Eksamen, DSDS, forår 2009

Introduktion til Scripting, Databaser og Systemarkitektur

Jonas Holbech
IT Universitetet i København*

3. juni 2009

- *Alle hjælpemidler er tilladte, dog ikke computer og kommunikationsmidler.*
- *Eksamen er skriftlig, fire timer (9-13), og den bedømmes fra 0 til 100 procent.*
- *Eksamen består af fem hovedeopgaver, der alle ønskes løst.*
- *Skriv med kuglepen og kun på den ene side af papiret.*
- *Et godt råd: Gennemlæs opgavesættet inden du begynder at besvare de enkelte opgaver.*
- *Du bestemmer selv, om du benytter HTML eller XHTML, bare vær konsekvent.*

Introduktion

I dette opgavessæt skal du implementere dele af en online skobutik.

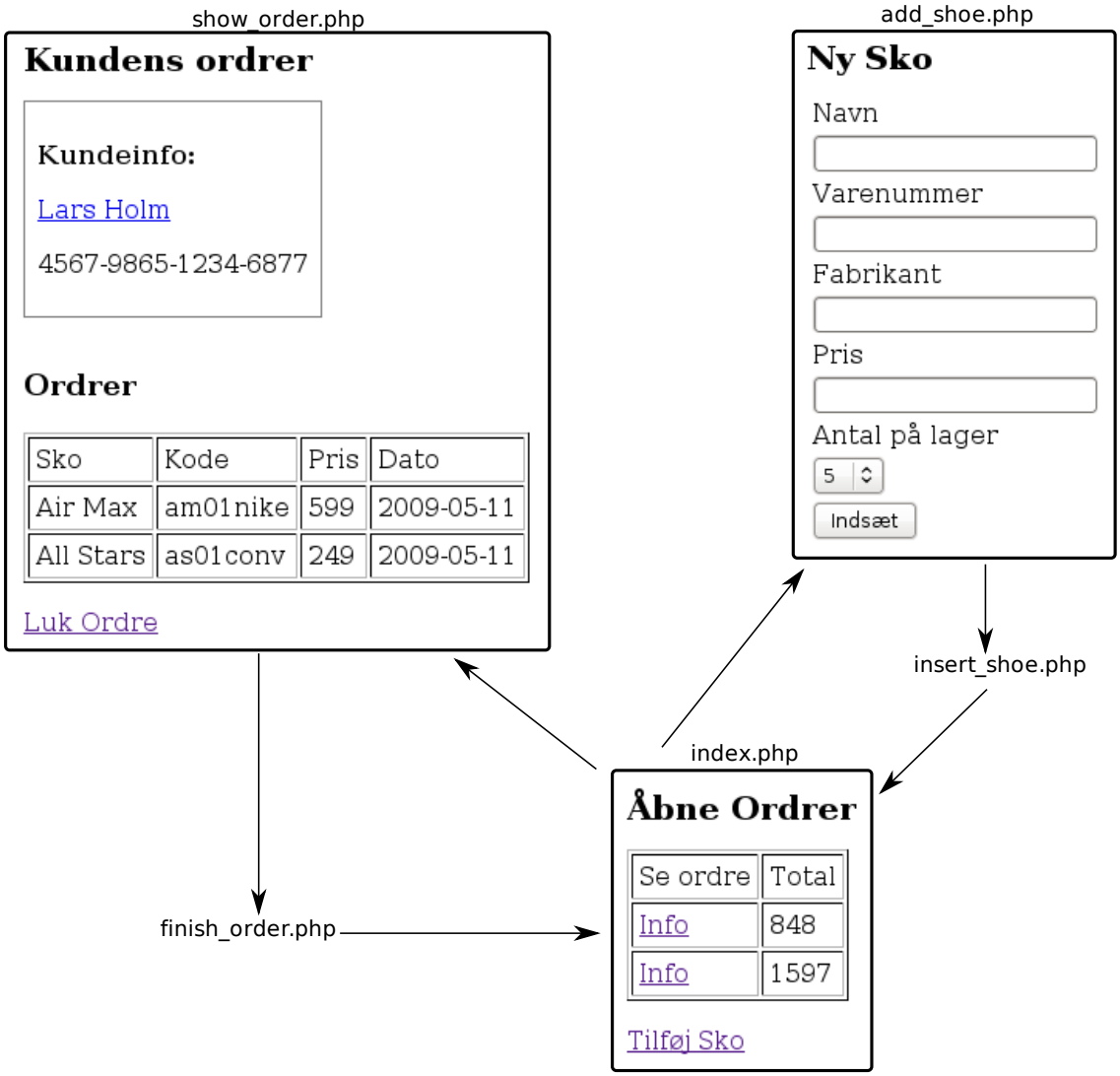
Den del af systemet, du skal arbejde med, er centreret omkring administration af kundernes ordrer. Det betyder, at hele den del af systemet, der vises i det følgende site-map, kræver man er logged ind.

