A11: Last Year's Exam

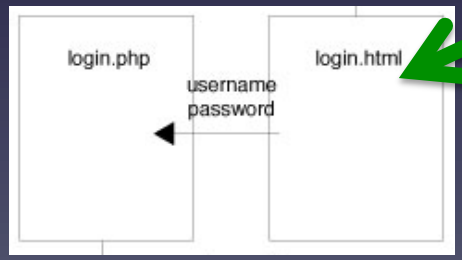
Agenda

- Design of Site map and Web-structure (3)
- Design of data model (1)
- Design of database transactions (2)
- Construction of HTML and PHP scripts (3)

Exercise 3: Design of Site map and Webstructure

General requirements for your "*La Pizzeria*" Web Service:

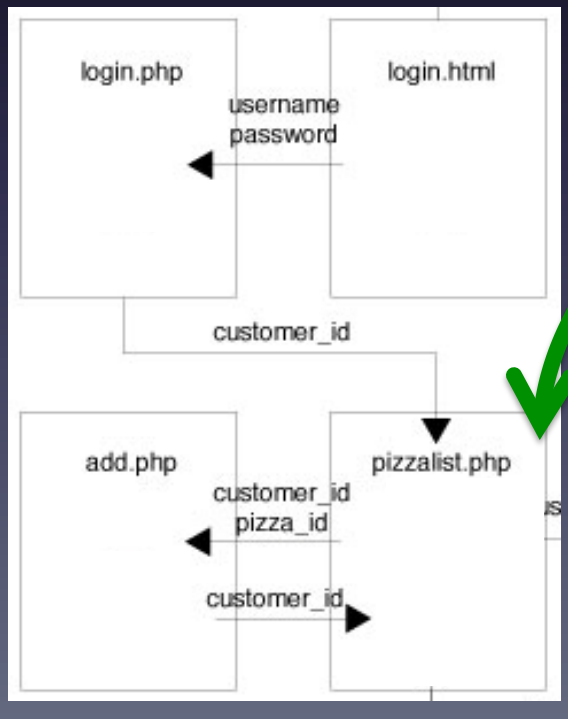
1. **Logging in:** Already registered customers should be able to log in to the service by entering their username and password. The actual registering of customers is *not* part of this assignment (i.e., it is perfectly fine to just manually insert a few customers into some appropriate MySQL database table of registered customers). If the log in is successful (i.e., the information entered by the customer matches the contents of the database), the customer should then be able to order pizzas (see further below). If the log in fails, the customer should get some appropriate error message. Your service only needs to perform the password validation once upon first entry (i.e., it does not have to validate users that "jump" into the middle of the service by "guessing" the name of a subsequent PHP script).



Exercise 3: Design of Site map and Webstructure

General requirements for your "*La Pizzeria*" Web Service:

2. **Pizza list:** The customer should be presented with a list of pizzas (name, ingredients, and price) that can be ordered, sorted by price (see the top part of Screen Shot 1 on last page of the exam). Each pizza should come with a link for adding it to the order basket the contents of which the customer should also somehow be able to see (see below).

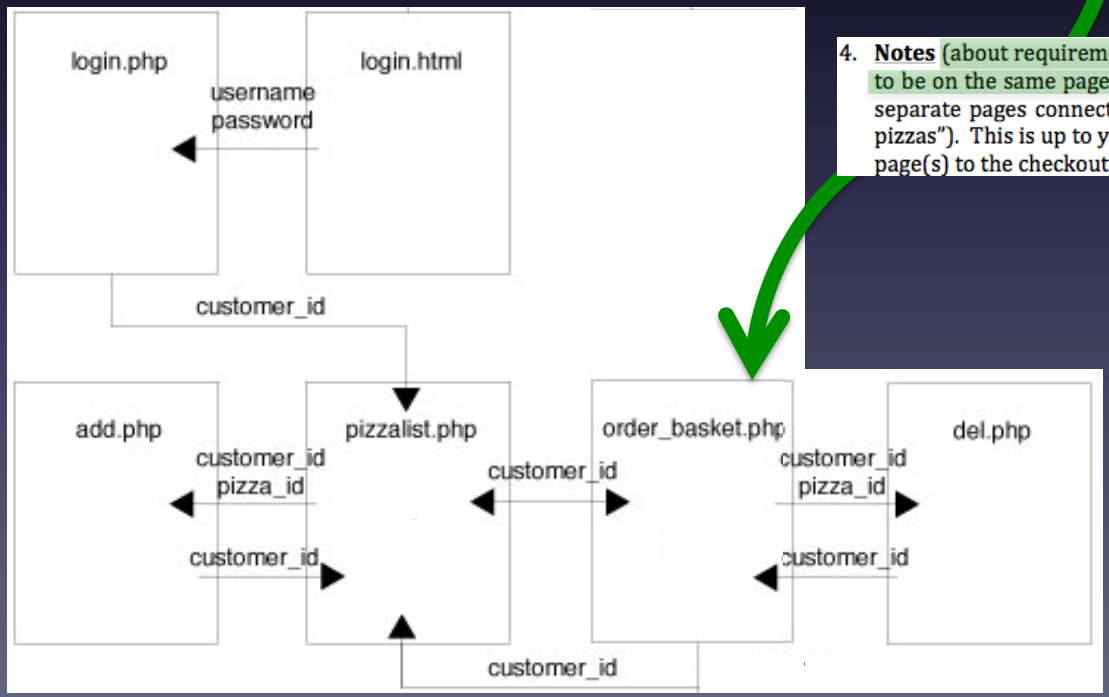


Exercise 3: Design of Site map and Webstructure

General requirements for your "*La Pizzeria*" Web Service:

3. **Order basket:** The customer should also be presented with the contents of the order basket (see the bottom part of Screen Shot 1). In the order basket, a pizza should be listed just with its name and price (i.e., without its ingredients). Each pizza should come with a link for removing it from the order basket. The basket should be stored as normal "persistent state" (i.e., in a suitable table in the database) and not as "session state".

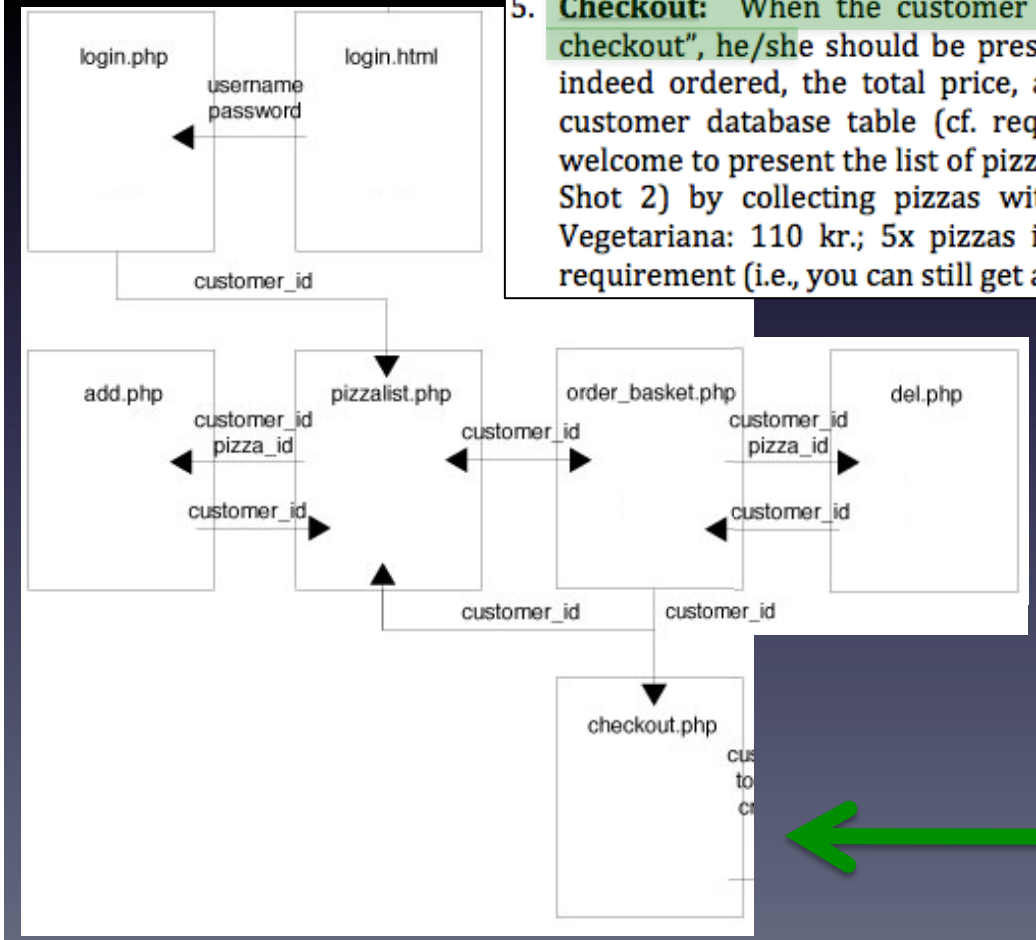
4. **Notes** (about requirements 2. and 3.): The "Pizza list" and the "Order basket" do not have to be on the same page (as is the case on Screen Shot 1); they can just as easily reside on separate pages connected by appropriate links (e.g., "show my basket" and "order more pizzas"). This is up to you. There also needs to be a way of taking the customer from these page(s) to the checkout (e.g., a "Proceed to checkout" link as in Screen Shot 1).



