

Projektperiode: forelæsning mandag 27. november 2000 Java-programmer med database-adgang

- Relationsdatabaser: tabeller (relationer), tupler, attributter
- Forespørgselsproget SQL
- Databasesystemet Postgres
- Tilgang til database-server fra Java programmer
- Tilgang til database-server fra Java appletter

Refer 2000-11-22, 2000-11-23

Databasebegreber

En *relationsdatabase* består af en samling navngivne tabeller (relationer).

En *tabel* består af to ting:

- Et skema (= tabeloverskrift):

Skemaet er tabellens form; det er uforanderligt.

Skemaet angiver navn og type på tabellens felter (attributter).

- En samling tupler (= tabellinier):

Samlingen af *tupler* (poster) er tabellens indhold; den kan variere over tid.

Hvert tuple indeholder et sæt værdier for tabellens felter

Rækkefølgen af tupperne i en tabel er ligegyldig.

Eksempel på en database: studenter, kurser, eksamener

Et udsnit af tabellen Student:

eftersnavn	fornavn	studienummer
Olesen	Peter	L2143
Hansen	Erika	J00007
Funder	Ulrik	Hg0014

Et udsnit af tabellen Kursus:

kursusnummer	kursusnavn
10181	Databehandling
45621	Landbrugszoologi
15351	Miljømodeller
15311	Matematisk Grundkursus

Et udsnit af tabellen Eksamen:

studienummer	kursusnummer	karakter
L2143	010181	8
L2143	045621	9
J00007	010181	11
J00007	015311	8
L2143	015311	10
Hg0014	015311	7

Tabeller og skemaer i eksemplet

Tabeller: Student, Kursus, Eksamen

Tabellernes skemaer:

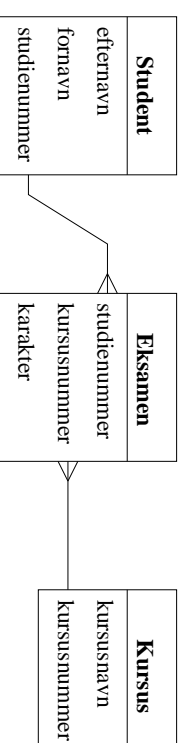
Student: (eftersnavn TEXT, fornavn TEXT, studienummer TEXT)

Kursus: (kursusnavn TEXT, kursusnummer INT)

Eksamen: (studienummer TEXT, kursusnummer INT, karakter INT)

Til hver student (studienumre) svarer nul eller flere eksamener.

Til hvert kursus (kursusnummer) svarer nul eller flere eksamener.



Operationer på relationsdatabaser

Når man arbejder med relationsdatabaser bruger man sproget SQL
SQL er opfundet af IBM ca. 1970.

I SQL skelner man ikke mellem store og små bogstaver (I modsætning til Java).

Her skriver vi SQL-ordreerne med stort, og tabelnavne og feltnavne med småt.

Indsættelse af værdier i tabellerne (SQL INSERT)

```
INSERT INTO student VALUES ('Olesen', 'Peter', '12143');
INSERT INTO student VALUES ('Hansen', 'Erika', '700007');
INSERT INTO student VALUES ('Funder', 'Ulrik', 'H90014');
INSERT INTO kursus VALUES (10181, 'Databelhandling');
INSERT INTO kursus VALUES (45621, 'Landbrugssociologi');
INSERT INTO kursus VALUES (15351, 'Miljømodeller');
INSERT INTO kursus VALUES (15311, 'Matematisk Grundkursus');
INSERT INTO eksamen VALUES ('12143', 010181, 8);
INSERT INTO eksamen VALUES ('12143', 045621, 9);
INSERT INTO eksamen VALUES ('700007', 010181, 11);
INSERT INTO eksamen VALUES ('700007', 015311, 8);
INSERT INTO eksamen VALUES ('12143', 015311, 10);
INSERT INTO eksamen VALUES ('H90014', 015311, 7);
```