Fra siden sælges en række sko, `shoes`. Disse sko kan så være købt af kunderne, `customers` og dermed blive til ordrer, `orders`.

Databasen er opbygget således at hvert køb af et par sko udgør sin egen række i tabellen `orders`, der refererer til tabellerne `shoes` og `customers`. Det er en simpel måde at håndtere ordrer på, men der er visse problematikker i det, som du bliver bedt om at kommentere på senere i opgavessættet.

*Adresse: Rued Langgaards Vej 7, 2300 København S, Danmark. Email: `holbech@itu.dk`.

Figur 1 viser et site-map for det ønskede system.



Figur 1: Site-map for den del af systemet, hvor brugeren/administratoren skal være logged ind. Et screenshot (kasser) i diagrammet angiver sider, brugeren kan se i systemet. Sidernes navne er angivet oven for hvert billede. En ikke-annoteret pil i diagrammet angiver, at en bruger kan klikke på et link i en side for at springe til en anden side. En annoteret pil i diagrammet angiver, at brugeren ved at klikke på en knap forårsager, at php-scriptet, annoteret på pilen, bliver kørt på serveren, og at php-scriptet videresender brugeren til en ny side.

Baggrund:

Filen `index.php` indeholder en oversigt over ordrer. For hver kunde med ordrer er summen af kundens ordrer angivet, samt et link til kundens ordrer. Linket går til `show_order.php`, hvor kundens `customerid` sendes med.

Filen `show_order.php` lister kundens kontaktoplysninger (på baggrund af `customerid`) og viser alle de ordrer, kunden har i databasen. I slutningen af siden er et link (Luk Ordre) til `finish_order.php`. I linket medsendes `customerid`.

Filen `finish_order.php` modtager et `customerid` og sletter alle ordrer i databasen med pågældene `customerid`. Derefter sendes brugeren tilbage til `index.php`.

Filen `add_shoe.php` indeholder en form, hvor administratoren kan tilføje nye sko.

Filen `helpers.php` indeholder en række hjælpefunktioner, der benyttes i forskellige dele af systemet. Funktionerne er beskrevet nedenfor.

Bemærk: I opgavesættet kan du frit benytte kode og funktioner fra andre delspørgsmål, uanset om du har løst opgaven eller ej.

helpers.php:
I det følgende antages det, at funktionerne `error`, `mydb_connect`, `show_header`, `show_footer`, `check_cookie`, `show_customer`, `check_cardnumber`, `check_shoecode`, `check_text`, `check_int` og `create_select` er tilgængelige i filen `helpers.php`.
Alle funktioner, du bliver bedt om at lave i dette opgavessæt, vil ligeledes være tilgængelige i filen til efterfølgende opgaver.

`error`: tager en streng som argument. Strengen udskrives og funktionen kalder `exit`, der stopper programmet.

`mydb_connect`: tager ingen argumenter og opretter forbindelse til systemets database.

`check_cookie`: tjekker, om der findes en cookie med navnet `admin`, og om cookien har værdien `loggedin`. Findes cookien, forlænges dens udløbstid med to timer. Findes den ikke kaldes funktionen `error` og programmet stopper. Funktionen skal du lave senere i opgavessættet.