Exercise 3: Design of Site map and Webstructure

General requirements for your "*La Pizzeria*" Web Service:

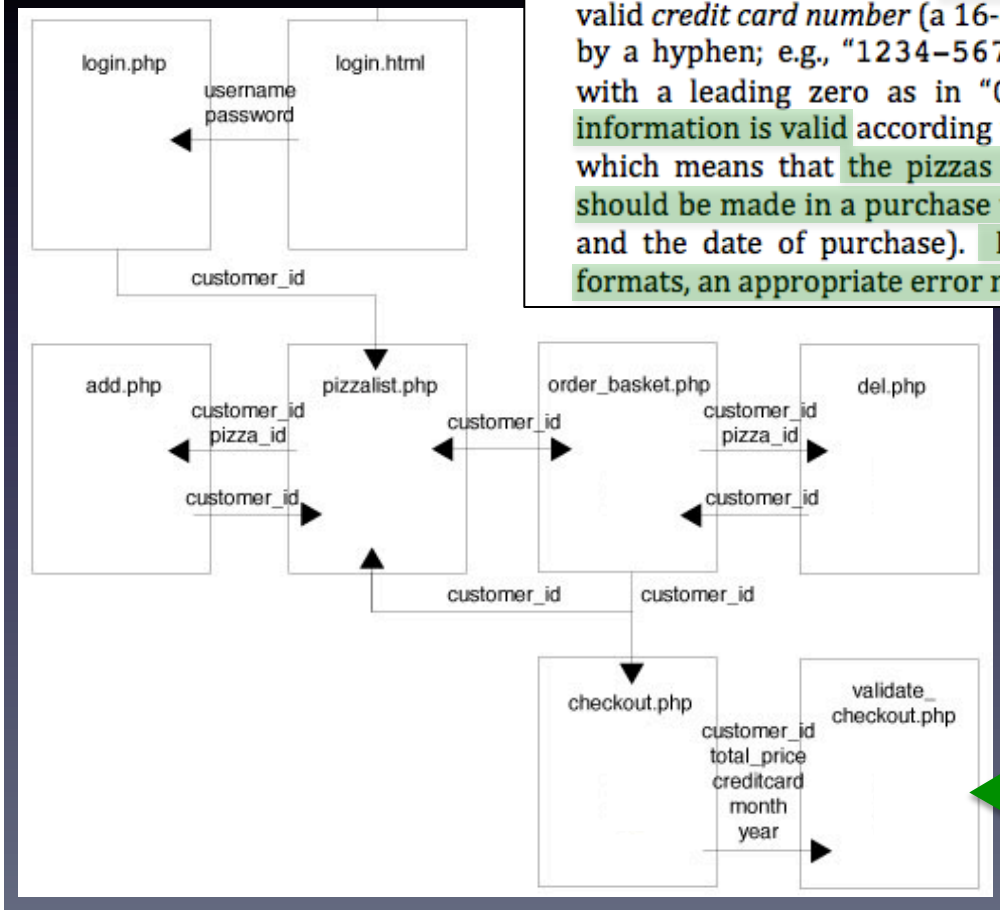
5. **Checkout:** When the customer is done ordering pizzas and has clicked "proceed to checkout", he/she should be presented with an alphabetically ordered list of the pizzas indeed ordered, the total price, and a delivery address as originally registered in the customer database table (cf. requirement 1. and Screen Shot 2). You are also very welcome to present the list of pizzas in a slightly more fancy version (not shown in Screen Shot 2) by collecting pizzas with the same name (e.g., 3x Margherita: 135 kr.; 2x Vegetariana: 110 kr.; 5x pizzas in total: 245 kr.). However, this is not a mandatory requirement (i.e., you can still get a 12 for a "perfect solution" without this feature).



Exercise 3: Design of Site map and Webstructure

General requirements for your "*La Pizzeria*" Web Service:

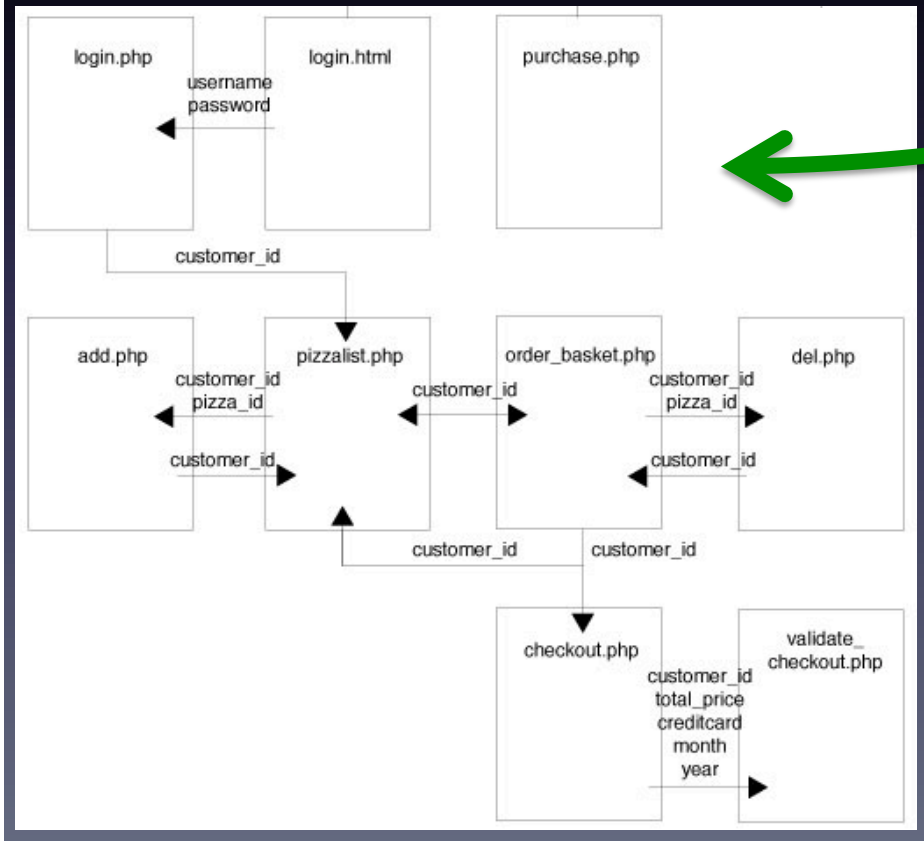
6. **Credit card validation:** Before a purchase is finalized, the customer needs to enter a valid *credit card number* (a 16-digit number where each 4 consecutive digits are separated by a hyphen; e.g., "1234-5678-9012-3456") and a legal *month* (two digits, possibly with a leading zero as in "09") and two-digit *year* (e.g., "15"). If the credit card information is valid according to the above formats, the purchase is considered *completed* which means that the pizzas should be removed from the order basket and an entry should be made in a purchase table (an entry should consist of a customer, the total price and the date of purchase). If the credit card information does not comply with the formats, an appropriate error message should be given.



Exercise 3: Design of Site map and Webstructure

General requirements for your "*La Pizzeria*" Web Service:

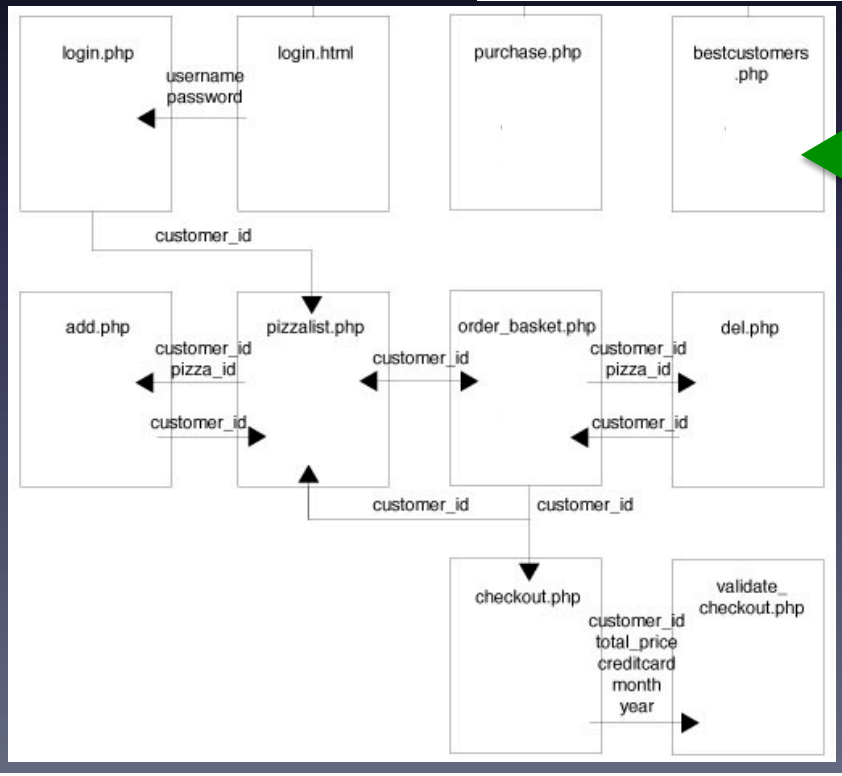
7. **Purchase history:** There should be an administration page (without password protection) that will show a list of all the purchases made (name, purchase price, and date of purchase), ordered by the purchase date.



