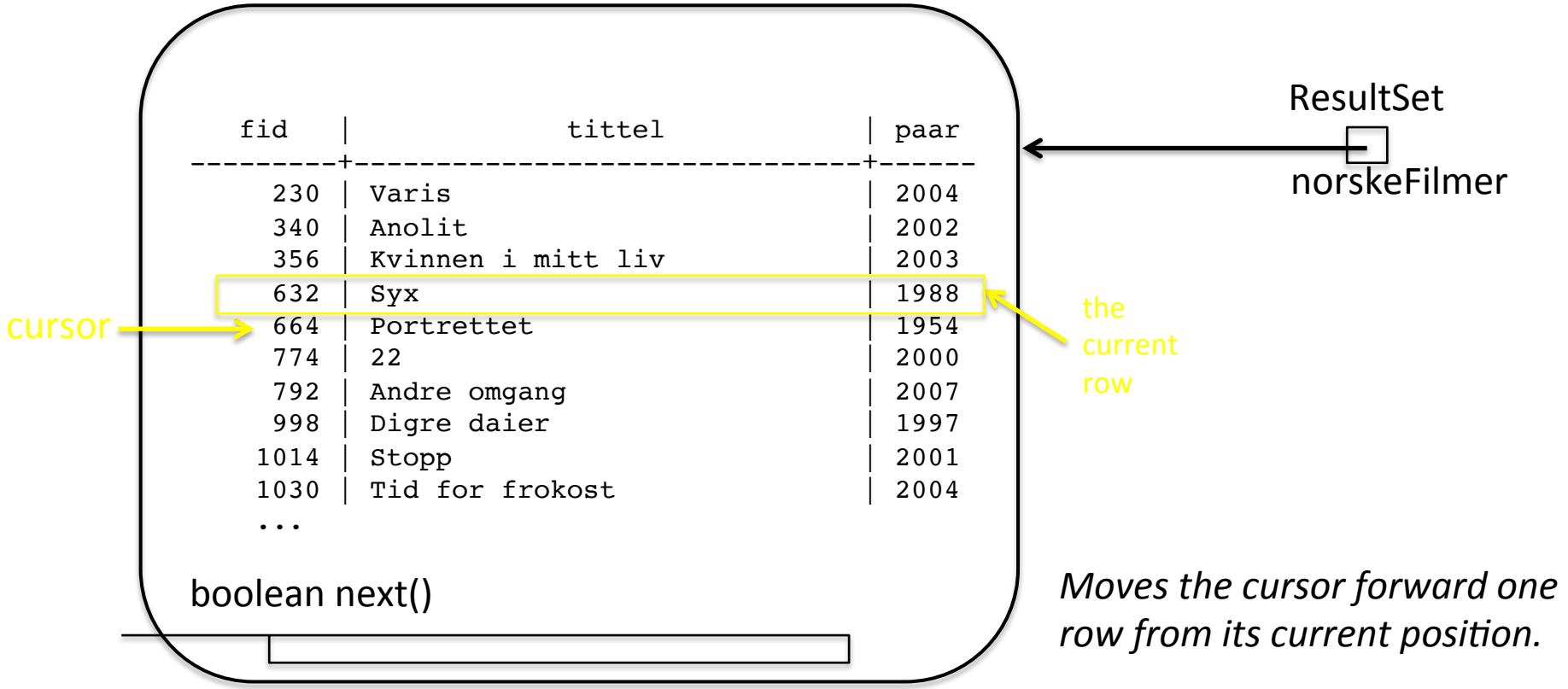

```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet



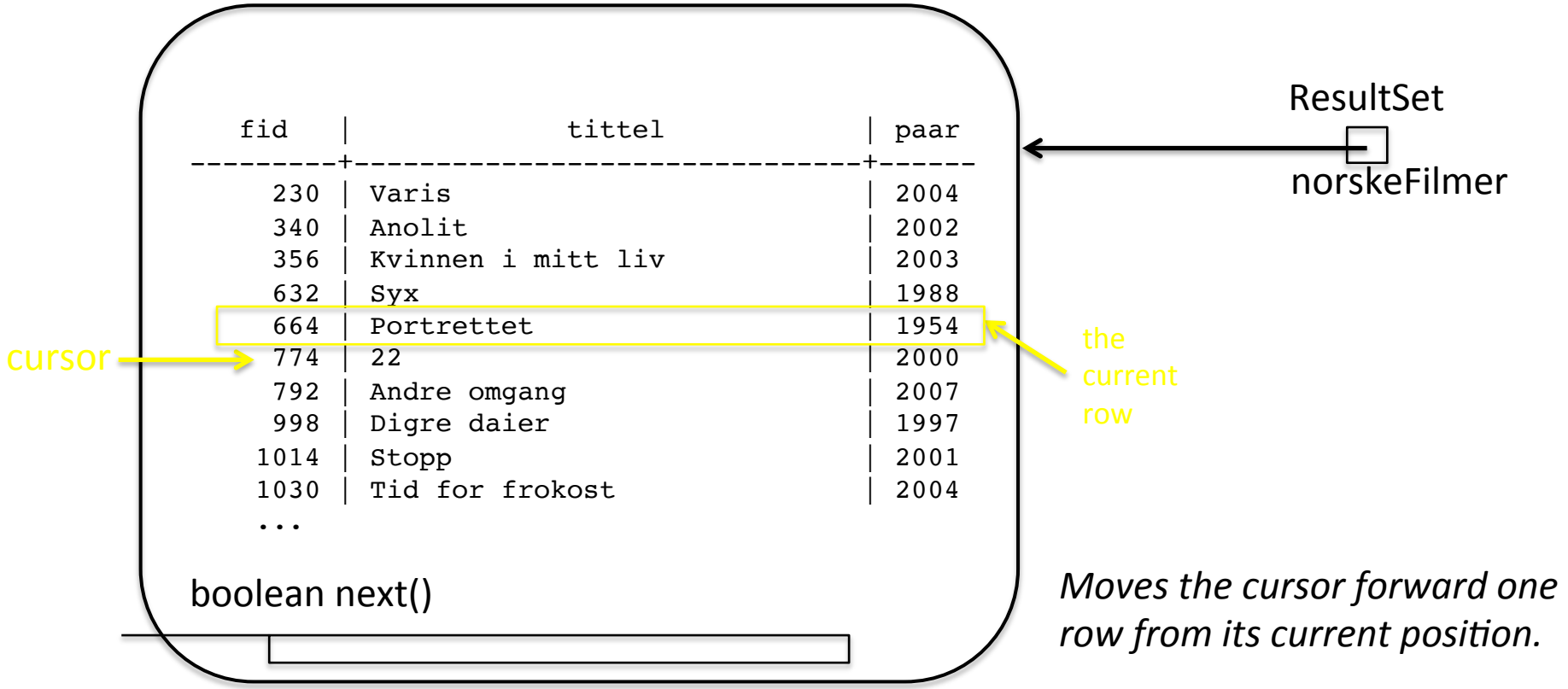