`show_customer`: tager et tal (`customerid`) som argument og udskriver kontaktoplysninger på den givne kunde. Findes kunden ikke kaldes funktionen `error` og programmet stopper. Funktionen skal du lave senere i opgavessættet.

`check_cardnumber`: tager en streng som argument og tjekker, om det er et gyldigt kortnummer. Er det ikke, kaldes funktionen `error` og programmet stopper. Det regulære udtryk, funktionen benytter, svarer til det, du bliver bedt om at lave i opgave 2.2.

`check_shoecode`: tager en streng som argument og tjekker, om det er et gyldigt varenummer for sko. Er det ikke, kaldes funktionen `error` og programmet stopper. Det regulære udtryk, funktionen benytter, svarer til det, du bliver bedt om at lave i opgave 2.1

`check_int`: tager et tal som argument og tjekker, om det er et gyldigt heltal. Er det ikke, kaldes funktionen `error` og programmet stopper.

`check_text`: tager en streng som argument og tjekker, om det indeholder ulovlige tegn (SQL-Injection). Indeholder strengen det, kaldes funktionen `error`, og programmet stopper.

`create_select`: funktionen opbygger et `select`-element med hundrede `option`-elementer, som bruges til at angive mængden af en bestemt type sko på lageret. Funktionen skal du lave senere i opgavessættet.

Koden for `show_header` og `show_footer`:

```
function show_header($title){
  echo "<html><head>
      <title>$title</title>
  </head><body>
  <h1>$title</h1>";
}

function show_footer(){
  echo "</body></html>";
}
```

Opgave 1 (15 procent) - HTML

Opgave 1.1 (7 procent)

Baggrund:

Tabellen i filen `index.php` fra Figur 1 viser alle kunders ordrer i systemet. Øverste række er kollonernes overskrift. Hver efterfølgende række er et link til ordren og den samlede sum, ordren udgør.

Din Opgave:

Lav HTML-koden for tabellen, svarende til `index.php`

Cellen "Info"er et link, der pegger på `show_order.php` og for hvert link sendes den pågældende kundes `customerid` med.

1. `customerid` for første info-række er 1
2. `customerid` for anden info-række er 2

Opgave 1.2 (8 procent)

Din Opgave:

Lav HTML-koden for formen i filen `add_shoe.php`.

Formen består af seks formelementer:

1. Et tekstfelt til skoens navn, hvor `name` er `name`
2. Et tekstfelt til skoens varenummer, hvor `name` er `shoecode`
3. Et tekstfelt til skoens fabrikant, hvor `name` er `manufacturer`
4. Et tekstfelt til skoens pris, hvor `name` er `price`
5. Et `select`-element hvor man kan angive, hvor mange par, der er på lager. `name` er `stock`
    - Elementet indeholder fire `option`-elementer med værdierne 5, 10, 15 og 20
6. En `submit`-knap, hvor `value` er `Tilføj Sko`

Formens `action` er `insert_shoe.php` og `method` er `post`

Opgave 2 (10 procent) - Regulære Udtryk

Bemærk: Det forventes, at du benytter ^ og $, hvor det er relevant

Opgave 2.1 (5 procent)

Baggrund:

Hver sko i systemet har sit eget varenummer (`shoecode`), og der ønskes et regulært udtryk, der matcher gyldige varenumre.

Din Opgave:

Skriv et regulært, udtryk der matcher strenge bestående af 2 store og/eller små bogstaver, efterfulgt af 2 tal og 4 store og/eller små bogstaver.

Hint: Eksempler på gyldige varenumre: `eR48tYue`, `jr29etrS` og `RE99REGh`

Opgave 2.2 (5 procent)

Baggrund:

Kunder, der bestiller varer, skal indtaste deres kortnummer, og der ønskes et regulært udtryk, der matcher strenge, der ligner gyldige kortnumre

Din Opgave:

Skriv et regulært udtryk, der matcher gyldige kortnumre.

Et kortnummer har formen 0000-0000-0000-0000 (4 tal, bindestreg, 4 tal, bindestreg, 4 tal, bindestreg og 4 tal).