Exercise 3: Design of Site map and Webstructure

General requirements for your "*La Pizzeria*" Web Service:

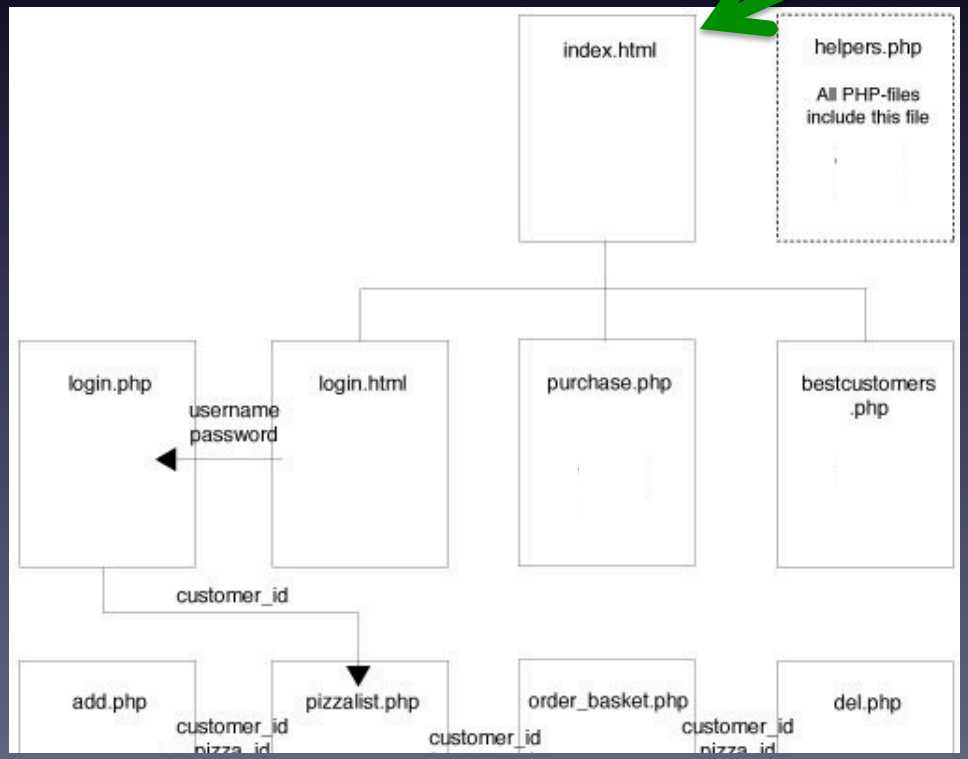
8. **Best customers:** There should also be an administration page (without password protection) that will show the top five best customers (i.e., the five customers who have spent the most money, in total, over time, buying pizzas), sorted by total amount spent. It should state the name of the customer, how many purchases he/she has made, the average amount spent per purchase, and, of course, the total sum spent.



Exercise 3: Design of Site map and Webstructure

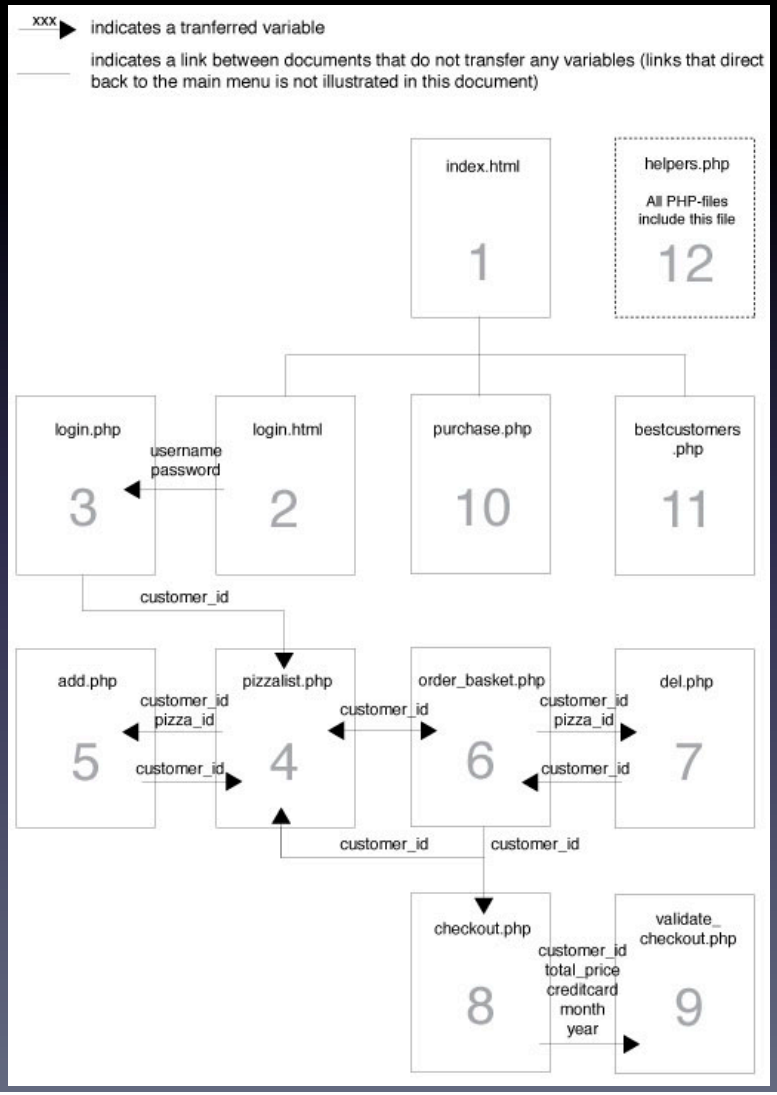
General requirements for your "*La Pizzeria*" Web Service:

9. **index.html**: Finally, there should be a simple "`index.html`" file with links to use the service. It should contain a link to start the service (i.e., to the customer "log in" page) and links to each of the two administration pages.



Exercise 3: Design of Site map and Webstructure

Done!



Agenda

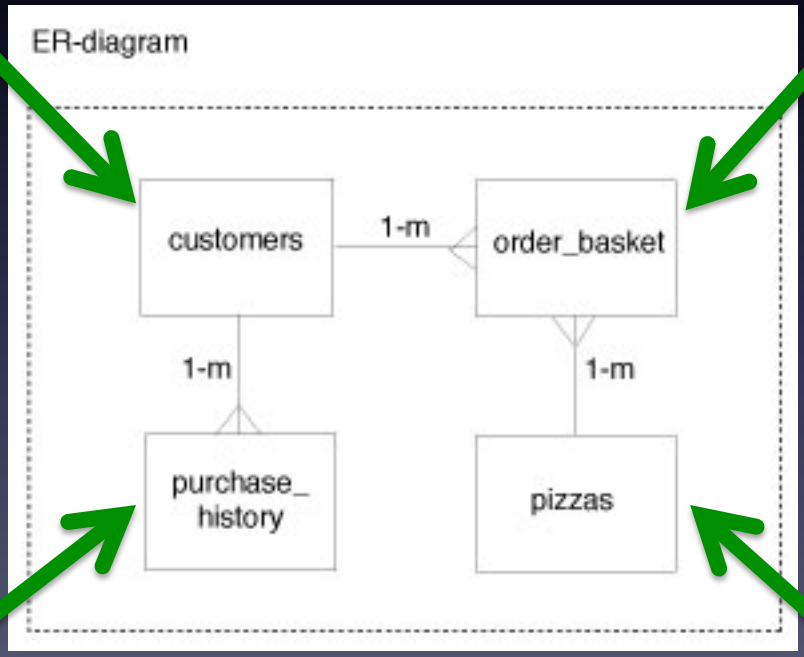
- ~~Design of Site map and Web-structure (3)~~
- Design of data model (1)
- Design of database transactions (2)
- Construction of HTML and PHP scripts (3)

Exercise 1: Design of data model (1a)

MySQL Database Design

Logging in: Already registered customers should be able to log in to the service by entering their username and password.