```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet



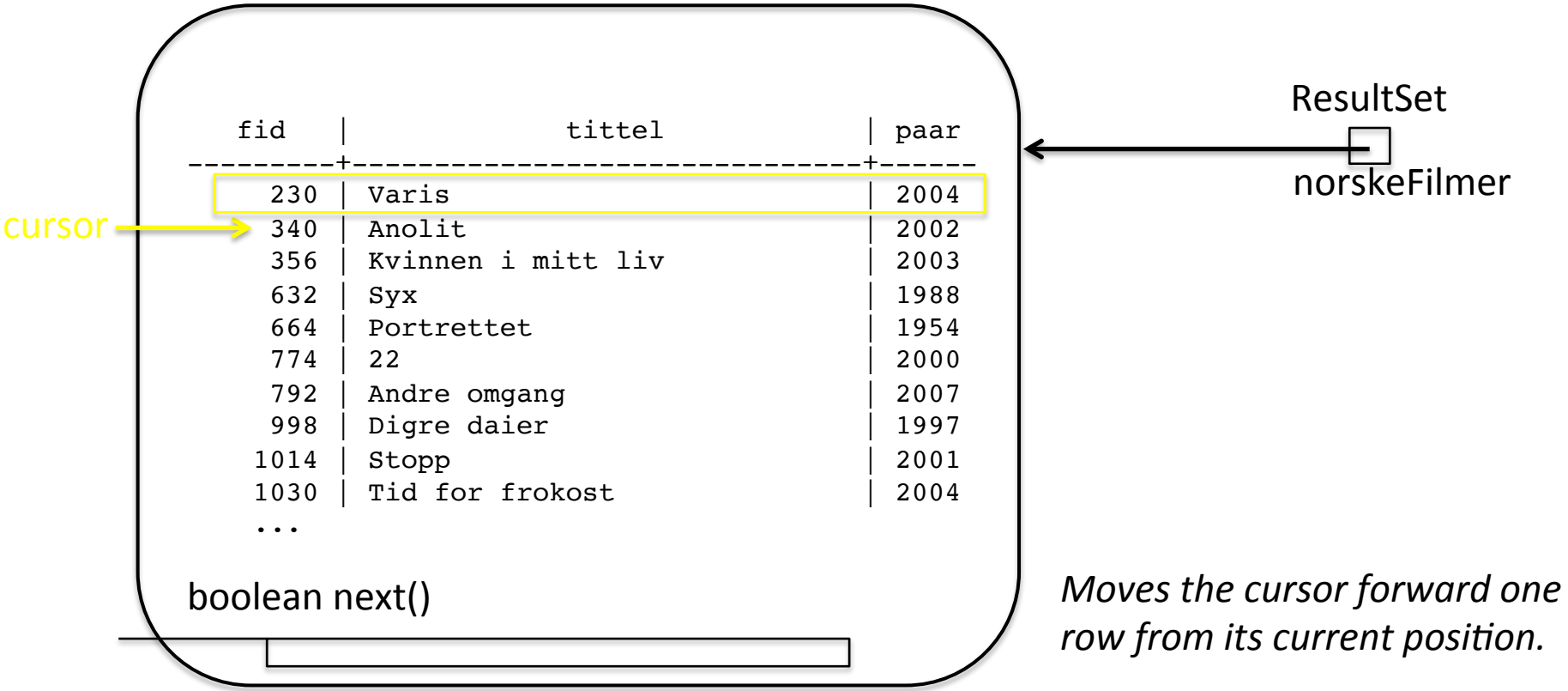

```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet



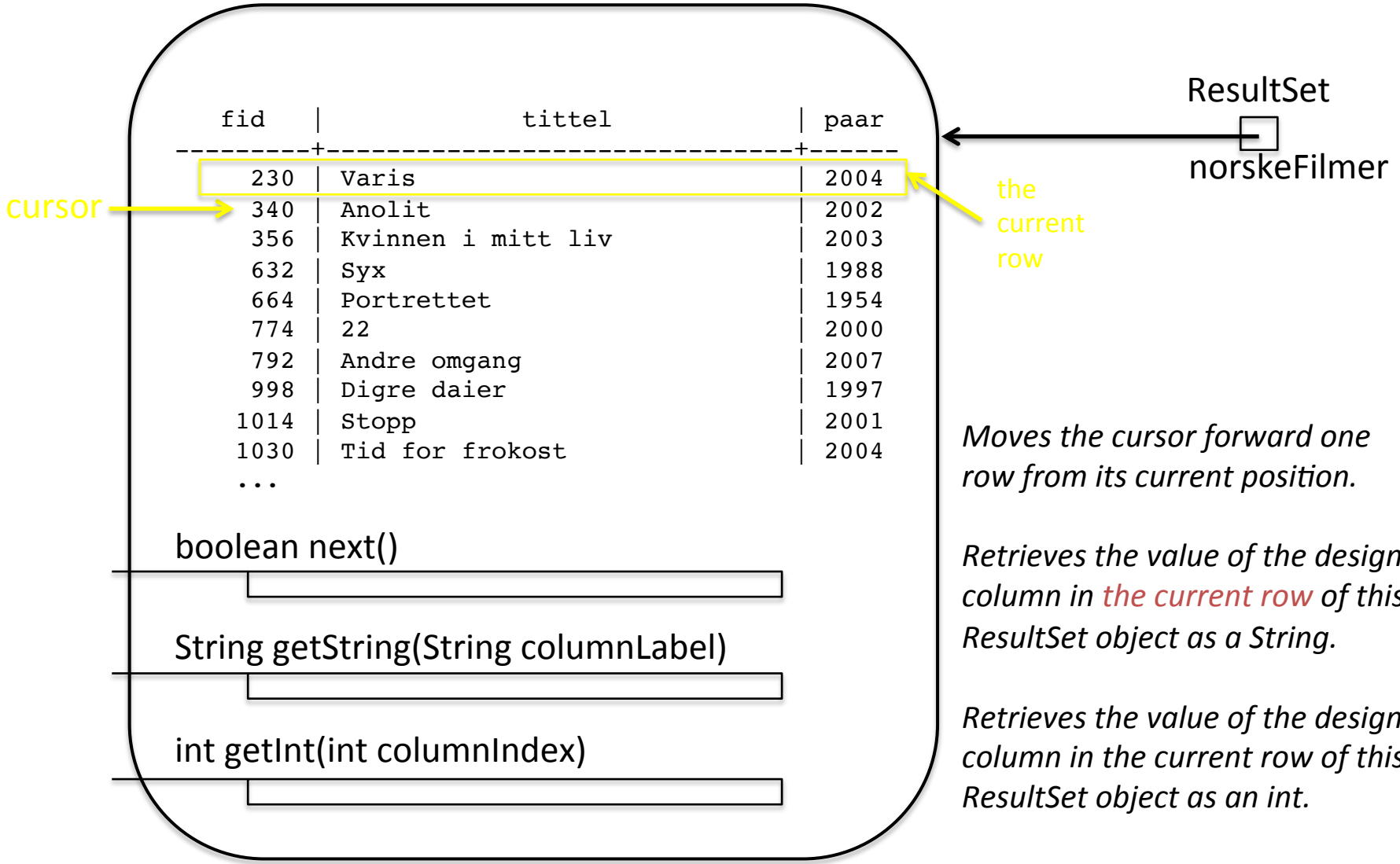

```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

Hente resultatene: `ResultSet`-objektet




```
ResultSet norskeFilmer = stm.executeQuery (spørring);
```

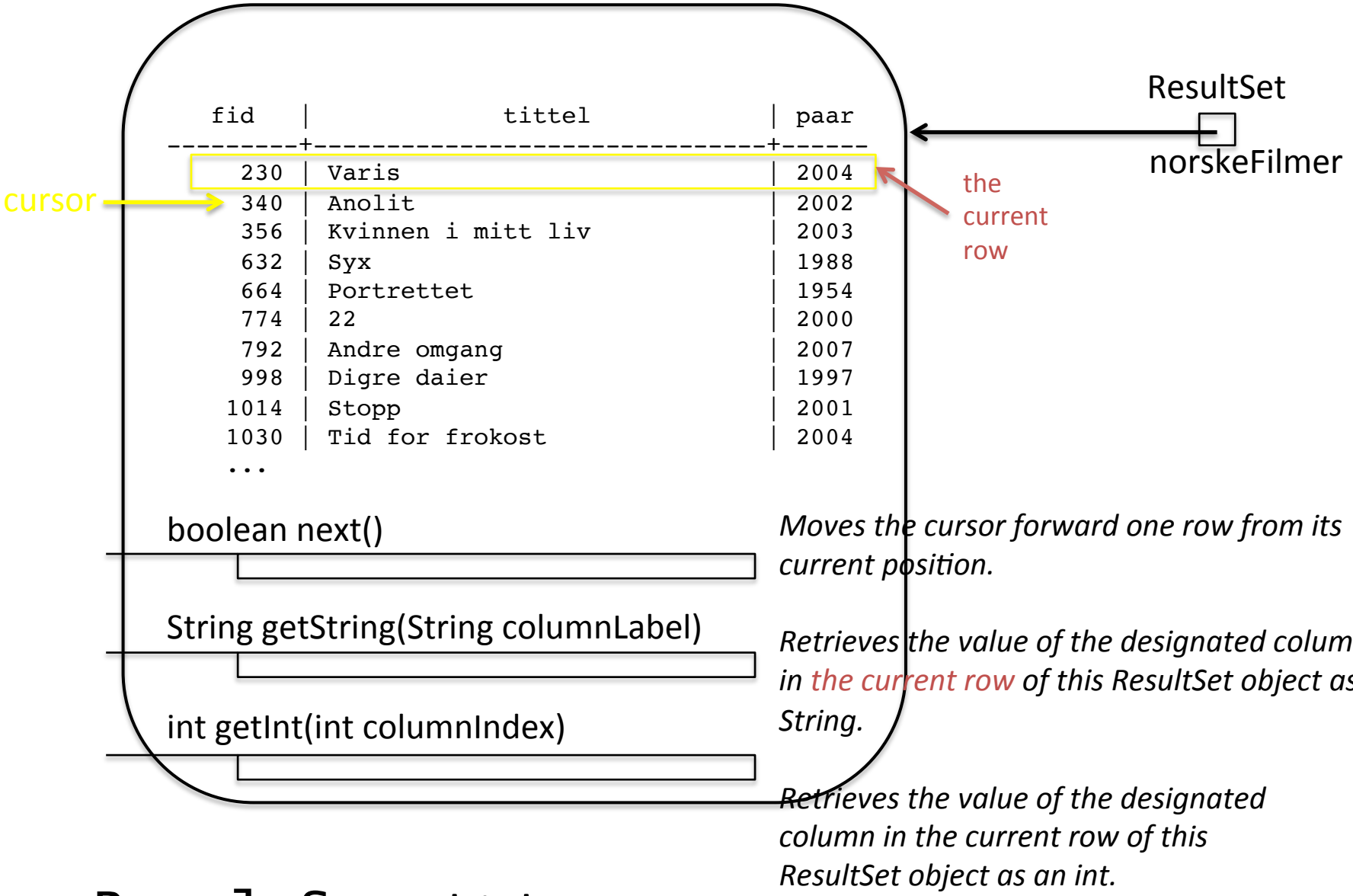
Hente resultatene: `ResultSet`-objektet



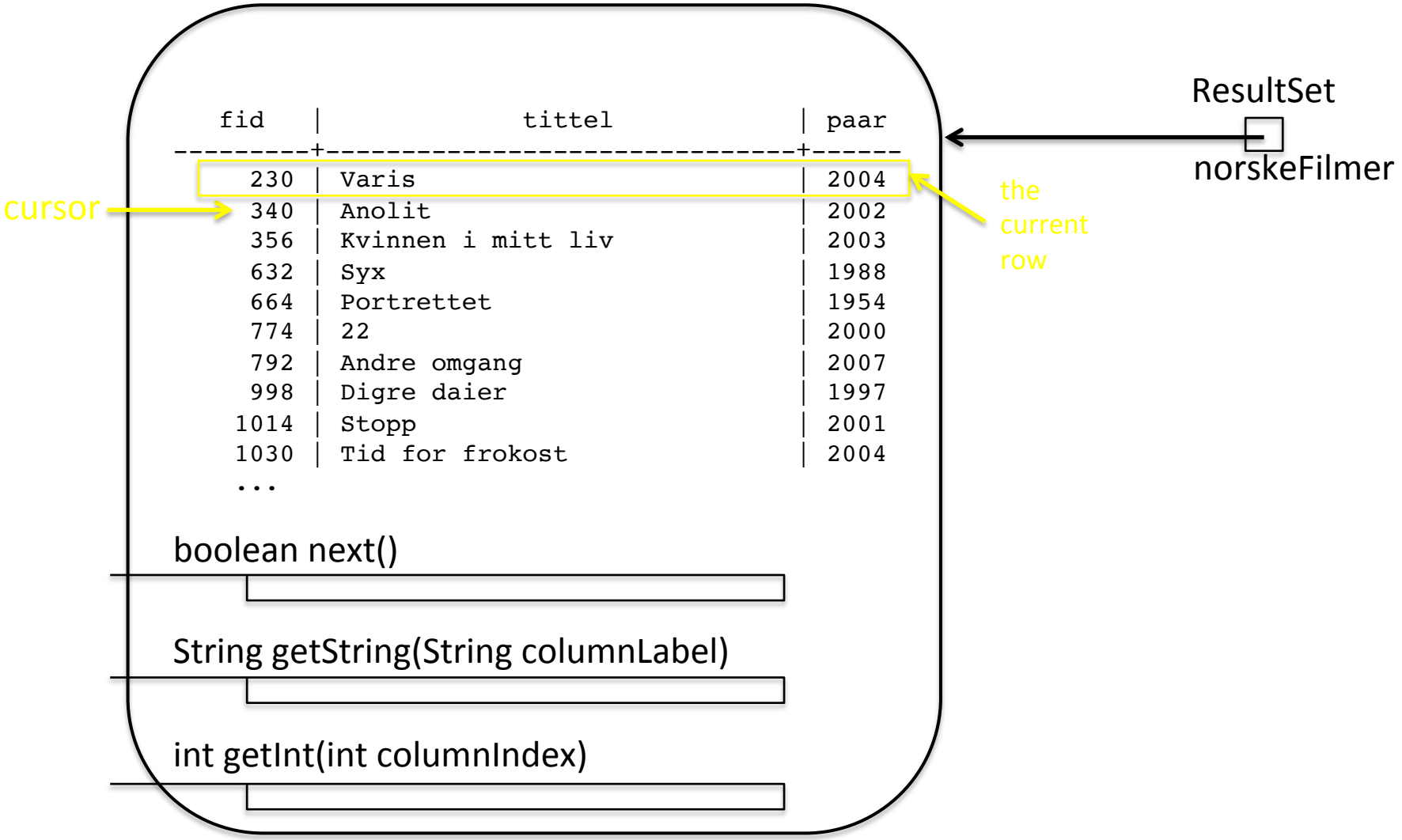
Moves the cursor forward one row from its current position.