Eksempler på gyldige kortnumre: `7865-4567-7822-0666` og `6584-5687-2132-4587`

Opgave 3 (20 procent) - PHP

Opgave 3.1 (7 procent)

Baggrund:

Funktionen `show_header` tager et argument (`$title`), som udskrives i <`title`> elementet og i et <`h1`> element. Hvis funktionen kaldes med en tom streng som argument, udskrives ugyldig HTML. (<`h1`><`/h1`> er ikke tilladt).

Din Opgave:

Lav funktionen `show_header` om, så den tjekker om `$title` argumentet er en tom streng. Er det en tom streng, skal der i både <`title`> og <`h1`> elementet udskrives "Velkommen"i stedet for den tomme streng. Funktionen skal stadig hedde `show_header`

Opgave 3.2 (7 procent)

Baggrund:

For at gøre det lettere at tjekke, om en bruger er logget ind og for at undgå, at brugerens cookie udløber, ønskes funktionen `check_cookie`.

Din Opgave:

Skriv funktionen `check_cookie`.

Funktionen skal tjekke, om cookien `admin` findes, og om dens værdi er `loggedin`.

Passer ovenstående, skal cookien forlænges med to timer.

Findes cookien ikke, eller indeholder den en forkert værdi, skal funktionen `error` kaldes med en passende fejlmeddelelse.

Hint: Det er kun navn, værdi og tid der er interessant for denne cookie.

Man forlænger en cookies udløbstid ved at sætte en ny cookie med samme navn/værdi og en ny udløbstid.

Opgave 3.3 (6 procent)

Baggrund:

`select` elementet i filen `add_shoe.php` indeholder fire statiske `option`-elementer. Det skal laves om, så der er 100 `option`-elementer med værdierne et til hundrede (1-100).

Din Opgave:

Skriv funktionen `create_select`, der skal udskrive eller returnere et `select`-element svarende til opgave 1.2 med hundrede `option`-elementer. `option`-elementerne skal have værdierne fra 1-100 begge inklusive.

`select`-elementets `name` attribut skal som tidligere have værdien `stock`.

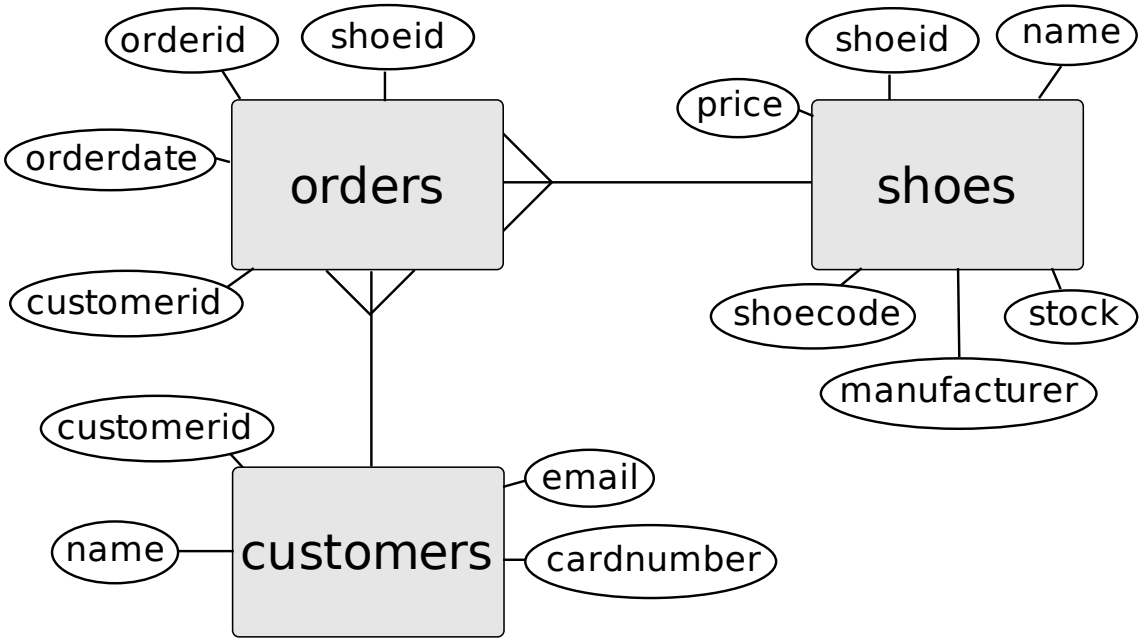
Vis efterfølgende, hvordan funktionen kaldes, så den udskriver elementerne.