Order basket: The customer should also be presented with the contents of the order basket (see the bottom part of Screen Shot 1). In the order basket, a pizza should be listed just with its name and price (i.e., without its ingredients). Each pizza should come with a



Purchase history:
show a list of all the purchases made (name, purchase price, and date of purchase)

Pizza list: The customer should be presented with a list of pizzas (name, ingredients, and price)

Exercise 1: Design of data model (1a)

MySQL Database Design

Table 1: customers

customer_id	username	password	address
1	charlotte	charlottepass	Somewhere 1
2	casper	casperpass	Somewhere 2
3	frank	frankpass	Somewhere 3
4	mads	madspass	Somewhere 4
5	johan	johanpass	Somewhere 5
6	sebastian	sebastianpass	Somewhere 6

AUTO_INCREMENT
PRIMARY KEY

Table 3: order_basket

order_id	customer_id	pizza_id
109	1	2
108	1	1
111	3	1
110	3	3

AUTO_INCREMENT
PRIMARY KEY

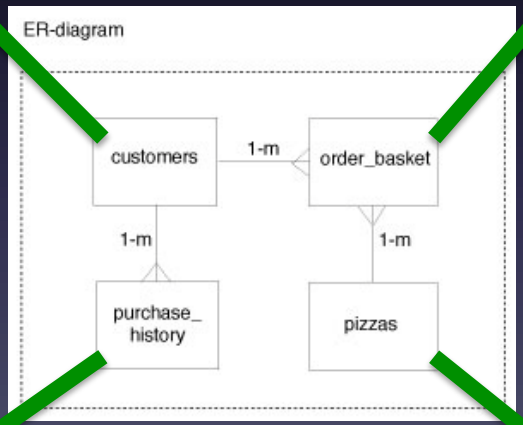


Table 4: purchase_history

purchase_id	customer_id	total_price	date
3	2	320	2012-01-19 18:56:
4	2	160	2012-01-19 18:58:38
5	3	45	2012-01-19 18:59:12
6	1	120	2012-01-19 19:01:23
7	5	385	2012-01-19 19:02:09
8	4	45	2012-01-19 19:03:07
9	1	255	2012-01-19 23:05:56
10	3	195	2012-01-19 23:54:50
11	3	100	2012-01-20 00:00:08
12	2	100	2012-01-20 00:01:18

AUTO_INCREMENT
PRIMARY KEY

Table 2: pizzas

pizza_id	pizza_name	pizza_ingredients	pizza_price
1	Margherita	Tomato and Cheese	45
2	Pepperoni	Tomato, Cheese, and Pepperoni	50
3	Calzone	Tomato, Cheese, Ham, Onions, and Oregano	55
4	Capricciosa	Tomato, Cheese, Mushrooms, Artichokes, Ham, and Olives	60
5	Vegetariana	Tomato, Cheese, Mushrooms, Olives, Bell Pepper, and Onions	65

AUTO_INCREMENT
PRIMARY KEY

FOREIGN KEY

Exercise 1: Design of data model (1a)

MySQL Database Design

Table 1: customers

customer_id	username	password	address
1	charlotte	charlottepass	Somewhere 1
2	casper	casperpass	Somewhere 2
3	frank	frankpass	Somewhere 3
4	mads	madspass	Somewhere 4
5	johan	johanpass	Somewhere 5
6	sebastian	sebastianpass	Somewhere 6

AUTO_INCREMENT
PRIMARY KEY

Table 2: pizzas

pizza_id	pizza_name	pizza_ingredients	pizza_price
1	Margherita	Tomato and Cheese	45
2	Pepperoni	Tomato, Cheese, and Pepperoni	50
3	Calzone	Tomato, Cheese, Ham, Onions, and Oregano	55
4	Capricciosa	Tomato, Cheese, Mushrooms, Artichokes, Ham, and Olives	60
5	Vegetariana	Tomato, Cheese, Mushrooms, Olives, Bell Pepper, and Onions	65

AUTO_INCREMENT
PRIMARY KEY

FOREIGN KEY

Table 3: order_basket

order_id	customer_id	pizza_id
109	1	2
108	1	1
111	3	1
110	3	3

AUTO_INCREMENT
PRIMARY KEY

Table 4: purchase_history

purchase_id	customer_id	total_price	date
3	2	320	2012-01-19 18:56:13
4	2	160	2012-01-19 18:58:38
5	3	45	2012-01-19 18:59:12
6	1	120	2012-01-19 19:01:23
7	5	385	2012-01-19 19:02:09
8	4	45	2012-01-19 19:03:07
9	1	255	2012-01-19 23:05:56
10	3	195	2012-01-19 23:54:50
11	3	100	2012-01-20 00:00:08
12	2	100	2012-01-20 00:01:18

AUTO_INCREMENT
PRIMARY KEY

description.txt:

Exercise 1a Table Design:

Følgende fire tabeller er oprettet i MySQL databasen:

Table 1: customers: Denne tabel indeholder stamoplysninger om kunderne. Data i denne tabel kan pt kun indsættes, redigeres eller fjernes "manuelt" , men under virkelige omstændigheder ville kunden kunne gøre det automatisk via en "opret en bruger"-funktion i webservicen.

Table 2: pizzas: Denne tabel indeholder stamoplysninger om pizzaerne. Data i denne tabel kan pt kun indsættes, redigeres eller fjernes "manuelt", men under virkelige omstændigheder ville administratorer af webservicen kunne gøre det automatisk via en administratorfunktion.

Table 3: order_basket: Denne tabel indeholder oplysninger om "ikke-afsluttede" ordrer, dvs. kunders bestillinger af én eller flere pizzaer, som endnu ikke er betalt. Data i denne tabel indsættes og fjernes automatisk, idet en kunde til- eller fravælger en pizza. En ordre fjernes også automatisk fra denne tabel, idet betalingen gennemføres. Tabellen fungerer på den måde hovedsageligt som midlertidig lagring af data. For at knytte en ordre op på de korrekte kunde- og pizzaoplysninger, linker tabellen desuden til "customers" og "pizzas".

Table 4: purchase_history: Denne tabel indeholder oplysninger om samtlige ordrer, der er foretaget af samtlige kunder. Data i denne tabel indsættes automatisk, idet en kunde gennemfører en ordre, dvs. betaler. Tabellen er derved det permanente lagringssted for de informationer, der ved betaling slettes fra "order_basket", dog med lidt ekstra informationer. For at knytte en ordre op på de korrekte kundeoplysninger, linker tabellen desuden til "customers".

Antal og opdeling af tabellerne er foretaget med henblik på at minimere "redundancy".

Exercise 1: Design of data model (1a)

MySQL Database Design

Exercise 1a - Table Design

Table 1: customers

customer_id	username	password	address
1	charlotte	charlottepass	Somewhere 1
2	casper	casperpass	Somewhere 2
3	frank	frankpass	Somewhere 3
4	mads	madspass	Somewhere 4
5	johan	johanpass	Somewhere 5
6	sebastian	sebastianpass	Somewhere 6

AUTO_INCREMENT
PRIMARY KEY

FOREIGN KEY

ER-diagram

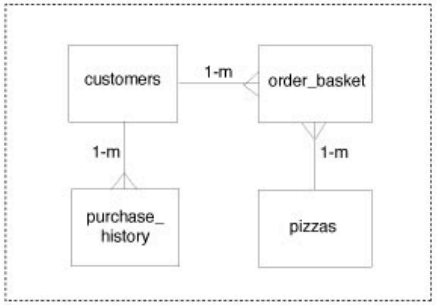


Table 2: pizzas

pizza_id	pizza_name	pizza_ingredients	pizza_price
1	Margherita	Tomato and Cheese	45
2	Pepperoni	Tomato, Cheese, and Pepperoni	50
3	Calzone	Tomato, Cheese, Ham, Onions, and Oregano	55
4	Capricciosa	Tomato, Cheese, Mushrooms, Artichokes, Ham, and Olives	60
5	Vegetariana	Tomato, Cheese, Mushrooms, Olives, Bell Pepper, and Onions	65

AUTO_INCREMENT
PRIMARY KEY

FOREIGN KEY

FOREIGN KEY

Table 3: order_basket

order_id	customer_id	pizza_id
109	1	2
108	1	1
111	3	1
110	3	3

AUTO_INCREMENT
PRIMARY KEY

Table 4: purchase_history

purchase_id	customer_id	total_price	date
3	2	320	2012-01-19 18:56:13
4	2	160	2012-01-19 18:58:38
5	3	45	2012-01-19 18:59:12
6	1	120	2012-01-19 19:01:23
7	5	385	2012-01-19 19:02:09
8	4	45	2012-01-19 19:03:07
9	1	255	2012-01-19 23:05:56
10	3	195	2012-01-19 23:54:50
11	3	100	2012-01-20 00:00:08
12	2	100	2012-01-20 00:01:18

AUTO_INCREMENT
PRIMARY KEY

Exercise 1: Design of data model (1b)

MySQL Table Definitions

Table 1: customers

customer_id	username	password	address
1	charlotte	charlottepass	Somewhere 1
2	casper	casperpass	Somewhere 2
3	frank	frankpass	Somewhere 3
4	mads	madspass	Somewhere 4
5	johan	johanpass	Somewhere 5
6	sebastian	sebastianpass	Somewhere 6

AUTO_INCREMENT
PRIMARY KEY

```
CREATE TABLE customers(
customer_id INT AUTO_INCREMENT PRIMARY KEY,
username VARCHAR(20) NOT NULL,
password VARCHAR(20) NOT NULL,
address VARCHAR (200) NOT NULL);
```

Table 2: pizzas

pizza_id	pizza_name	pizza_ingredients	pizza_price
1	Margherita	Tomato and Cheese	45
2	Pepperoni	Tomato, Cheese, and Pepperoni	50
3	Calzone	Tomato, Cheese, Ham, Onions, and Oregano	55
4	Capricciosa	Tomato, Cheese, Mushrooms, Artichokes, Ham, and Olives	60
5	Vegetariana	Tomato, Cheese, Mushrooms, Olives, Bell Pepper, and Onions	65