Retrieves the value of the designated column in the current row of this ResultSet object as a String.

Retrieves the value of the designated column in the current row of this ResultSet object as an int.

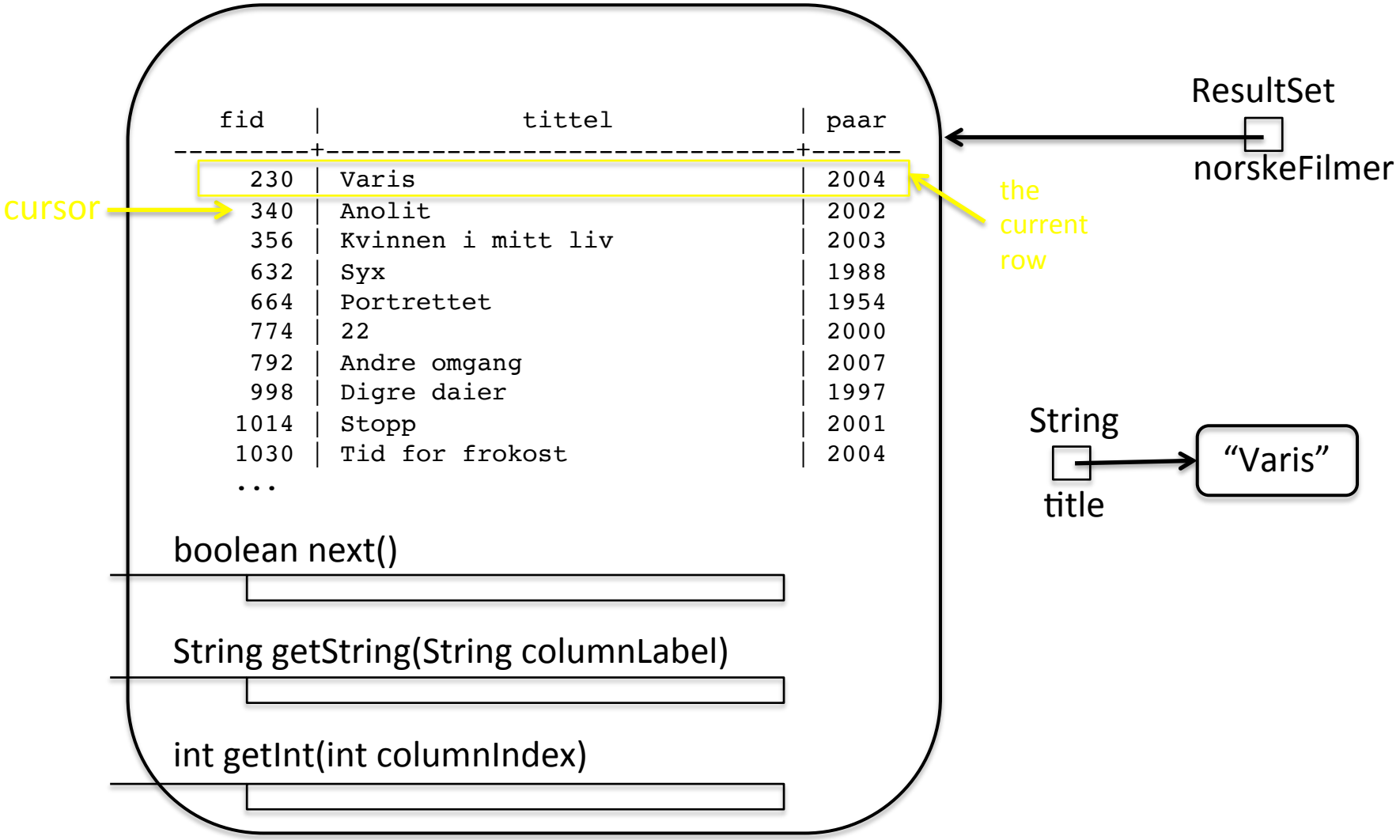


Et `ResultSet`-objekt



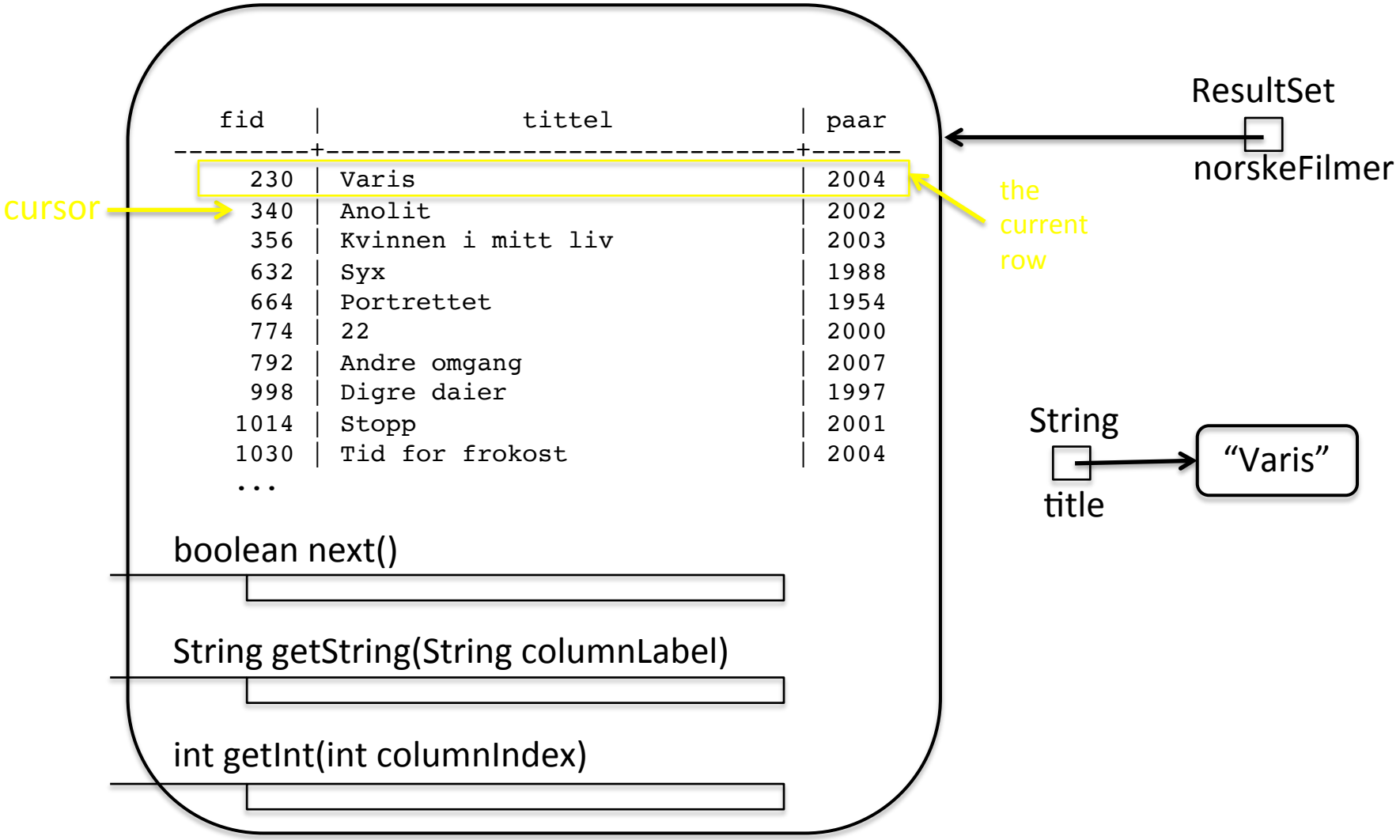
Et `ResultSet`-objekt

```
String title = norskeFilmer.getString(2);
```