Hint: Det forventes at du benytter en løkke i din besvarelse af opgaven.

Opgave 4 (25 procent) - SQL / Datamodel

Baggrund:

Datamodellen for systemet er angivet nedenfor som et ER-diagram.



Figur 2: E/R-diagram for systemet. De firkantede kasser angiver entiteterne (tabellerne), og de ovale cirkler angiver attributter (felter), tilknyttet de enkelte tabeller. En-til-mange relationer er angivet med "krageføddder".

Tabellerne `shoes` og `customers` er oprettet i MySQL med nedenstående SQL-kommandoer. Husk at felter, der er erklæret `AUTO_INCREMENT`, starter med 1 og tæller opad.

Feltet `shoecode` fra tabellen `shoes` er skoens varenummer.

Feltet `cardnumber` fra tabellen `customers` er kundens kortnummer.

```
CREATE TABLE shoes (
    shoeid INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(200),
    shoecode VARCHAR(8),
    manufacturer VARCHAR(200),
    price INT,
    stock INT
) ENGINE=InnoDB;
```

```
INSERT INTO shoes (name, shoecode, manufacturer, price, stock)
    VALUES ('Air Max', 'am01nike', 'Nike', 599, 10);
INSERT INTO shoes (name, shoecode, manufacturer, price, stock)
    VALUES ('All Stars', 'as01conv', 'Converse', 249, 5);
INSERT INTO shoes (name, shoecode, manufacturer, price, stock)
    VALUES ('Pink Lady', 'pl02nike', 'Nike', 499, 4);

CREATE TABLE customers (
    customerid INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255),
    email VARCHAR(255) UNIQUE,
    cardnumber VARCHAR(19)
) ENGINE=InnoDB;

INSERT INTO customers (name, email, cardnumber)
    VALUES ('Lars Holm', 'lh@gmail.com', '4567-9865-1234-6877');
INSERT INTO customers (name, email, cardnumber)
    VALUES ('Ulla Hansen', 'uh@ulla.dk', '7812-3548-5621-6475');
INSERT INTO customers (name, email, cardnumber)
    VALUES ('Nanna Hoff', 'nanna@hotmail.com', '8456-3215-5487-5615');
```

Opgave 4.1 (5 procent)

Konstruér en SQL-kommando til at oprette tabellen `orders`. Tabellen skal have følgende felter:

- `orderid` skal være af typen `INT`. Være primær nøgle og benytte `AUTO_INCREMENT`
- `shoeid` skal være af typen `INT` og referere til feltet `shoeid` i tabellen `shoes`
- `customerid` skal være af typen `INT` og referere til feltet `customerid` i tabellen `customers`
- `orderdate` skal være af typen `DATE`

Opgave 4.2 (5 procent)

Din Opgave:

Skriv en eller flere SQL-kommandoer, der indsætter følgende fem rækker/ordrer i tabellen `orders`.

Til feltet `orderdate` kan du bare benytte MySQL's funktion `NOW()`.

- Lars har bestilt et par "Air Max"og et par "All Stars"

- Ulla har bestilt to par "Pink Lady"og et par "Air Max"

Opgave 4.3 (5 procent)

Din Opgave:

Lav en SQL kommando, der henter kundernes `customerid` og den samlede sum, kunden har bestilt for. Der ønskes ikke rækker for kunder, der ikke har bestilt.

Hint:

Du får brug for MySQL funktionen `SUM` og `GROUP BY`

Opgave 4.4 (5 procent)

Din Opgave:

Vis outputtet af følgende SQL kommando i et skema

```
SELECT email, COUNT(orderid)
    FROM customers
    LEFT JOIN orders
        ON customers.customerid=orders.customerid
    GROUP BY orders.customerid
    ORDER BY COUNT(orderid) ASC;
```

Opgave 4.5 (5 procent)

Baggrund:

Som databasen er opbygget nu, vil der være to næsten ens rækker (redundans) i tabellen `orders`, hvis en kunde køber to par ens sko. Det bør undgås.