AUTO_INCREMENT
PRIMARY KEY

FOREIGN KEY

```
CREATE TABLE pizzas(
pizza_id INT AUTO_INCREMENT PRIMARY KEY,
pizza_name VARCHAR(100) NOT NULL,
pizza_ingredients VARCHAR(200) NOT NULL,
pizza_price INT(4) NOT NULL);
```

Table 3: order_basket

order_id	customer_id	pizza_id
109	1	2
108	1	1
111	3	1
110	3	3

AUTO_INCREMENT
PRIMARY KEY

```
CREATE TABLE order_basket(
order_id INT AUTO_INCREMENT PRIMARY KEY,
customer_id INT NOT NULL,
pizza_id INT NOT NULL,
FOREIGN KEY (customer_id) REFERENCES customers (customer_id),
FOREIGN KEY (pizza_id) REFERENCES pizzas (pizza_id)
);
```

Table 4: purchase_history

purchase_id	customer_id	total_price	date
3	2	320	2012-01-19 18:56:13
4	2	160	2012-01-19 18:58:38
5	3	45	2012-01-19 18:59:12
6	1	120	2012-01-19 19:01:23
7	5	385	2012-01-19 19:02:09
8	4	45	2012-01-19 19:03:07
9	1	255	2012-01-19 23:05:56
10	3	195	2012-01-19 23:54:50
11	3	100	2012-01-20 00:00:08
12	2	100	2012-01-20 00:01:18

AUTO_INCREMENT
PRIMARY KEY

```
CREATE TABLE purchase_history(
purchase_id INT AUTO_INCREMENT PRIMARY KEY,
customer_id INT NOT NULL,
total_price INT NOT NULL,
date DATETIME NOT NULL,
FOREIGN KEY (customer_id) REFERENCES customers (customer_id)
);
```

Exercise 1: Design of data model (1b)

MySQL Table Definitions

```
CREATE TABLE customers(
customer_id INT AUTO_INCREMENT PRIMARY KEY,
username VARCHAR(20) NOT NULL,
password VARCHAR(20) NOT NULL,
address VARCHAR (200) NOT NULL);
```

```
CREATE TABLE pizzas(
pizza_id INT AUTO_INCREMENT PRIMARY KEY,
pizza_name VARCHAR(100) NOT NULL,
pizza_ingredients VARCHAR(200) NOT NULL,
pizza_price INT(4) NOT NULL);
```

```
CREATE TABLE order_basket(
order_id INT AUTO_INCREMENT PRIMARY KEY,
customer_id INT NOT NULL,
pizza_id INT NOT NULL,
FOREIGN KEY (customer_id) REFERENCES customers (customer_id),
FOREIGN KEY (pizza_id) REFERENCES pizzas (pizza_id)
);
```

```
CREATE TABLE purchase_history(
purchase_id INT AUTO_INCREMENT PRIMARY KEY,
customer_id INT NOT NULL,
total_price INT NOT NULL,
date DATETIME NOT NULL,
FOREIGN KEY (customer_id) REFERENCES customers (customer_id)
);
```

description.txt:

Exercise 1b - Table Definitions:

Table 1: customers: I denne tabel er "customer_id" defineret som primary key og med auto_increment, således at hver kunde ved oprettelsen automatisk får deres eget unikke id. Kundens adresse kan med fordel være opdelt i flere kolonner, men er for overskuelighedens skyld her samlet i blot én, nemlig "address".

Table 2: pizzas: I denne tabel er "pizza_id" defineret som primary key og med auto_increment, således at hver type pizza ved oprettelsen automatisk får sit eget unikke id.

Table 3: order_basket: I denne tabel er "customer_id" og "pizza_id" defineret som foreign keys for "customer_id" og "pizza_id" i hhv. "customers" og pizzas"tabellerne. Dette linker tabellen sammen med "customers" og "pizzas" tabellerne.

Table 4: purchase_history: I denne tabel er "customer_id" defineret som foreign key for "customer_id" i tabellen "customers", hvilket linker de to tabeller sammen.

Exercise 1: Design of data model (1c)

MySQL Data Insertions

```
INSERT INTO customers (username, password, address)
VALUES ('charlotte', 'charlottepass', 'Somewhere 1'),
('casper', 'casperpass', 'Somewhere 2'),
('frank', 'frankpass', 'Somewhere 3'),
('mads', 'madspass', 'Somewhere 4'),
('johan', 'johanpass', 'Somewhere 5'),
('sebastian', 'sebastianpass', 'Somewhere 6');
```

```
CREATE TABLE customers(
customer_id INT AUTO_INCREMENT PRIMARY KEY,
username VARCHAR(20) NOT NULL,
password VARCHAR(20) NOT NULL,
address VARCHAR (200) NOT NULL);
```

```
INSERT INTO pizzas
(pizza_name, pizza_ingredients, pizza_price)
VALUES ('Margherita', 'Tomato and Cheese', '45'),
('Pepperoni', 'Tomato, Cheese, and Pepperoni', '50'),
('Calzone', 'Tomato, Cheese, Ham, Onions, and Oregano', '55'),
('Capricciosa', 'Tomato, Cheese, Mushrooms, Artichokes, Ham, and Olives', '60'),
('Vegetariana', 'Tomato, Cheese, Mushrooms, Olives, Bell Pepper, and Onions', '65');
```

```
CREATE TABLE pizzas(
pizza_id INT AUTO_INCREMENT PRIMARY KEY,
pizza_name VARCHAR(100) NOT NULL,
pizza_ingredients VARCHAR(200) NOT NULL,
pizza_price INT(4) NOT NULL);
```

Agenda

- ~~Design of Site map and Web-structure (3)~~
- ~~Design of data model (1)~~
- Design of database transactions (2)
- Construction of HTML and PHP scripts (3)

Exercise 2: Design of database transactions

2_database_transactions.sql:

```
SELECT purchase_history.customer_id,
SUM(total_price),
COUNT(purchase_history.customer_id),
AVG(total_price),username FROM purchase_history, customers
WHERE customers.customer_id=purchase_history.customer_id
GROUP BY username ORDER BY SUM(total_price) DESC LIMIT 5;
```

```
SELECT purchase_history.customer_id, username, total_price, date
FROM purchase_history, customers
WHERE customers.customer_id=purchase_history.customer_id
ORDER BY date;
```

```
SELECT * FROM customers;
```

```
SELECT * FROM pizzas;
```

```
INSERT INTO order_basket (customer_id, pizza_id)
VALUES ('$customer_id', '$pizza_id');
```

```
SELECT order_basket.pizza_id, pizza_name, pizza_price
FROM order_basket, pizzas
WHERE pizzas.pizza_id=order_basket.pizza_id
AND customer_id=$customer_id;
```

```
DELETE FROM order_basket
WHERE pizza_id = '$pizza_id'
AND customer_id=$customer_id
LIMIT 1;
```

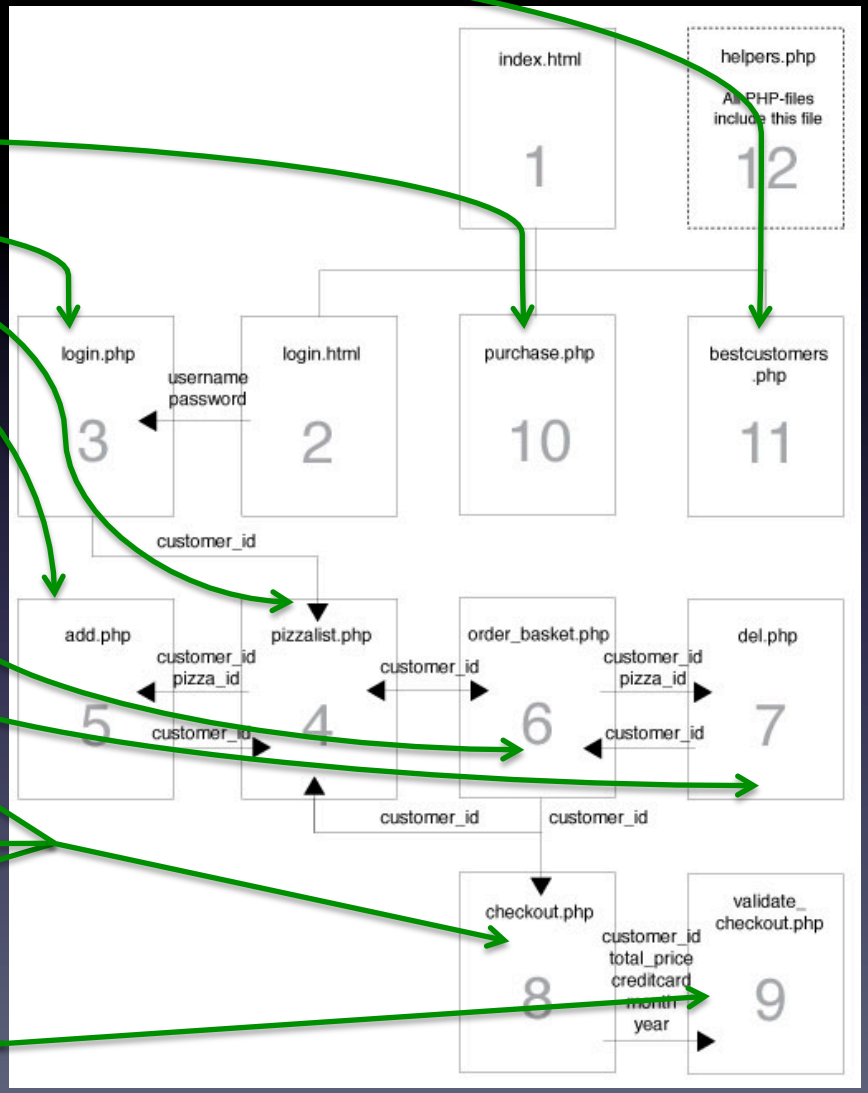
```
SELECT pizza_name, pizza_price
FROM order_basket, pizzas
WHERE pizzas.pizza_id=order_basket.pizza_id
AND customer_id=$customer_id
ORDER BY pizza_name;
```

```
SELECT SUM(pizza_price)
FROM order_basket, pizzas
WHERE pizzas.pizza_id=order_basket.pizza_id
AND customer_id=$customer_id;
```

```
SELECT address FROM customers WHERE customer_id = '$customer_id';
```

```
INSERT INTO purchase_history (customer_id, total_price, date)
VALUES ('$customer_id', '$total_price', '$date');
```

```
DELETE FROM order_basket
WHERE customer_id = '$customer_id';
```