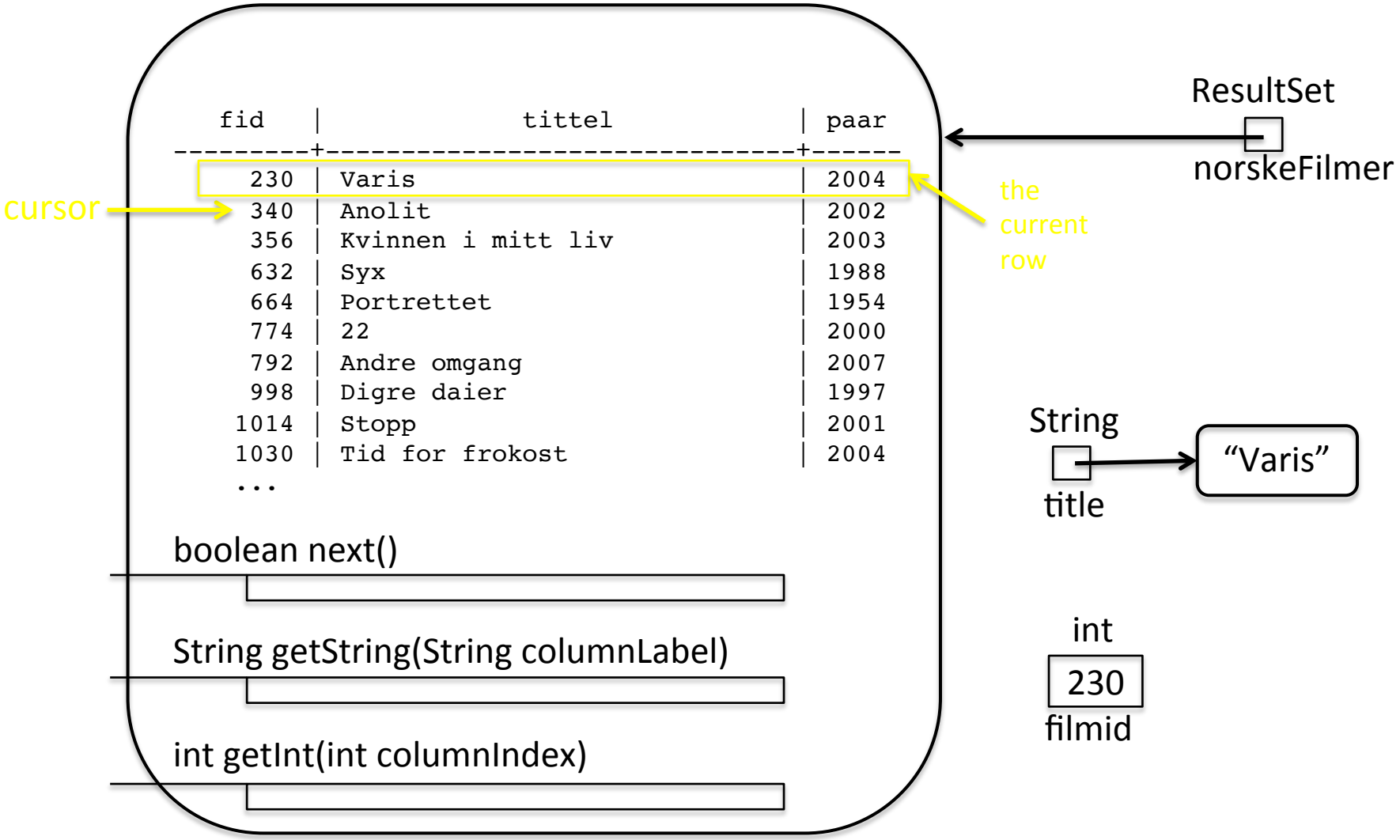
Et `ResultSet`-objekt

```
String title = norskeFilmer.getString(2);
```



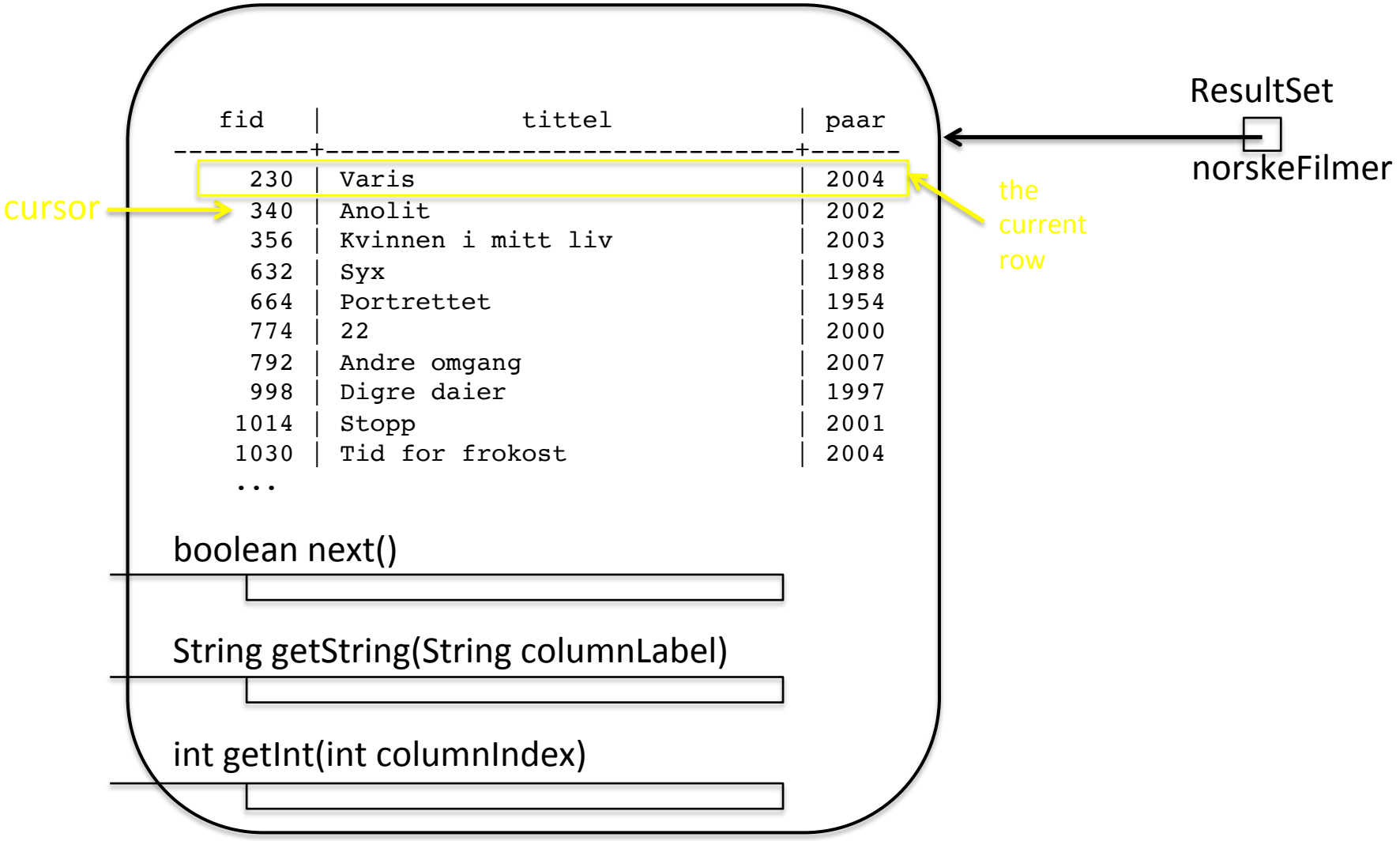
Et `ResultSet`-objekt

```
String title = norskeFilmer.getString(2);
int filmid = norskeFilmer.getInt("fid");
```