Oprettelse af tabeller (SQL CREATE)

```
CREATE TABLE student (efternavn TEXT, fornavn TEXT, studienummer TEXT);
CREATE TABLE kursus (kursusnummer INT, kursusnavn TEXT);
CREATE TABLE eksamen (studienummer INT, kursusnummer INT, karakter INT);
```

PsqlFrontend

Man kan kigge på databasens tabeller, og tabellernes skemaer ved at anvende programmet PsqlFrontend (se projekthjemmesiden):

Knappen 'Show All Tables' viser alle tabeller i databasen

Knappen 'Show Table' viser indholdet (navn på felter og tupler) af en tabel

Knappen 'Show Table Info' viser definitionen af en tabel (skema)

Forespørgsler på databasen (SQL SELECT)

Vis alle kurser:

```
SELECT * FROM kursus;
```

Vis alle kurser sorteret efter kursusnavn:

```
SELECT * FROM kursus ORDER BY kursusnavn;
```

Vis alle kurser med kursusnummer 010181:

```
SELECT * FROM kursus WHERE kursusnummer = 010181;
```

Vis kursusnumre på alle kurser der har været holdt eksamen i:

```
SELECT kursusnummer FROM eksamen;
```

Samme, uden dubletter:

```
SELECT DISTINCT kursusnummer FROM eksamen;
```

Forespørgsler der involverer flere tabeller (samkøring: join)

Vis kursusnavn på alle kurser der har været holdt eksamen i.

Hvordan — kursusnavnet står jo ikke i tabellen Eksamen?

Lav en samkøring (join) mellem tabellerne Eksamen og Kursus:

```
SELECT DISTINCT kursusnavn
FROM kursus, eksamen
WHERE kursus.kursusnummer = eksamen.kursusnummer;
```

Vis alle studerende der har været til eksamen i kursus 10181:

```
SELECT DISTINCT fornavn, efternavn FROM student, eksamen
WHERE student.studienummer = eksamen.studienummer
AND kursusnummer = 10181;
```

Vis alle studerende der har været til eksamen i kurset 'Databehandling':

```
SELECT DISTINCT fornavn, efternavn FROM student, eksamen, kursus
WHERE student.studienummer = eksamen.studienummer
AND kursus.kursusnummer = eksamen.kursusnummer
AND kursus.kursusnavn = 'Databehandling';
```

Opdatering (SQL UPDATE)

Nedsæt alle karakterer mellem 7 og 11 med 1:

```
UPDATE eksamen
SET karakter = karakter-1
WHERE 7 <= karakter AND karakter <= 11;
```

Sletning (SQL DELETE)

Fjern alle eksamener med karakterer under 9:

```
DELETE FROM eksamen WHERE karakter < 9;
```

Fjern alle studerende der har fået mindst 11 i en eksamen.

```
DELETE FROM student
WHERE student.studienummer = eksamen.studienummer
AND eksamen.karakter >= 11;
```

Aggregerede udtræk: COUNT, SUM, AVG, MIN, MAX

Vis antal eksamener hver studerende har gået til:

```
SELECT studienummer, COUNT(karakter) FROM eksamen
GROUP BY studienummer;
```

Opret ny tabel ud fra forespørgsel (SQL SELECT INTO)

Resultatet af en forespørgsel (SELECT) er en anonym tabel.

Man kan direkte gemme resultatet i en ny navngiven tabel, f.eks. Eksamensantal:

```
SELECT studienummer, count(karakter) AS antal INTO eksamensantal
FROM eksamen GROUP BY studienummer;
```

Nye felter kan navngives med AS.

Den nye tabel kan bruges i nye forespørgsler:

```
SELECT efternavn, fornavn, antal FROM student, eksamensantal
WHERE student.studienummer = eksamensantal.studienummer;
```

Finde de studerende der aldrig har fået 11 eller derover?

```
SELECT * INTO stud11 FROM student;
```

```
DELETE FROM stud11
```

```
WHERE stud11.studienummer = eksamen.studienummer
AND eksamen.karakter >= 11;
```

```
SELECT * FROM stud11;
```

JDBC: Brug af relationsdatabaser fra Java-programmer

En relationsdatabaseserver er et program der passer en samling databaser.