Exercise 2: Design of database transactions

2_database_transactions.sql:

```
SELECT purchase_history.customer_id,
SUM(total_price),
COUNT(purchase_history.customer_id),
AVG(total_price),username FROM purchase_history, customers
WHERE customers.customer_id=purchase_history.customer_id
GROUP BY username ORDER BY SUM(total_price) DESC LIMIT 5;
```

```
SELECT purchase_history.customer_id, username, total_price, date
FROM purchase_history, customers
WHERE customers.customer_id=purchase_history.customer_id
ORDER BY date;
```

```
SELECT * FROM customers;
```

```
SELECT * FROM pizzas;
```

```
INSERT INTO order_basket (customer_id, pizza_id)
VALUES ('$customer_id', '$pizza_id');
```

```
SELECT order_basket.pizza_id, pizza_name, pizza_price
FROM order_basket, pizzas
WHERE pizzas.pizza_id=order_basket.pizza_id
AND customer_id=$customer_id;
```

```
DELETE FROM order_basket
WHERE pizza_id = '$pizza_id'
AND customer_id=$customer_id
LIMIT 1;
```

```
SELECT pizza_name, pizza_price
FROM order_basket, pizzas
WHERE pizzas.pizza_id=order_basket.pizza_id
AND customer_id=$customer_id
ORDER BY pizza_name;
```

```
SELECT SUM(pizza_price)
FROM order_basket, pizzas
WHERE pizzas.pizza_id=order_basket.pizza_id
AND customer_id=$customer_id;
```

```
SELECT address FROM customers WHERE customer_id = '$customer_id';
```

```
INSERT INTO purchase_history (customer_id, total_price, date)
VALUES ('$customer_id', '$total_price', '$date');
```

```
DELETE FROM order_basket
WHERE customer_id = '$customer_id';
```

description.txt:

Exercise 2 - Database Transactions:

SQL queries used in relation to page 3 login.php:
Validating username and password
SELECT * FROM customers WHERE username = '$username';

The query selects all data from the table "customers" where the username is the same as the username typed in by the user.

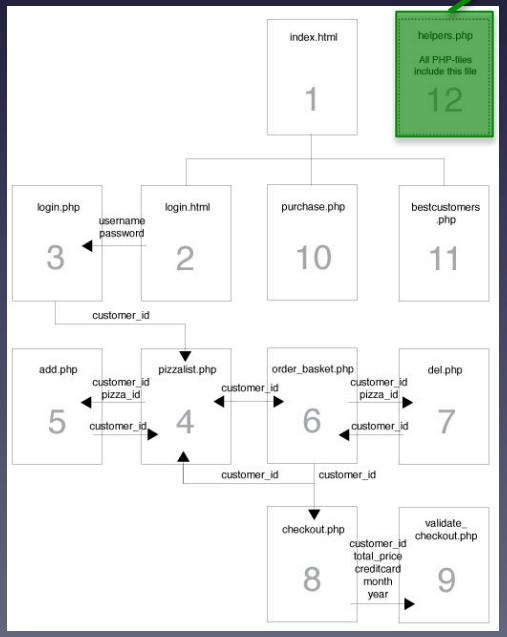
#SQL queries used in relation to page 4 pizzalist.php:
Generating the pizzalist
SELECT * FROM pizzas ORDER BY pizza_price;
The query selects all data from the table "pizzas" and sorts the data by the pizzas prices.

.... And so on

Agenda

- ~~Design of Site map and Web-structure (3)~~
- ~~Design of data model (1)~~
- ~~Design of database transactions (2)~~
- Construction of HTML and PHP scripts (3)

Exercise 4: Construction of HTML and PHP scripts



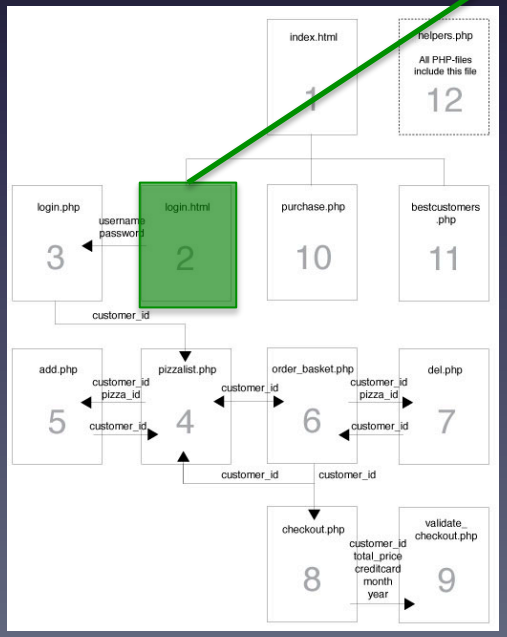
helper.php

```php
//Function that connects to the MySQL database
function mydb_connect() {
    $dbhost = "mysql.itu.dk";
    ....

//Functions that validate input
function chk_creditcard($creditcard, $customer_id){
    if ( preg_match('/^[0-9]{4}-[0-9]{4}-[0-9]{4}-[0-9]{4}$/', $creditcard) == 0 ) {
        error_checkout("Please type a valid credit card number",$customer_id);
    }
}
```

+ other validations

```php
function error($message) {
    echo "<html>
          <body>
          <p>Error: $message</p>
          <p>Go back and <a href='login.html'>try again</a></p>
          </body>
          </html>" ;
    exit;
}
```

Exercise 4: Construction of HTML and PHP scripts



login.html

```
<form action="login.php" method="post">
    <p>Username: <input type="text" name="username" size="40" /></p>
    <p>Password: <input type="password" name="password" size="40" /> </p>
    <p><input type="submit" value="Login" /> </p>
</form>
```

Login Page

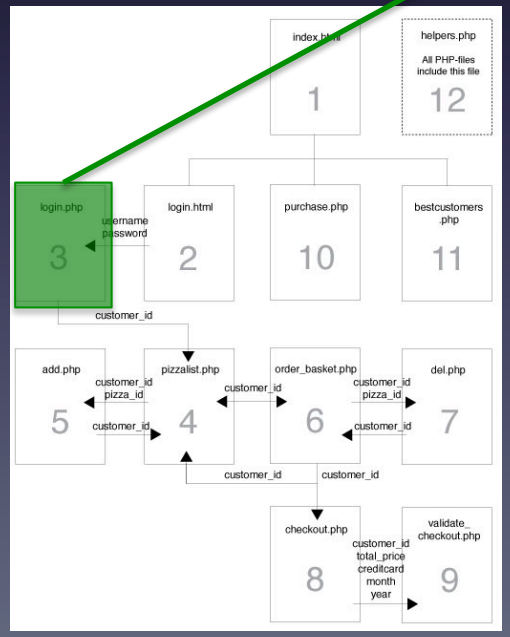
Username:

Password:

Login

Go to the main menu

Exercise 4: Construction of HTML and PHP scripts

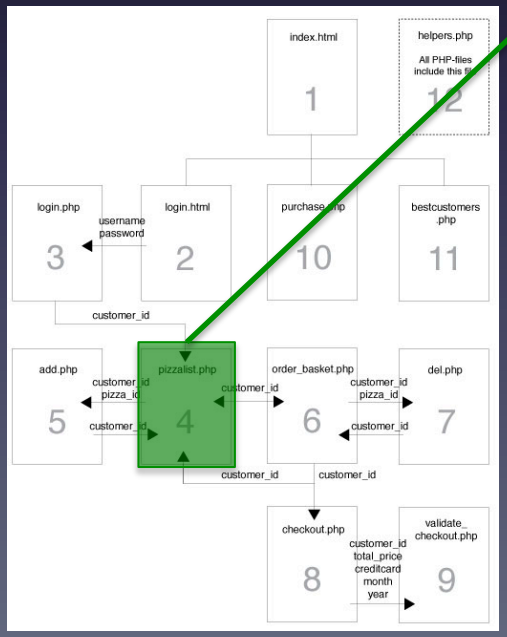


login.php

```php
/*The following commands receive the username and password from the login.html
file and store them in variables.*/
$username = $_REQUEST['username'];
$password = $_REQUEST['password'];
 /*The following commands use the standard functions "mysql_query"
 and "mysql_fetch_array" to request and retrieve the data that matches the given
 username and store it in the variables $rows and $row.*/
 $rows = mysql_query("SELECT * FROM customers WHERE username = '$username';");
 $row = mysql_fetch_array($rows);

 .
 .
 .

if ($password != $row['password']) {
    error("You have typed the wrong password");
}

$customer_id = $row['customer_id'];
header("Location: pizzalist.php?customer_id=$customer_id");
```