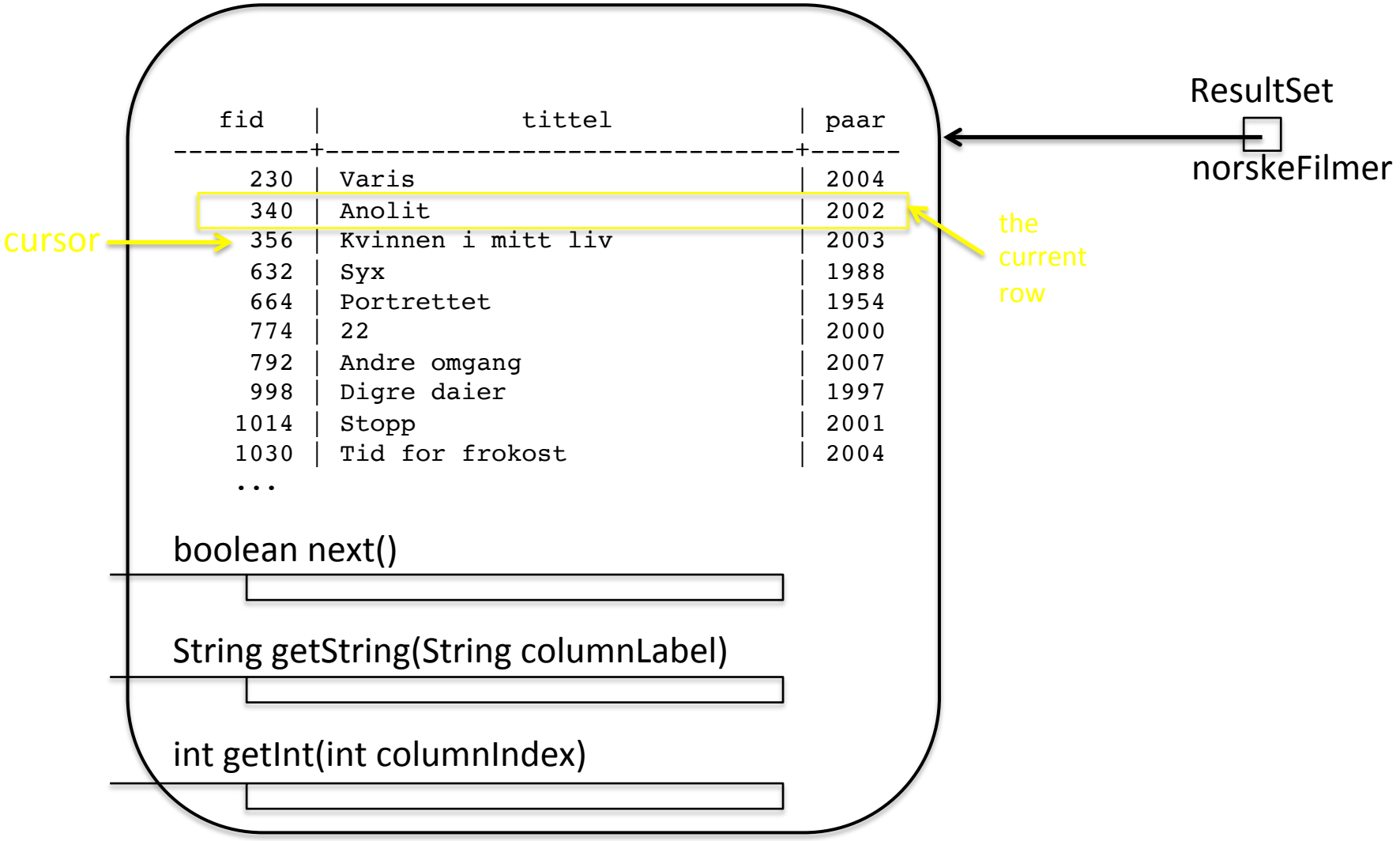


Et `ResultSet`-objekt

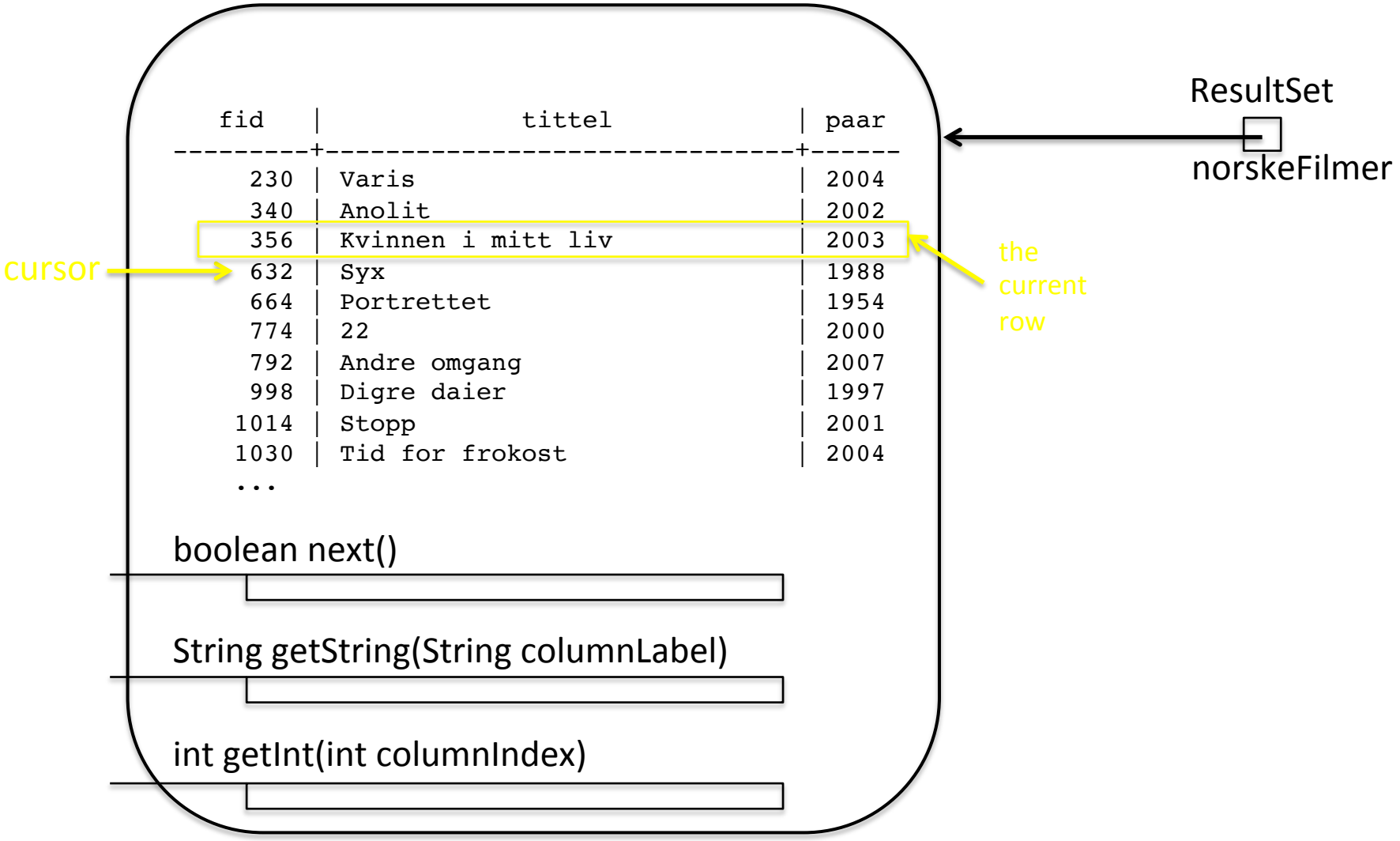
```
String title = norskeFilmer.getString(2);
int filmid = norskeFilmer.getInt("fid");
```



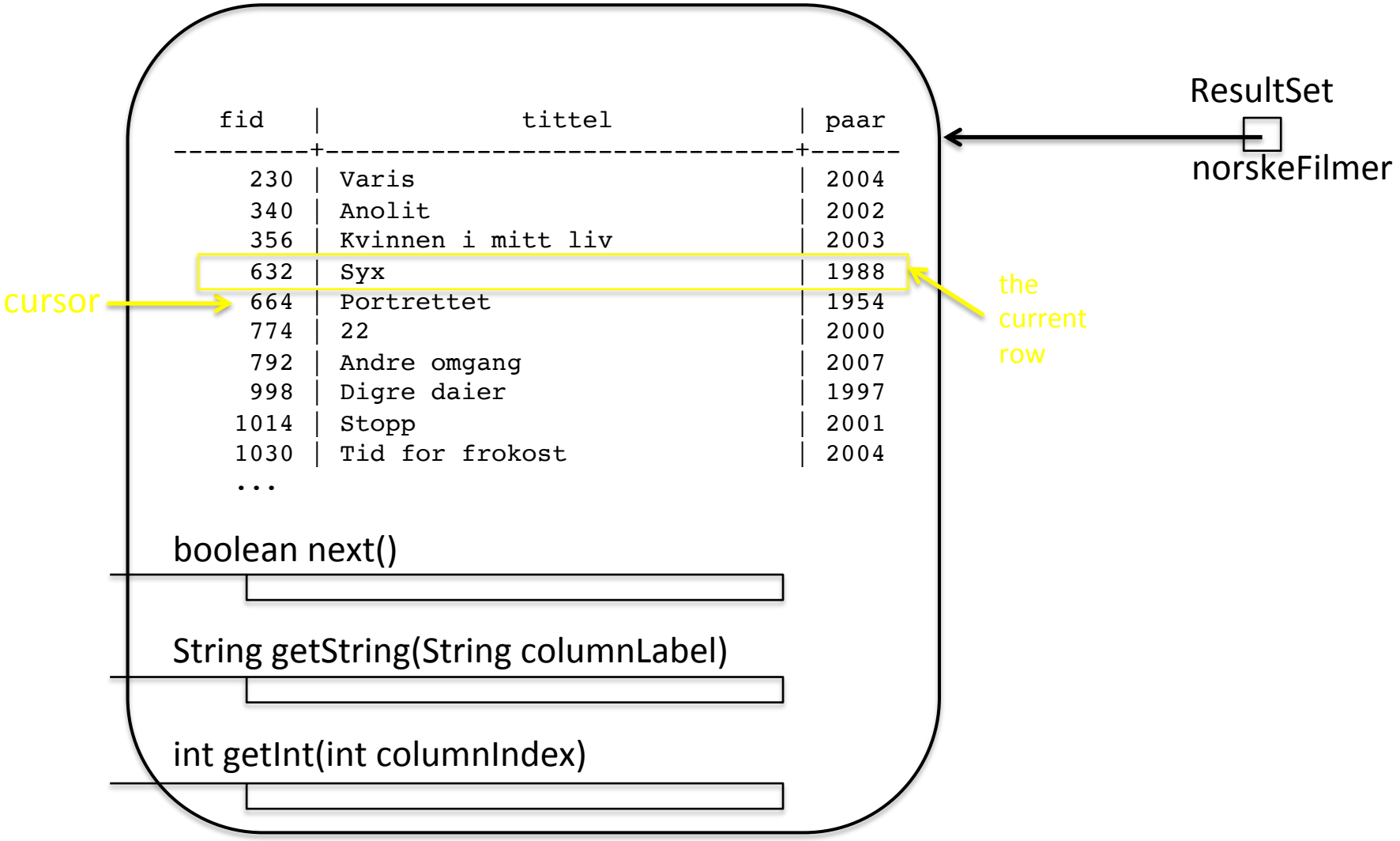

```
while (  norskeFilmer.next() ) {
                fm = new Film(norskeFilmer.getInt("fid"));
                fm.tittel = norskeFilmer.getString("tittel");
                fm.produksjonsår = norskeFilmer.getInt("paar");
                filmer.add(fm);
}
```




```
while (  norskeFilmer.next() ) {
                fm = new Film(norskeFilmer.getInt("fid"));
                fm.tittel = norskeFilmer.getString("tittel");
                fm.produksjonsår = norskeFilmer.getInt("paar");
                filmer.add(fm);
}
```



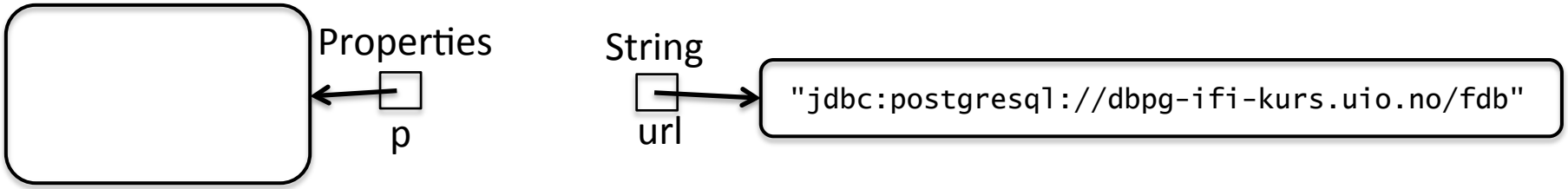

```
while (  norskeFilmer.next() ) {
                fm = new Film(norskeFilmer.getInt("fid"));
                fm.tittel = norskeFilmer.getString("tittel");
                fm.produksjonsår = norskeFilmer.getInt("paar");
                filmer.add(fm);
}
```

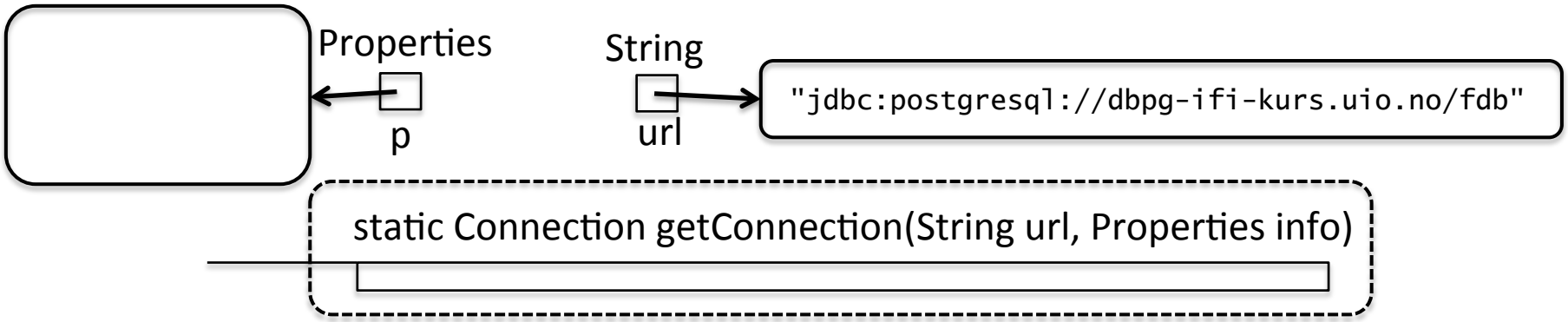



```
while (  norskeFilmer.next() ) {
                fm = new Film(norskeFilmer.getInt(1));
                fm.tittel = norskeFilmer.getString("tittel");
                fm.produksjonsår = norskeFilmer.getInt(3);
                filmer.add(fm);
}
```