Derudover er der et problem. Hvis en kunde den ene dag bestiller et par sko til 400, og de dagen efter bliver sat op til 500. Den pris kunden er gået med til vil i så fald have ændret sig.

Din Opgave:

1. Beskriv hvordan man ville kunne udvide den eksisterende database, så man undgår redundans i `orders` tabellen?

2. Beskriv hvordan man ville kunne løse problemet med priser, der skifter efter kunden har bestilt, men inden ordren er blevet behandlet?

Bemærk: Der ønskes en kort forklaring af de nødvendige ændringer. Du må gerne supplere med et simpelt ER-diagram (som Figur 2)

Du må gerne samle de to spørgsmål i et svar.

Opgave 5 (30 procent) - Web Service

Opgave 5.1 (7 procent)

Baggrund:

På siden `show_order.php` fra Figur 1 er der en lille kasse med kundens kontakoplysninger. Disse oplysninger er hentet fra databasen på baggrund af kundens id. Du skal lave funktionen, der genererer kassens indhold.

Din Opgave:

Skriv funtionen `show_customer`, der tager et tal (`customerid`) som argument, henter den pågældende række i databasen og udskriver en overskrift (kundeinfo:), kundens navn, kontonummer og email.

Selve kassen er en <`div`>, og hver element kan passende ligge i en paragraf (<`p`>). Brugerens navn skal være et `mailto:` link, der benytter brugerens email. I screenshottet er der tilføjet yderligere formattering, (eks en ramme). Det kan du se bort fra.

Findes kunden ikke, skal du kalde funktionen `error` med en passende meddelelse.

Bemærk: Du kan antage at der allerede er oprettet forbindelse til databasen og at alle funktioner fra filen `helpers.php` er tilgængelige.

Opgave 5.2 (7 procent)

Din Opgave:

Lav scriptet `show_order.php`.

Scriptet modtager formvariablen `customerid` fra `index.php` (`$_REQUEST['customerid']`).

Scriptet skal tjekke formvariablen, udskrive kundens kontaktoplysninger via funktionen `show-customer` fra opgave 5.1, liste alle kundens ordrer, og til sidst skal der være et link til `finish_order.php`, hvor `customerid` skal sendes med.

Bemærk:

Det er hele scriptet du skal skrive, scriptet skal derfor udgøre en komplet og validerende HTML side.

Du skal selv inkludere nødvendige filer.

Husk at tjekke om brugeren er logget ind samt at tjekke alle formvariable.

Opgave 5.3 (8 procent)

Din Opgave:

Skriv scriptet `insert_shoe.php`.

Scriptet modtager formvariabler fra `add_shoe.php` og skal indsætte en række i tabellen `shoes`.

Når der er indsat en række, sendes brugeren tilbage til `index.php` med PHP funktionen `header`.

Bemærk:

Det er hele scriptet du skal skrive.

Du skal selv inkludere nødvendige filer.

Du kan antage at formvariablen `price` skal indeholde et heltal (Integer)

Husk at tjekke om brugeren er logget ind samt at tjekke alle formvariable.

Opgave 5.4 (8 procent)

Baggrund:

Scriptet `finish_order.php` afslutter en kundes ordrer ved simpelthen at slette dem. Tanken er, at administratoren har sendt varen, opkrævet betaling og efterfølgende slettet ordren.

Din Opgave:

Skriv scriptet `finish_order.php`. Scriptet modtager et `customerid`, der benyttes til at slette pågældende rækker i databasen.

Når rækkerne er slettet skal brugeren sendes tilbage til filen `index.php`.

Har brugeren ikke nogle ordrer i databasen, skal funktionen `error` kaldes med en passende fejlmeddelelse.

Bemærk:

Det er hele scriptet du skal skrive.

Du skal selv inkludere nødvendige filer.

Husk at tjekke om brugeren er logget ind samt at tjekke alle formvariable.