Notice: 'helper' function

Exercise 4: Construction of HTML and PHP scripts



pizzalist.php

```php
$customer_id = $_REQUEST['customer_id'];

echo '<table border="1"><tr><th>Pizza name</th><th>Ingredients</th>
<th>Price</th><th>Add to order basket</th></tr>';

$rows = mysql_query ("SELECT * FROM pizzas ORDER BY pizza_price;");
while ($row = mysql_fetch_array($rows)) {
    $pizza_name=$row ['pizza_name'];
    $pizza_ingredients=$row ['pizza_ingredients'];
    $pizza_price=$row ['pizza_price'];
    $pizza_id=$row ['pizza_id'];
    echo "<tr>
            <td>$pizza_name</td>
            <td>$pizza_ingredients</td>
            <td>$pizza_price kr</td>
            <td><a href='add.php?customer_id=$customer_id&pizza_id=$pizza_id'>Add</a></td>
        </tr>";
}

echo "</table>";

echo "<p>Show my <a href='order_basket.php?customer_id=$customer_id'>order basket</a></p>";
echo "<p>Proceed to <a href='checkout.php?customer_id=$customer_id'>checkout</a></p>";
```

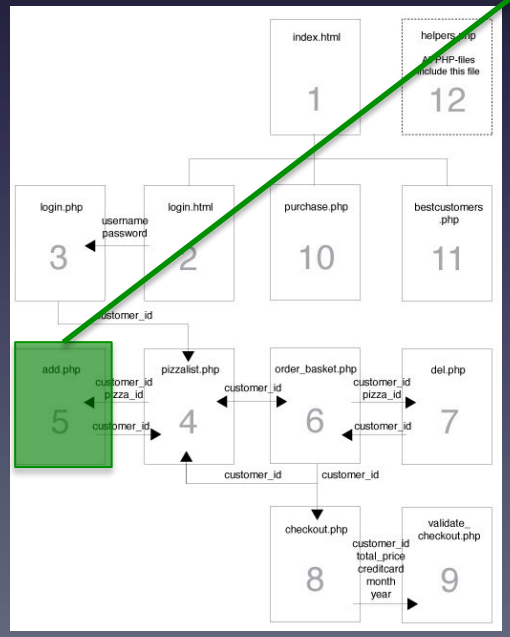
Pizzalist

Pizza name	Ingredients	Price	Add to order basket
Margherita	Tomato and Cheese	45 kr	Add
Pepperoni	Tomato, Cheese, and Pepperoni	50 kr	Add
Calzone	Tomato, Cheese, Ham, Onions, and Oregano	55 kr	Add
Capricciosa	Tomato, Cheese, Mushrooms, Artichokes, Ham, and Olives	60 kr	Add
Vegetariana	Tomato, Cheese, Mushrooms, Olives, Bell Pepper, and Onions	65 kr	Add

Show my order basket

Proceed to checkout

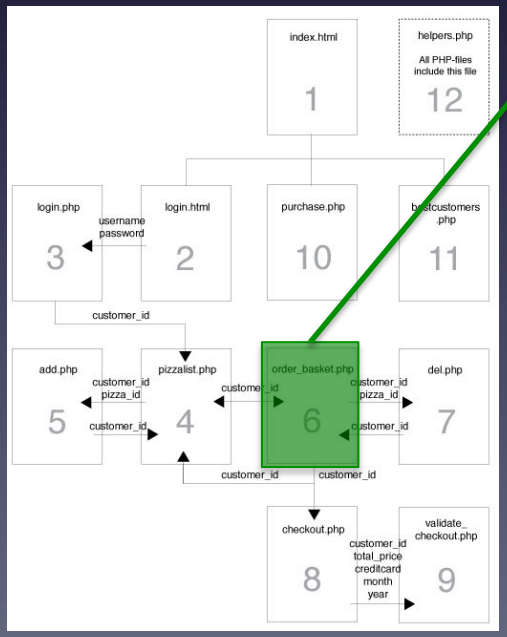
Exercise 4: Construction of HTML and PHP scripts



add.php

```
$customer_id = $_REQUEST['customer_id'];
$pizza_id = $_REQUEST['pizza_id'];

mysql_query ("  INSERT INTO order_basket (customer_id, pizza_id)
                VALUES ('$customer_id', '$pizza_id');");

header("Location: pizzalist.php?customer_id=$customer_id");
```

Exercise 4: Construction of HTML and PHP scripts



order_basket.php

```php
$customer_id = $_REQUEST['customer_id'];

$rows = mysql_query ("SELECT order_basket.pizza_id, pizza_name, pizza_price
FROM order_basket, pizzas
WHERE pizzas.pizza_id=order_basket.pizza_id AND customer_id=$customer_id;");

echo '<table border="1"><tr><th>Pizza name</th>
<th>Price</th><th>Delete from Order Basket</th></tr>';

while ($row = mysql_fetch_array($rows)) {
    $pizza_name=$row ['pizza_name'];
    $pizza_price=$row ['pizza_price'];
    $pizza_id=$row ['pizza_id'];
    echo "<tr>
            <td>$pizza_name</td>
            <td>$pizza_price kr</td>
            <td><a href='del.php?customer_id=$customer_id&pizza_id=$pizza_id'>Delete</a></td>
        </tr>";
}
echo "</table>";


echo "<p> Order more <a href='pizzalist.php?customer_id=$customer_id'>pizzas</a></p>";
echo "<p>Proceed to <a href='checkout.php?customer_id=$customer_id'>checkout</a></p>";
```

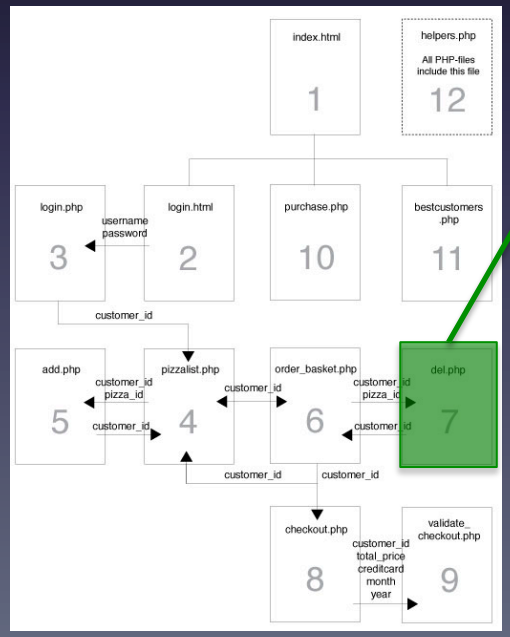
Order Basket

Pizza name	Price	Delete from Order Basket
Margherita	45 kr	Delete
Margherita	45 kr	Delete
Calzone	55 kr	Delete
Calzone	55 kr	Delete
Calzone	55 kr	Delete
Vegetariana	65 kr	Delete

Order more pizzas

Proceed to checkout

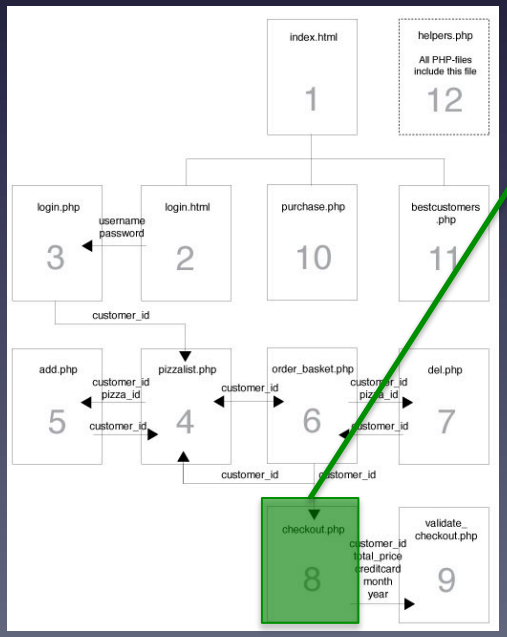
Exercise 4: Construction of HTML and PHP scripts



del.php

```php
$customer_id = $_REQUEST['customer_id'];
$pizza_id = $_REQUEST['pizza_id'];

mysql_query ("DELETE FROM order_basket
              WHERE pizza_id = '$pizza_id'
              AND customer_id=$customer_id LIMIT 1;");

header("Location: order_basket.php?customer_id=$customer_id");
```