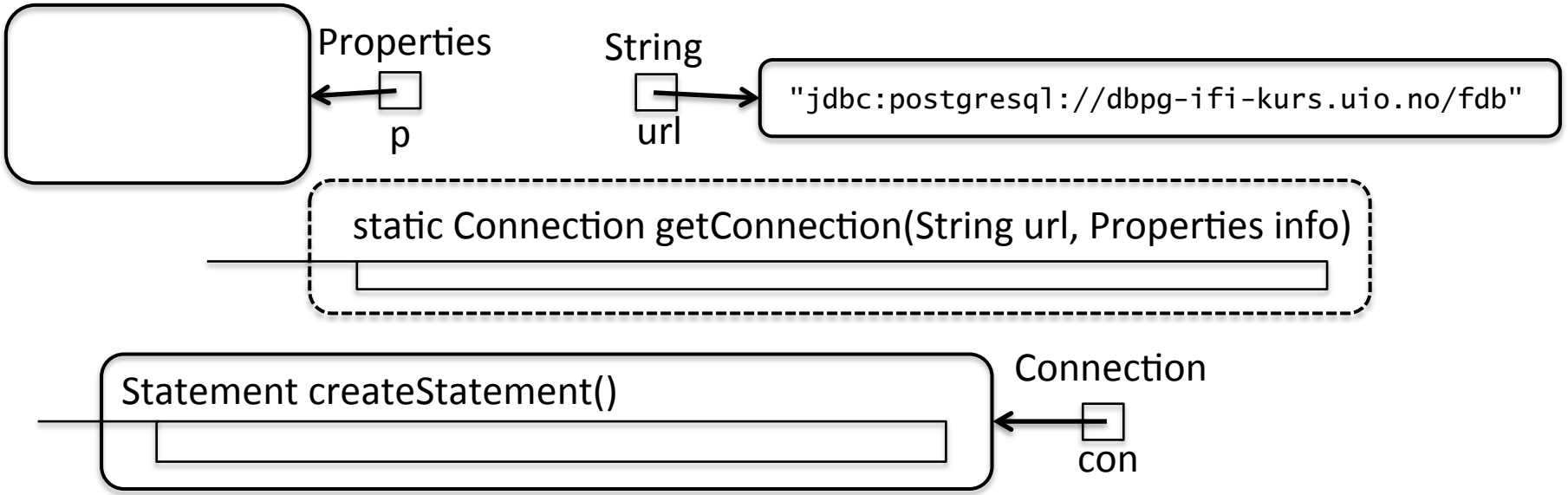
```
Properties p = new Properties();
p.setProperty("ssl", "true");
p.setProperty("sslfactory", "org.postgresql.ssl.NonValidatingFactory");
p.put("user", "michael");
p.put("password", passord); // "ifikurs_ro");
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

Connection con = DriverManager.getConnection(url, p); }

Statement stm = con.createStatement();

String spørring = "select F.filmid as fid, "
                    +"       F.title as tittel, "
                    +"       F.prodyear as paar "
                    +"from film F natural join Filmcountry C "
                   +"where C.country='Norway'" ;

ResultSet norskeFilmer = stm.executeQuery (spørring);

while (norskeFilmer.next() ) {
        fm = new Film(norskeFilmer.getInt("fid"));
        fm.tittel = norskeFilmer.getString("tittel");
        fm.produksjonsår = norskeFilmer.getInt("paar");
        filmer.add(fm);
}
```

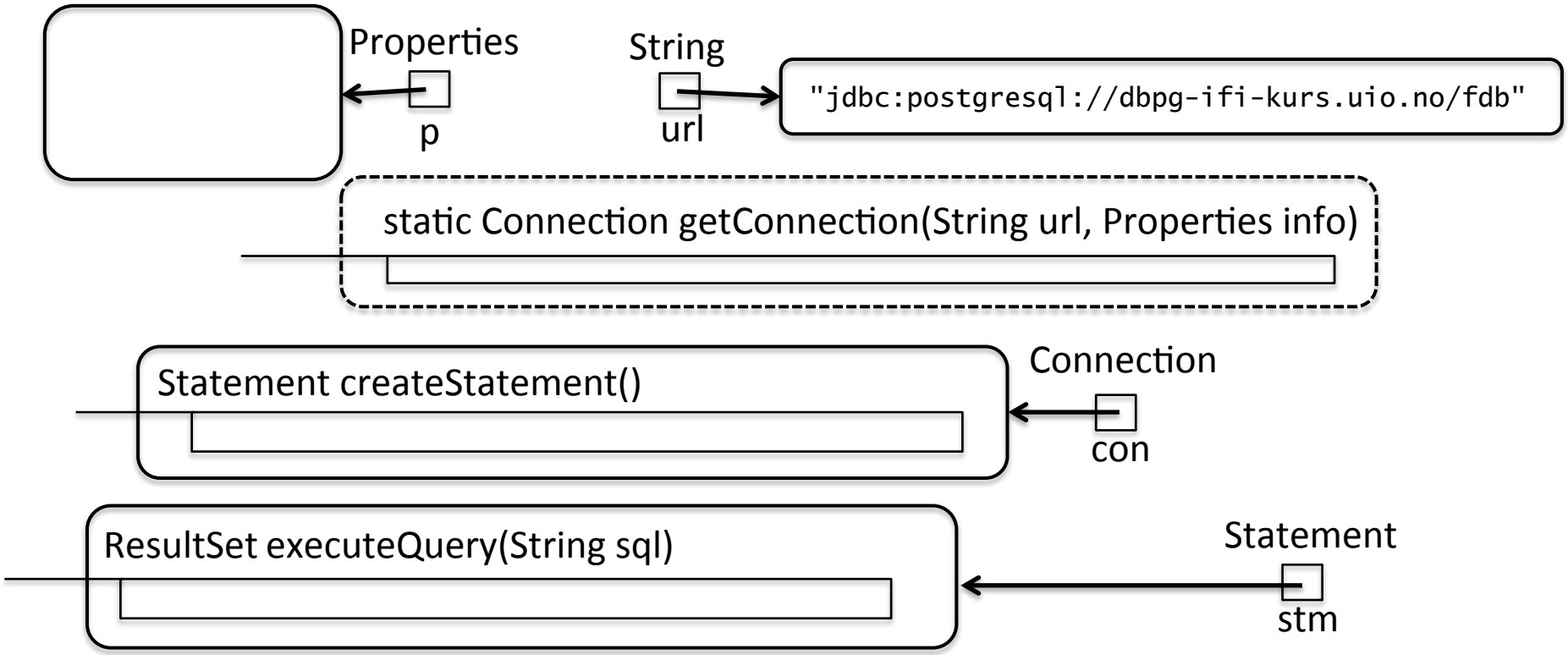
```
Properties p = new Properties();
p.setProperty("ssl", "true");
p.setProperty("sslfactory", "org.postgresql.ssl.NonValidatingFactory");
p.put("user", "michael");
p.put("password", passord); // "ifikurs_ro");
String url = "jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb";

String spørring = "select F.filmid as fid, "
                    +"       F.title as tittel, "
                    +"       F.prodyear as paar "
                    +"from film F natural join Filmcountry C "
                   +"where C.country='Norway'" ;


DriverManager.getConnection
                    (url,p).createStatement( ).executeQuery(spørring);


while (norskeFilmer.next() ) {
        fm = new Film(norskeFilmer.getInt("fid"));
        fm.tittel = norskeFilmer.getString("tittel");
        fm.produksjonsår = norskeFilmer.getInt("paar");
        filmer.add(fm);
}
```

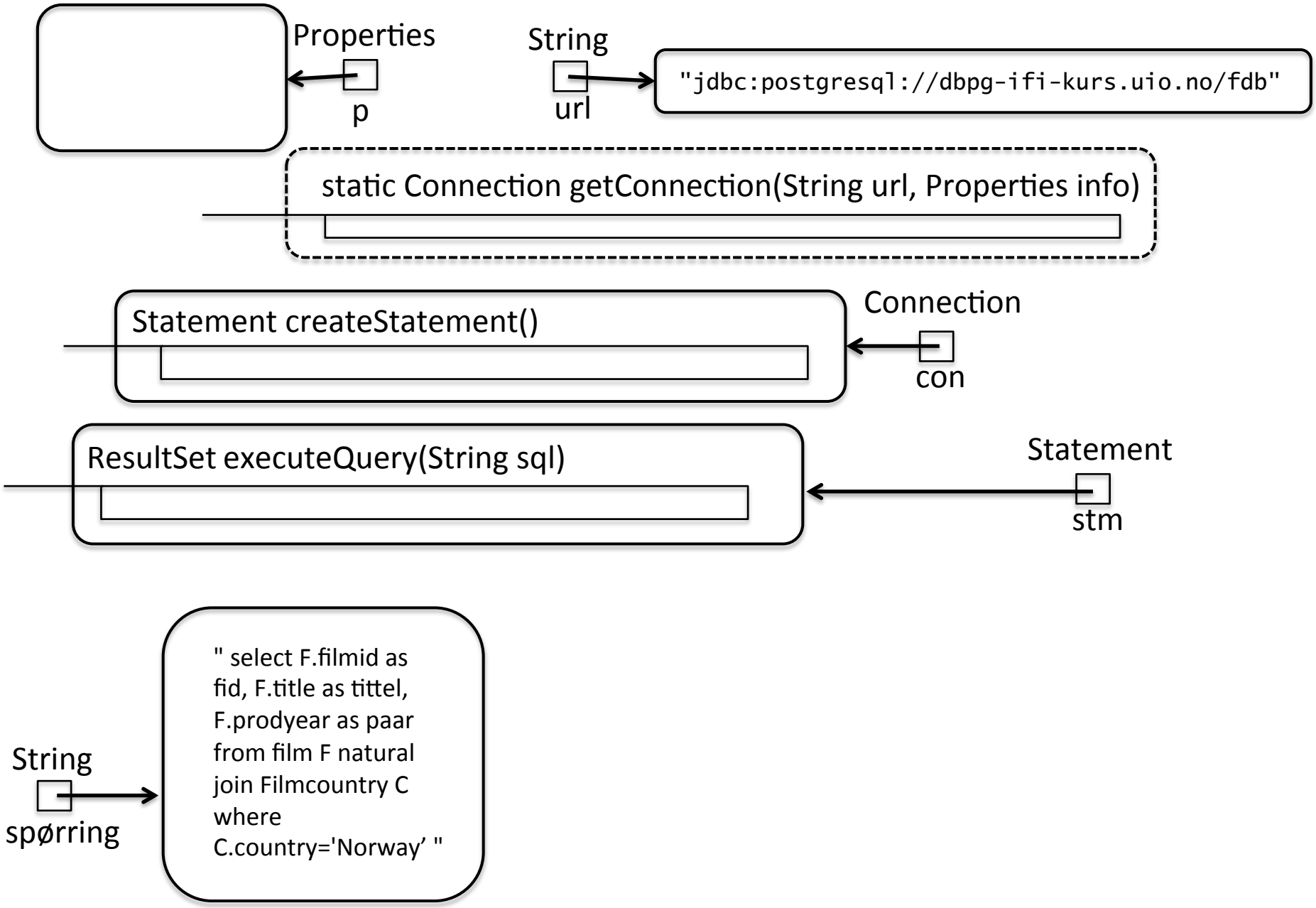

Properties
p
String
url
"jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb"