Databaseserveren er typisk forbundet til et netværk, f.eks. Internet.

Der er mange forskellige databaseservere: DB2, Ingres, Oracle, Microsoft SQL Server, MySQL, ...

Her bruges Postgres, en gratis server fra Berkeley Univ, Californien.

Java kommunikerer med databaser via JDBC = Java Database Connectivity.

Java-programmet starter med at oprette en forbindelse til databasen.

Det sker ved hjælp af en URL (Uniform Resource Locator), som på WWW:

```
jdbc:postgres://mysql.it-c.dk/gp
```

Kommunikationsprotokollen er jdbc

Subprotokollen er postgresql

Databaseserveren kører på maskinen mysql.it-c.dk

Databasen hedder gp.

Desuden skal man angive brugernavn og løsen (password) for øjeren af databasen.

Opgave: Få forbindelse med gp fra PsqfFrontend.

Et Java-program der læser fra databasen (StudierDB.java)

```
import java.io.*; import java.sql.*;

public class StudierDB {
    public static void main(String[] args) throws SQLException {
        Connection db; // The connection to the database
        Statement st; // A statement to run queries with

        String url = "jdbc:postgresql://mysql.it-c.dk/gp";
        String usr = "gp";
        String pwd = "e2000";

        try { Class.forName("postgresql.Driver"); }
        catch (ClassNotFoundException e)
        { System.out.println("Cannot find the Postgresql driver"); }

        db = DriverManager.getConnection(url, usr, pwd);
        st = db.createStatement();

        // query database and print result (next slide)
        ...
        st.close();
        db.close();
    }
}
```

Slet en hel tabel (SQL DROP TABLE)

Vi skal ikke bruge tabellen Eksamensantal mere:

```
DROP TABLE eksamensantal;
```

```

Spørg database og læs resultatet fra databasen (StudierDB.java)
String query = "SELECT * FROM student ORDER BY fornavn";
st.execute(query);
ResultSet rs = st.getResultSet();
while (rs.next()) {
    String enavn = rs.getString("efternavn");
    String fnavn = rs.getString("fornavn");
    String stnr = rs.getString("studienummer");
    System.out.println(fnavn + " " + enavn + " (" + stnr + ")");
}

```

Nogle praktiske forhold

Klientmaskinen er den maskine på hvilken Java-programmet kører.

På klientmaskinen skal man installere Java-arkivet postgresql.jar.

Til JDK 1.1 skal man bruge postgresql.jar, til JDK 1.2 og 1.3 skal man bruge postgresql2.jar.

Kopier arkivet til C:\...\jdk118\lib eller C:\...\jdk13\lib, afhængig af JDK-version.

Java-arkivene postgresql.jar og postgresql2.jar fås på projektsiden
<http://www.it-c.dk/courses/GP/E2000/projekter.html>

Man skal også forælle Java hvor på maskinen arkivet ligger ved at sætte CLASSPATH:

```

set CLASSPATH=. ; C:\...\jdk118\lib\classes.zip ; C:\...\jdk118\lib\postgresql.jar

```

eller

```

set CLASSPATH=. ; C:\...\jdk13\lib\classes.zip ; C:\...\jdk13\lib\postgresql2.jar

```

Under MS Windows/95/98 kan dette indstilles i Klientmaskinens C:\autoexec.bat

Under MS Windows/NT/2000 skal man gå ind i My Computer | Settings | ...

At forbinde Java-appliketter med databasen

Det fungerer på samme måde som ved Java-programmer.

Arkivet postgresql.jar skal ligge udpakket på webserveren (ikke klientmaskinen).

Det udpakkede postgresql-arkiv skal ligge i det katalog der indeholder appliettens .class fil.

```

jar xf postgresql.jar

```

Databaserveren skal køre på samme maskine som webserveren.

En appliet må nemlig kun kommunikere med den maskine som applietten kommer fra (at sikkerhedshensyn).