Exercise 4: Construction of HTML and PHP scripts



checkout.php

```php
$customer_id = $_REQUEST['customer_id'];

echo "<p><b>You have ordered:</b></p>";
echo '<table border="1"><tr><th>Pizza</th><th>Price</th></tr>';

 $rows = mysql_query ("SELECT pizza_name, pizza_price
                       FROM order_basket, pizzas
                       WHERE pizzas.pizza_id=order_basket.pizza_id
                       AND customer_id=$customer_id
                       ORDER BY pizza_name;");
while ($row = mysql_fetch_array($rows)) {
    $pizza_name=$row ['pizza_name'];
    $pizza_price=$row ['pizza_price'];
    echo "<tr>
            <td>$pizza_name</td>
            <td>$pizza_price kr</td>
          </tr>";
}
echo "</table>";
echo "<p><b>The total price is:</b></p>";
$rows = mysql_query("SELECT SUM(pizza_price)
                     FROM order_basket, pizzas
                     WHERE pizzas.pizza_id=order_basket.pizza_id
                     AND customer_id=$customer_id;");
$row = mysql_fetch_array($rows);
$total_price=$row ['SUM(pizza_price)'];
echo "$total_price kr";
echo "<p><b>The order will be delivered to:</b></p>";
$rows = mysql_query("SELECT address FROM customers WHERE customer_id = '$customer_id';");
$row = mysql_fetch_array($rows);
$address=$row ['address'];
echo "$address";

echo "<form action='validate_checkout.php'>
          <input type='hidden' name='customer_id' value='$customer_id'/>
          <input type='hidden' name='total_price' value='$total_price'/>
          <p>Credit Card Number:
              <input type='text' name='creditcard' size='25'/> (xxxx-xxxx-xxxx-xxxx)
          </p>
          <p>Expiry: <input type='text' name='month' size='3'/> /
              <input type='text' name='year' size='3'/> (mm-yy)
          </p>
          <input type='submit' value='Purchase' />
      </form>";

echo "<p> Order more <a href='pizzalist.php?customer_id=$customer_id'>pizzas</a></p>";
```

Checkout

You have ordered:

Pizza	Price
Calzone	55 kr
Calzone	55 kr
Calzone	55 kr
Margherita	45 kr
Margherita	45 kr
Vegetariana	65 kr

The total price is:

320 kr

The order will be delivered to:

Somewhere 3

Credit Card Information

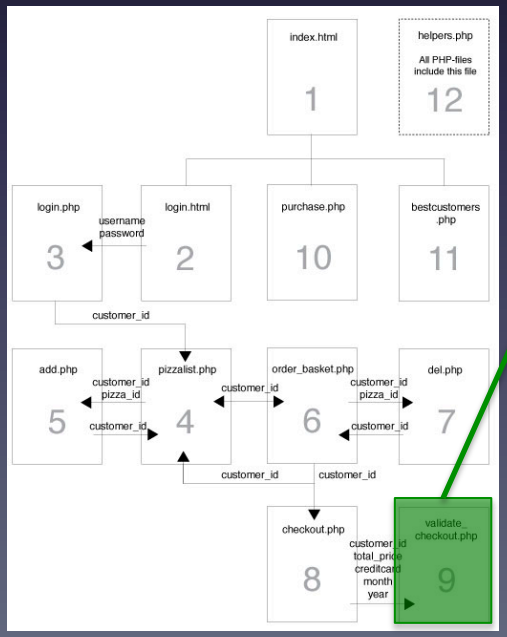
Credit Card Number: (xxxx-xxxx-xxxx-xxxx)

Expiry: / (mm-yy)

Purchase

Order more pizzas

Exercise 4: Construction of HTML and PHP scripts



validate_checkout.php

```php
$customer_id = $_REQUEST['customer_id'];
$total_price=$_REQUEST['total_price'];
$creditcard=$_REQUEST['creditcard'];
$month=$_REQUEST['month'];
$year=$_REQUEST['year'];

chk_creditcard($creditcard, $customer_id);
chk_month($month, $customer_id);
chk_year($year, $customer_id);
```

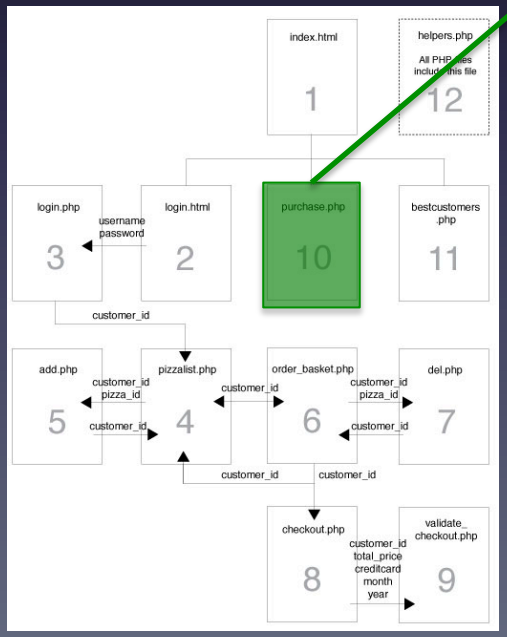
Notice: 'helper' functions

```php
$date =  date("Y-m-d H:i:s", time());

mysql_query ("INSERT INTO purchase_history (customer_id, total_price, date)
              VALUES ('$customer_id', '$total_price', '$date');");

mysql_query ("DELETE FROM order_basket WHERE customer_id = '$customer_id';");

show_header("The transaction is completed");

echo " Your pizza delivery is on its way!";
```

The transaction is completed

Your pizza delivery is on its way!

Exercise 4: Construction of HTML and PHP scripts



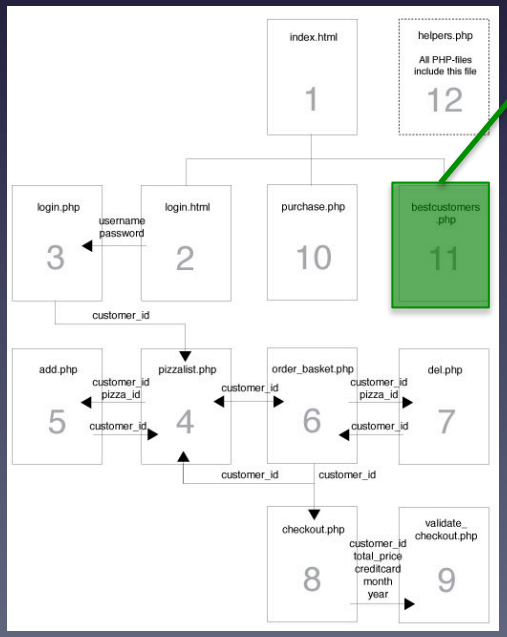
purchase.php

```
echo '<table border="1"><tr><th>Customer name</th><th>Purchase Price</th><th>Date</th></tr>';
$rows = mysql_query ("SELECT purchase_history.customer_id, username, total_price, date
                      FROM purchase_history, customers
                      WHERE customers.customer_id=purchase_history.customer_id
                      ORDER BY date;");
while ($row = mysql_fetch_array($rows)) {
    $username=$row ['username'];
    $total_price=$row ['total_price'];
    $date=$row ['date'];
    echo "<tr>
            <td>$username</td>
            <td>$total_price kr</td>
            <td>$date</td>
          </tr>";
}
echo "</table>";
```

History of all purchases

Customer name	Purchase Price	Date
casper	320 kr	2012-01-19 18:56:13
casper	160 kr	2012-01-19 18:58:38
frank	45 kr	2012-01-19 18:59:12
charlotte	120 kr	2012-01-19 19:01:23
johan	385 kr	2012-01-19 19:02:09
mads	45 kr	2012-01-19 19:03:07
charlotte	255 kr	2012-01-19 23:05:56
frank	195 kr	2012-01-19 23:54:50
frank	100 kr	2012-01-20 00:00:08
casper	100 kr	2012-01-20 00:01:18
casper	115 kr	2012-01-20 02:19:13
mads	105 kr	2012-01-20 02:24:31

Exercise 4: Construction of HTML and PHP scripts



bestcustomers.php

```
echo '<table border="1">
        <tr><th>Customer name</th>
            <th>Number of purchases</th>
            <th>Average amount per purchase</th>
            <th>Total Sum</th>
        </tr>';

$rows = mysql_query ("SELECT purchase_history.customer_id,
                      SUM(total_price),
                      COUNT(purchase_history.customer_id),
                      AVG(total_price),username
                      FROM purchase_history, customers
                      WHERE customers.customer_id=purchase_history.customer_id
                      GROUP BY username
                      ORDER BY SUM(total_price) DESC LIMIT 5;");

while ($row = mysql_fetch_array($rows)) {
    $username=$row ['username'];
    $sum_price=$row ['SUM(total_price)'];
    $avg_price=$row ['AVG(total_price)'];
    $count_purchase=$row ['COUNT(purchase_history.customer_id)'];
    echo "<tr>
            <td>$username</td>
            <td>$count_purchase</td>
            <td>$avg_price kr</td>
            <td>$sum_price kr</td>
        </tr>";
}
echo "</table>";
```

Top 5 of the Best Customers

Customer name	Number of purchases	Average amount per purchase	Total Sum
frank	15	103.3333 kr	1550 kr
casper	4	173.7500 kr	695 kr
johan	1	385.0000 kr	385 kr
charlotte	2	187.5000 kr	375 kr
mads	2	75.0000 kr	150 kr

Final remark/questions?

- NO Sessions/Cookies for the examen
- NO CSS for the examen
- Questions?