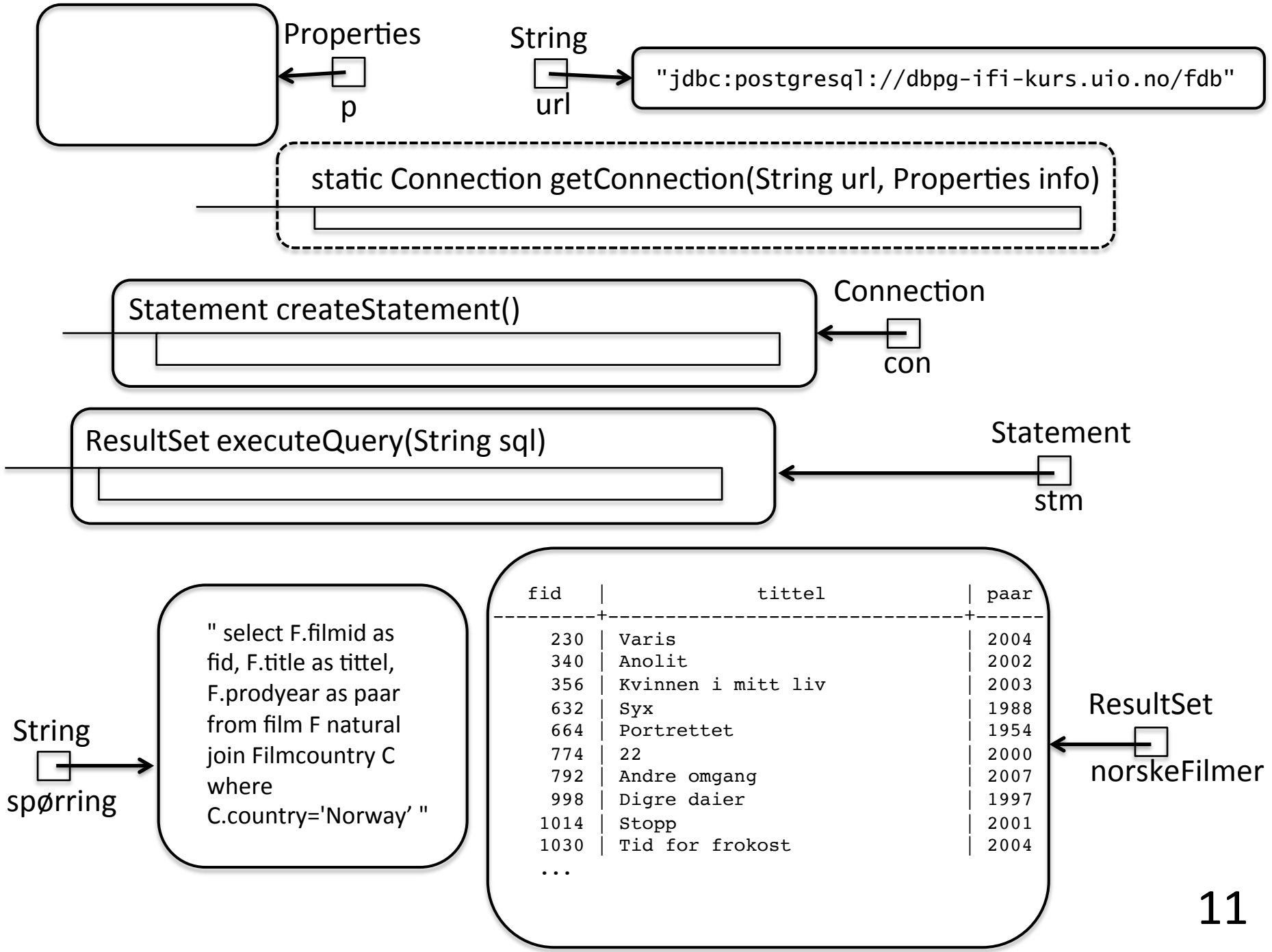


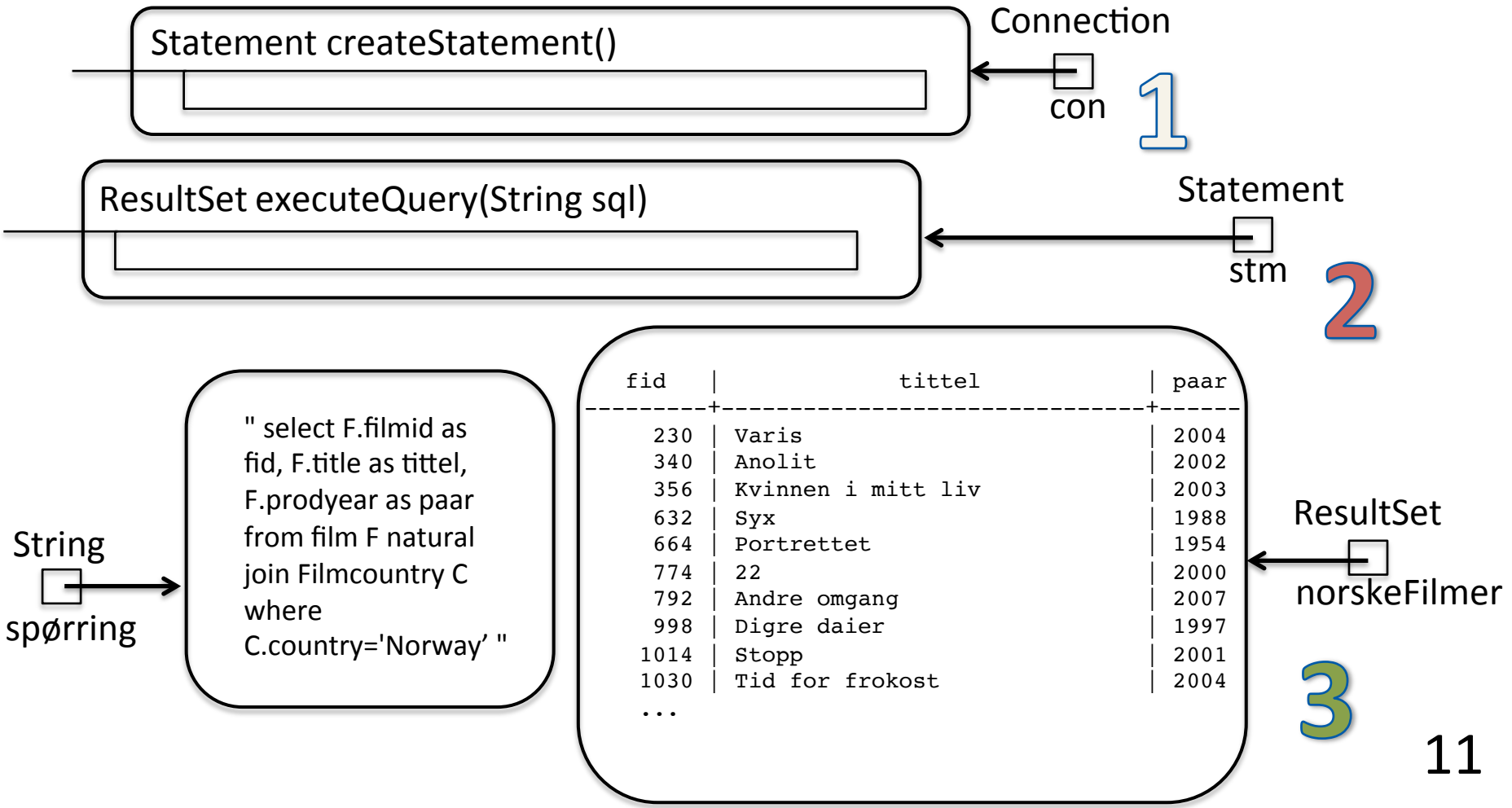
Klassedatastruktur for klassen DriverManager