Eksempel på appliet med databaseadgang (StudieApplet.java)

```

import java.sql.*; import java.awt.*;
import java.awt.event.*; import java.applet.Applet;

public class StudieApplet extends Applet {
    TextField stnrInd = new TextField(15);
    TextField kurnrInd = new TextField(10);
    TextField karInd = new TextField(10);
    TextArea ud = new TextArea(4, 40);
    Button tilfoej = new Button("Tilføj");
    Button soeg = new Button("Vis alle");

    Connection db;
    Statement st;

    public void init() { ... } // Called when the applet is loaded
    public void destroy() { ... } // Called when applet is removed

    class TilfoejYtter implements ActionListener { ... }
    void addTodbatabase(String stnr, int kurnr, int kar) { ... }
    class SoegGlytter implements ActionListener { ... }
    void searchDataatabase(int kurnr) { ... }
}

```

Init- og destroy-metoderne (StudiEApplEet.java)

```
public void init() {
    add(new Label("Studienummer")); add(studnrInd);
    add(new Label("Kursusnummer")); add(kurnrInd);
    add(new Label("Karakter")); add(karInd);
    add(tilfoej); tilfoej.addActionListener(new TilfoejIlytter());
    add(soeg); soeg.addActionListener(new SoegIlytter()); add(ud);
    String url = "jdbc:postgressql://mysql.it-c.dk/gp";
    String usr = "gp", pwd = "e2000";

    try {
        Class.forName("postgressql.Driver");
        db = DriverManager.getConnection(url, usr, pwd);
        st = db.createStatement();
    } catch (Exception e)
    { ud.append("Kan ikke åbne databasen: " + e.getMessage()); }

    public void destroy() {
        try {
            st.close();
        } catch (SQLException e)
        { ud.setText("Kan ikke lukke databasen: " + e.getMessage()); }
    }
}
```

Søglytter og søgemetode (StudiEApplEet.java)

```
class SoegIlytter implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        searchDatabase(Integer.parseInt(kurnrInd.getText()));
    }
}

void searchDatabase(int kurnr) {
    try {
        String query = "select fornavn, efternavn, karakter "
            + " from eksamen, student "
            + " where eksamen.studienummer = student.studienummer "
            + " and eksamen.kursusnummer = " + kurnr
            + " order by fornavn, efternavn";
        st.execute(query);
        ResultSet rs = st.getResultSet();
        ud.setText("");
        while (rs.next()) {
            String enavn = rs.getString("efternavn");
            String fnavn = rs.getString("fornavn");
            int kar = rs.getInt("karakter");
            ud.append(fnavn + " " + enavn + " fik " + kar + "\n");
        }
    } catch (SQLException e)
    { ud.append("Databasesøgning gik galt: " + e.getMessage()); }
}
```

Tilfølytter og tilføimetode (StudiEApplEet.java)

```
class TilfoejIlytter implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        addToDatabase(studnrInd.getText(),
            Integer.parseInt(kurnrInd.getText()),
            Integer.parseInt(karInd.getText()));
    }
}

void addToDatabase(String studnr, int kurnr, int kar) {
    try {
        String query = "INSERT INTO eksamen VALUES ( "
            + query + " " + studnr + " "
            + query + " " + kurnr + " "
            + query + " " + kar + " "
            + query + " )";
        ud.append(query + "\n");
        st.execute(query);
    } catch (SQLException e) {
        ud.append("Databasopdatering gik galt: " + e.getMessage());
    }
}
```

Nogle fælttyper:

SQL type	Java type	værdier
INT	int	heltal
TEXT	String	tegnstrenge
FLOAT8	double	kommatal
BOOL	boolean	logiske værdier (sand, falsk)
DATE	java.sql.Date	datoer (år, måned, dag)
TIME	java.sql.Time	klokkeslet (time, minutter, sekunder)

Husk:

- En relationsdatabase består af en samling navngivne tabeller
- En tabel (relation) består af et skema (tabeloverskrift) og en samling tupler (poster)
- Forespørgselsproget SQL bruges til at arbejde med relationsdatabaser
- Java-programmer og appletter kan kommunikere med databaser over netværket ved hjælp af JDBC
- Benyt PgsqlFrontend (enten som alm. program eller applet) til at afprøve SQL ordrer
- Kig på projekthjemmesiden for flere oplysninger