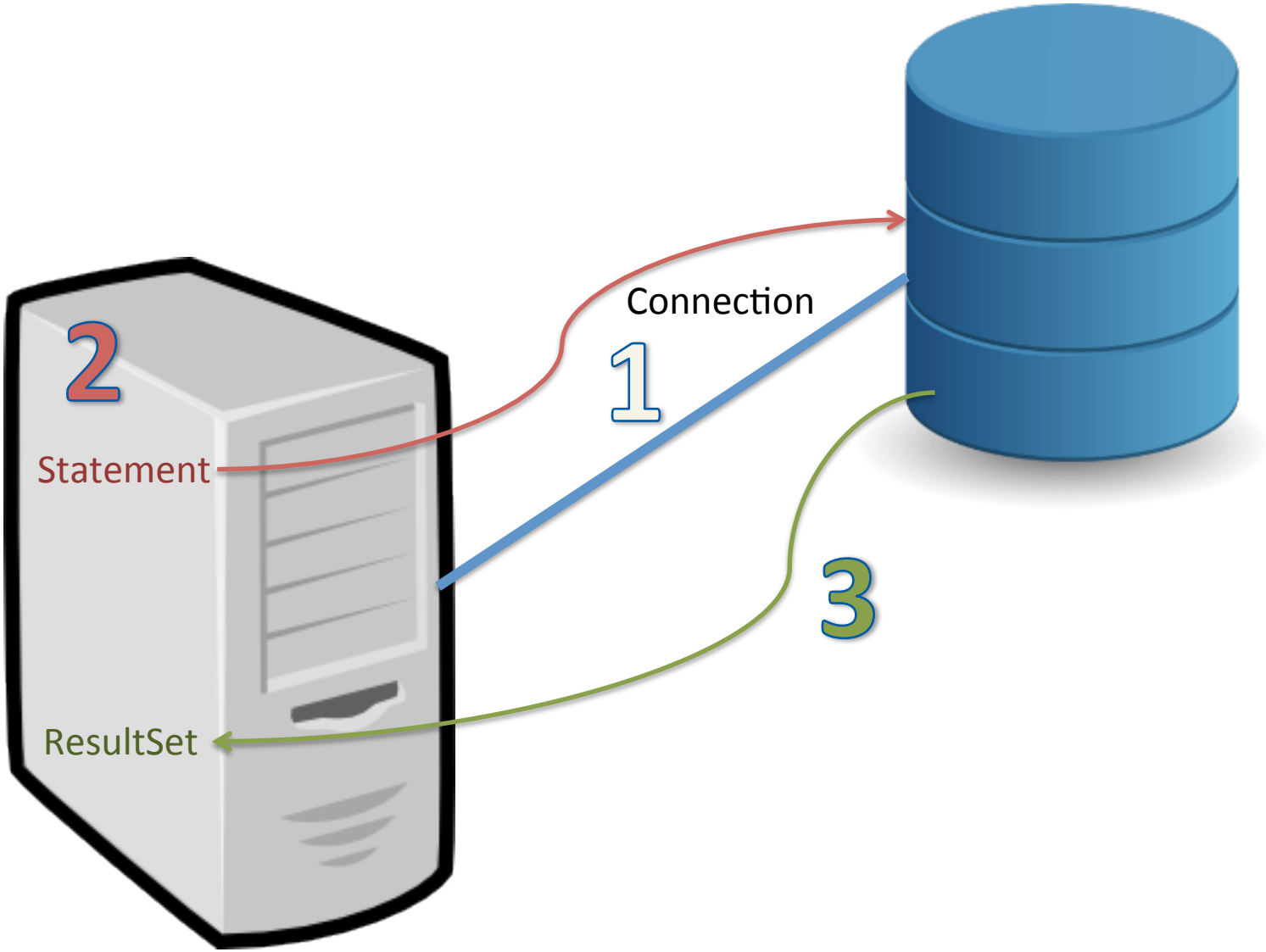

Properties
p
String
url
"jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb"
static Connection getConnection(String url, Properties info)
Statement createStatement()
Connection
con


Properties
p
String
url
"jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb"
static Connection getConnection(String url, Properties info)
Statement createStatement()
Connection
con
ResultSet executeQuery(String sql)
Statement
stm


Properties
p
String
url
"jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb"
static Connection getConnection(String url, Properties info)
Statement createStatement()
Connection
con
ResultSet executeQuery(String sql)
Statement
stm
String
spørring
" select F.filmid as fid, F.title as tittel, F.prodyear as paar from film F natural join Filmcountry C where C.country='Norway' "


Properties
p
String
url
"jdbc:postgresql://dbpg-ifi-kurs.uio.no/fdb"
static Connection getConnection(String url, Properties info)
Statement createStatement()
Connection
con
ResultSet executeQuery(String sql)
Statement
stm
String
spørring
" select F.filmid as fid, F.title as tittel, F.prodyear as paar from film F natural join Filmcountry C where C.country='Norway' "
fid | tittel | paar
230 | Varis | 2004
340 | Anolit | 2002
356 | Kvinnen i mitt liv | 2003
632 | Syx | 1988
664 | Portrettet | 1954
774 | 22 | 2000
792 | Andre omgang | 2007
998 | Digre daier | 1997
1014 | Stopp | 2001
1030 | Tid for frokost | 2004
...
ResultSet
norskeFilmer


Statement createStatement()
Connection
con
1
ResultSet executeQuery(String sql)
Statement
stm
2
String
spørring
" select F.filmid as fid, F.title as tittel, F.prodyear as paar from film F natural join Filmcountry C where C.country='Norway' "
fid | tittel | paar
230 | Varis | 2004
340 | Anolit | 2002
356 | Kvinnen i mitt liv | 2003
632 | Syx | 1988
664 | Portrettet | 1954
774 | 22 | 2000
792 | Andre omgang | 2007
998 | Digre daier | 1997
1014 | Stopp | 2001
1030 | Tid for frokost | 2004
...
ResultSet
norskeFilmer
3


2
Statement
Connection
1
3
ResultSet

Parametriserte spørringer

- Hva om vi ønsker å bruke strings eller annen brukerinput?
- Det er ikke lurt å konkatenere disse på «vanlig» måte!

Om vi f.eks. har en string som inneholder følgende:

```
String input = "O'boy"
```

Ser vi kanskje at dette kan føre til problemer i denne spørringen:

```
String query = "SELECT * FROM tabell WHERE name = '"+input+"' "
System.out.println(query)
```

Skriv ut følgende: `SELECT * FROM tabell WHERE name = 'O'boy'`

Parametriserte spørringer

- Hva om vi ønsker å bruke strings eller annen brukerinput?
- Det er ikke lurt å konkatenere disse på «vanlig» måte!

Løsning: Vi bruker PreparedStatement i stedet for Statement slik:

```
int aar = 2002;
String tittel = "Rachida";
String s = "SELECT filmid FROM film WHERE title = ? AND prodyear = ?";

try {
    PreparedStatement stm = connection.prepareStatement(s);
    stm.setString(1, tittel);
    stm.setInt(2, aar);

    ResultSet res = stm.executeQuery();

    ...

} catch(SQLException e) {
    // Noe gikk galt!
    e.printStackTrace();
}
```

Nyttig SQL til oblig 6

- For datalagring snakker vi gjerne om **CRUD** – Create, Read, Update, Delete
- Create tilsvarer «`INSERT INTO`», Read tilsvarer «`SELECT`»

- I tillegg kan det være nyttig å vite om hvordan vi oppdaterer og sletter data i SQL:

- **`UPDATE`** – oppdaterer rader som tilfredsstiller en betingelse
  - ```
    UPDATE tabellnavn
    SET kolonne1 = verdi1, kolonne2 = verdi2, ...
    WHERE betingelse;
    ```
    f.eks.:
    ```
    UPDATE timeliste
    SET beskrivelse = 'Ny beskrivelse'
    WHERE timelistenr = 3;
    ```
- **`DELETE`** – sletter rader som tilfredsstiller en betingelse
  - ```
    DELETE FROM tabellnavn
    where betingelse
    ```
    f.eks.:
    ```
    DELETE FROM timeliste
    WHERE timelistenr = 8;
    ```

Oppgave

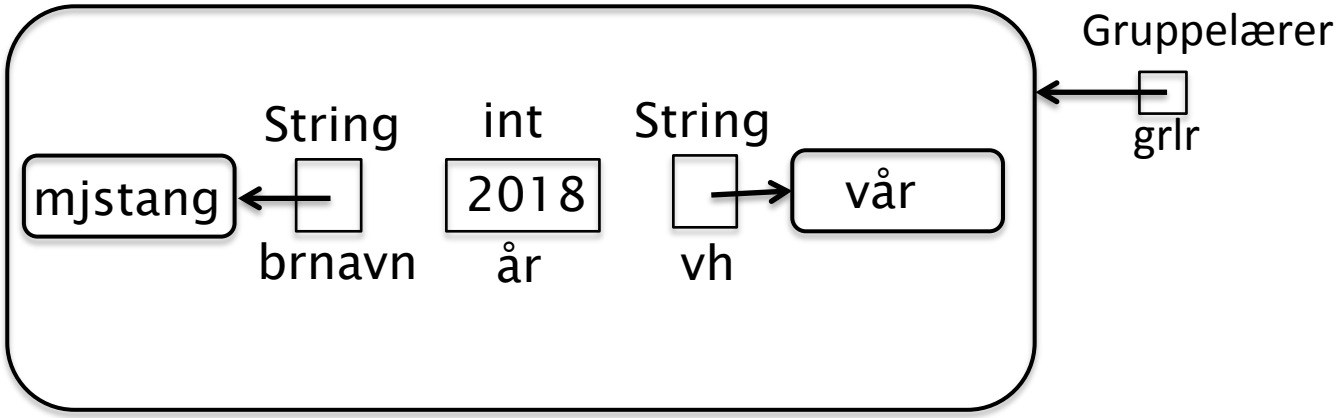
Foreslå spørringer som det er vanskelig å gjøre med select-setningen, hvor Java eller Python ville ha vært til (stor) hjelp for å få riktig svar.

Har du opplevd oppgaver som du skulle løse med SQL hvor du har savnet 'verktøy' fra Python/Java eller andre programmeringsspråk?

```
create table Gruppelærer (
        brnavn varchar(8),
        år          int,
        vh          varchar(4)
);
```

```
create table Gruppelærer (
      brnavn varchar(8),
      år         int,
      vh         varchar(4)
);
```

```
  brnavn |  år   | vh
---------+------+-----
 mjstang | 2018 | vår
 mjstang | 2017 | høst
```



http://heim.ifi.uio.no/inf1300/oppgaver/tidlEksmOppg/2011-2.pdf

```
create table Gruppelærer (
        brnavn varchar(8),
        år          int,
        vh          varchar(4)
);
```

For hver gruppelærer, finn det lengste antall semestere vedkommende har vært gruppelærer sammenhengende. F.eks. vil en som har vært gruppelærer våren 2008, våren 2009, høsten 2009, våren 2010, våren 2011 og høsten 2011 ha tre sammenhengende semestere på det meste (våren 2009, høsten 2009 og våren 2010) på grunn av brudd høsten 2008 og høsten 2010.

```
create view kodeSem as
      ( select brnavn, år*2 as sk
        from Gruppelærer
              where vh = 'vår' )
   union
      ( select brnavn, år*2 + 1 as sk
        from Gruppelærer
              where vh = 'høst' ) ;

create view diffSem as
      select k2.brnavn, k2.sk, (k2.sk - k1.sk) as diff
      from kodeSem k1, kodeSem k2
      where k1.brnavn = k2.brnavn and
           k2.sk >= k1.sk ;

create view diffHull as
      select d1.brnavn, d1.sk , d1.diff
      from diffSem d1
      where d1.diff+1 not in ( select d2.diff
                                    from diffSem d2
                                          where d2.brnavn = d1.brnavn and
                                          d2.sk = d1.sk and
                                          d2.diff > d1.diff ) ;

select R.brnavn, max(antsmh) as makssammenhengende
from ( select brnavn, sk, ( min(diff) + 1 ) as antsmh
      from diffHull
      group by brnavn, sk ) as R
group by R.brnavn ;
```

```
create table Gruppelærer (
      brnavn varchar(8),
      år         int,
      vh         varchar(4)
);
```