introduction to

SCRIPTING, DATABASES, SYSTEM ARCHITECTURE

SQL III: SQL + PHP = Web Services !

Claus Brabrand

(((brabrand@itu.dk)))

Associate Professor, Ph.D.

(((Software and Systems)))

IT University of Copenhagen

Agenda

- Repetition of SQL
- Foreign Keys
- SQL + PHP = Web Services !
- Let's make a Web Service !

ITU Online Course Evaluation

- Please take a moment to fill in the **ITU Online Course Evaluation (!)**

Agenda

- **Repetition of SQL**
- **Foreign Keys**
- **SQL + PHP = Web Services !**
- **Let's make a Web Service !**

SQL: Main Commands

- **Data definition**:
  - CREATE TABLE // creates a new table
  - DROP TABLE // deletes a table
  - SHOW TABLES // see tables defined
  - DESCRIBE // info about a table
- **Data manipulation**:
  - INSERT INTO .. VALUES (..) // insert record(s)
  - **SELECT .. FROM .. WHERE // retrieves info !!!**
  - DELETE FROM .. WHERE // delete record(s)
  - UPDATE .. SET .. WHERE // changes record(s)

Join (1/3)

mailing_list:

name	email
Claus	brabrand@itu.dk
Barack	obama@hotmail.com
John	jdoe@notmail.com

phone_numbers:

email	phone
brabrand@itu.dk	7218 5076
obama@hotmail.com	212-555-0000
jdoe@notmail.com	123-456-7890

```
SELECT *
FROM mailing_list , phone_numbers ;
```

name	email	email	phone
Claus	brabrand@itu.dk	brabrand@itu.dk	7218 5076
Barack	obama@hotmail.com	brabrand@itu.dk	7218 5076
John	jdoe@notmail.com	brabrand@itu.dk	7218 5076
Claus	brabrand@itu.dk	obama@hotmail.com	212-555-0000
Barack	obama@hotmail.com	obama@hotmail.com	212-555-0000
John	jdoe@notmail.com	obama@hotmail.com	212-555-0000
Claus	brabrand@itu.dk	jdoe@notmail.com	123-456-7890
Barack	obama@hotmail.com	jdoe@notmail.com	123-456-7890
John	jdoe@notmail.com	jdoe@notmail.com	123-456-7890

"join" operation:

gives all combos!

Join (2/3)

mailing_list:

name	email
Claus	brabrand@itu.dk
Barack	obama@hotmail.com
John	jdoe@notmail.com

phone_numbers:

email	phone
brabrand@itu.dk	7218 5076
obama@hotmail.com	212-555-0000
jdoe@notmail.com	123-456-7890

```
SELECT *
FROM mailing_list , phone_numbers
WHERE mailing_list.email = phone_numbers.email ;
```

	name	email	email	phone
✓	Claus	brabrand@itu.dk	brabrand@itu.dk	7218 5076
✗	Barack	obama@hotmail.com	brabrand@itu.dk	7218 5076
✗	John	jdoe@notmail.com	brabrand@itu.dk	7218 5076
✗	Claus	brabrand@itu.dk	obama@hotmail.com	212-555-0000
✓	Barack	obama@hotmail.com	obama@hotmail.com	212-555-0000
✗	John	jdoe@notmail.com	obama@hotmail.com	212-555-0000
✗	Claus	brabrand@itu.dk	jdoe@notmail.com	123-456-7890
✗	Barack	obama@hotmail.com		
✓	John	jdoe@notmail.com		

"join" operation:

gives all combos!

now, only the rows we want :-)

name	email	email	phone
Claus	brabrand@itu.dk	brabrand@itu.dk	7218 5076
Barack	obama@hotmail.com	obama@hotmail.com	212-555-0000
John	jdoe@notmail.com	jdoe@notmail.com	123-456-7890

Join (3/3)

mailing_list:

name	email
Claus	brabrand@itu.dk
Barack	obama@hotmail.com
John	jdoe@notmail.com

phone_numbers:

email	phone
brabrand@itu.dk	7218 5076
obama@hotmail.com	212-555-0000
jdoe@notmail.com	123-456-7890

```
SELECT name, mailing_list.email, phone
FROM mailing_list , phone_numbers
WHERE mailing_list.email = phone_numbers.email ;
```

	name ✓	email ✓	email ✗	phone ✓
✓	Claus	brabrand@itu.dk	brabrand@itu.dk	7218 5076
✗	Barack	obama@hotmail.com	brabrand@itu.dk	7218 5076
✗	John	jdoe@notmail.com	brabrand@itu.dk	7218 5076
✗	Claus	brabrand@itu.dk	obama@hotmail.com	212-555-0000
✓	Barack	obama@hotmail.com	obama@hotmail.com	212-555-0000
✗	John	jdoe@notmail.com	obama@hotmail.com	212-555-0000
✗	Claus	brabrand@itu.dk	jdoe@notmail.com	123-456-7890
✗	Barack	obama@hotmail.com	jdo	
✓	John	jdoe@notmail.com	jdo	

name	email	phone
Claus	brabrand@itu.dk	7218 5076
Barack	obama@hotmail.com	212-555-0000
John	jdoe@notmail.com	123-456-7890

"join" operation:

gives all combos!

now, only the rows we want :-)

and only the col's we want :-)

Join (normal, left, right)

a:

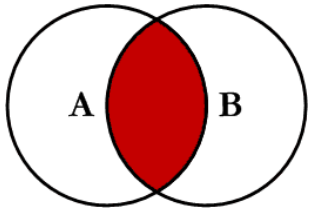
name	id
Anna	1
Brian	2
Claire	3

b:

id	course
1	DSDS
2	GSD
4	IWJX

- Normal join:

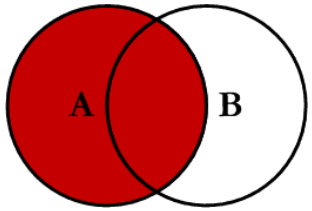
```
SELECT name, a.id, course
  FROM a , b WHERE a.id = b.id ;
```



name	id	course
Anna	1	DSDS
Brian	2	GSD

- Left join:

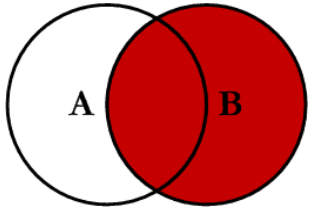
```
SELECT name, a.id, course
  FROM a LEFT JOIN b ON a.id = b.id ;
```



name	id	course
Anna	1	DSDS
Brian	2	GSD
Claire	3	NULL

- Right join:

```
SELECT name, b.id, course
  FROM a RIGHT JOIN b ON a.id = b.id ;
```



name	id	course
Anna	1	DSDS
Brian	2	GSD
NULL	4	IWJX

RIGHT JOIN

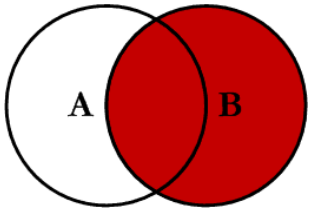
suspects:

name	dna_profile
Jack the Ripper	ACTGC
Jason	ACGTC
Amagermanden	ACTTC

crimescenes:

place	item	dna_found
Living Room	Table	ACCTC
Living Room	Lamp	AGTGC
Hallway	Chainsaw	ACCTC
Bedroom	Knife	ACTGC
Bathroom	Shampoo	ACCTC
Kitchen	Milk Carton	ACTTC

```
SELECT place, item, name
FROM suspects RIGHT JOIN crimescenes
     ON suspects.dna_profile = crimescene.dna_found ;
```



We, of course, have that:

```
x RIGHT JOIN y
      ||
y LEFT JOIN x
```

place	item	name (if match)
Living Room	Table	NULL
Living Room	Lamp	NULL
Hallway	Chainsaw	NULL
Bedroom	Knife	Jack the Ripper
Bathroom	Shampoo	NULL
Kitchen	Milk Carton	Amagermanden

GROUP BY

```
SELECT dept, SUM(amount)
FROM expenses
GROUP BY dept ;
```

expenses:

expense	dept	year	amount
salary	research	2001	490000
salary	sales	2002	1500000
salary	research	2002	500000
coffee	research	2003	800
coffee	sales	2003	300
salary	sales	2003	1600000
salary	research	2003	510000

"GROUP BY" causes the records to be sorted and grouped wrt. their `dept` value:

expense	dept	year	amount
salary	research	2001	490000
salary	research	2002	500000
coffee	research	2003	800
salary	research	2003	510000
salary	sales	2002	1500000
coffee	sales	2003	300
salary	sales	2003	1600000

THEN the result is produced which involves calculating the 'SUM(amount)' values:

dept	SUM(amount)
research	1500800
sales	3100300

GROUP BY (cont'd)

```
SELECT expense, dept, SUM(amount)
FROM expenses
GROUP BY expense, dept ;
```

expenses:

expense	dept	year	amount
salary	research	2001	490000
salary	sales	2002	1500000
salary	research	2002	500000
coffee	research	2003	800
coffee	sales	2003	300
salary	sales	2003	1600000
salary	research	2003	510000

- The sum of expenses grouped by expense ***and*** department:

The table is first grouped by expense:

exp.	dept	year	amount
coffee	research	2003	800
coffee	sales	2003	300
salary	research	2001	490000
salary	sales	2002	1500000
salary	research	2002	500000
salary	sales	2003	1600000
salary	research	2003	510000

THEN, the table is sub-grouped by dept.:

exp.	dept	year	amount
coffee	research	2003	800
coffee	sales	2003	300
salary	research	2001	490000
salary	research	2002	500000
salary	research	2003	510000
salary	sales	2002	1500000
salary	sales	2003	1600000

FINALLY, the result is produced:

exp.	dept	SUM(am)
coffee	research	800
coffee	sales	300
salary	research	1500000
salary	sales	3100000

Agenda

- Repetition of SQL
- Foreign Keys
- SQL + PHP = web services !
- Let's make a Web Service !

Foreign Keys

mailing_list:

name	email
Claus	brabrand@itu.dk
Barack	obama@hotmail.com
John	jdoe@notmail.com

phone_numbers:

email	phone
brabrand@itu.dk	7218 5076
obama@hotmail.com	212-555-0000
jdoe@notmail.com	123-456-7890

- Recall:

```
CREATE TABLE mailing_list (
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE phone_numbers (
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL
);
```

Ensure "referential integrity" :
i.e., that `phone_number.email` is always in `mailing_list.email` !

- We want to **link**:
  - `mailing_list.email` ⟵ `phone_numbers.email`

- We can use a ***foreign key:***

```
CREATE TABLE mailing_list (
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
) ENGINE=InnoDB;
```

```
CREATE TABLE phone_numbers (
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    FOREIGN KEY(email) REFERENCES
        mailing_list(email)
) ENGINE=InnoDB;
```

NB: some (annoying) stuff we have to write b/c we use Foreign Keys [long story...]

Agenda

- Repetition of SQL
- Foreign Keys
- SQL + PHP = web services !
- Let's make Web Services !

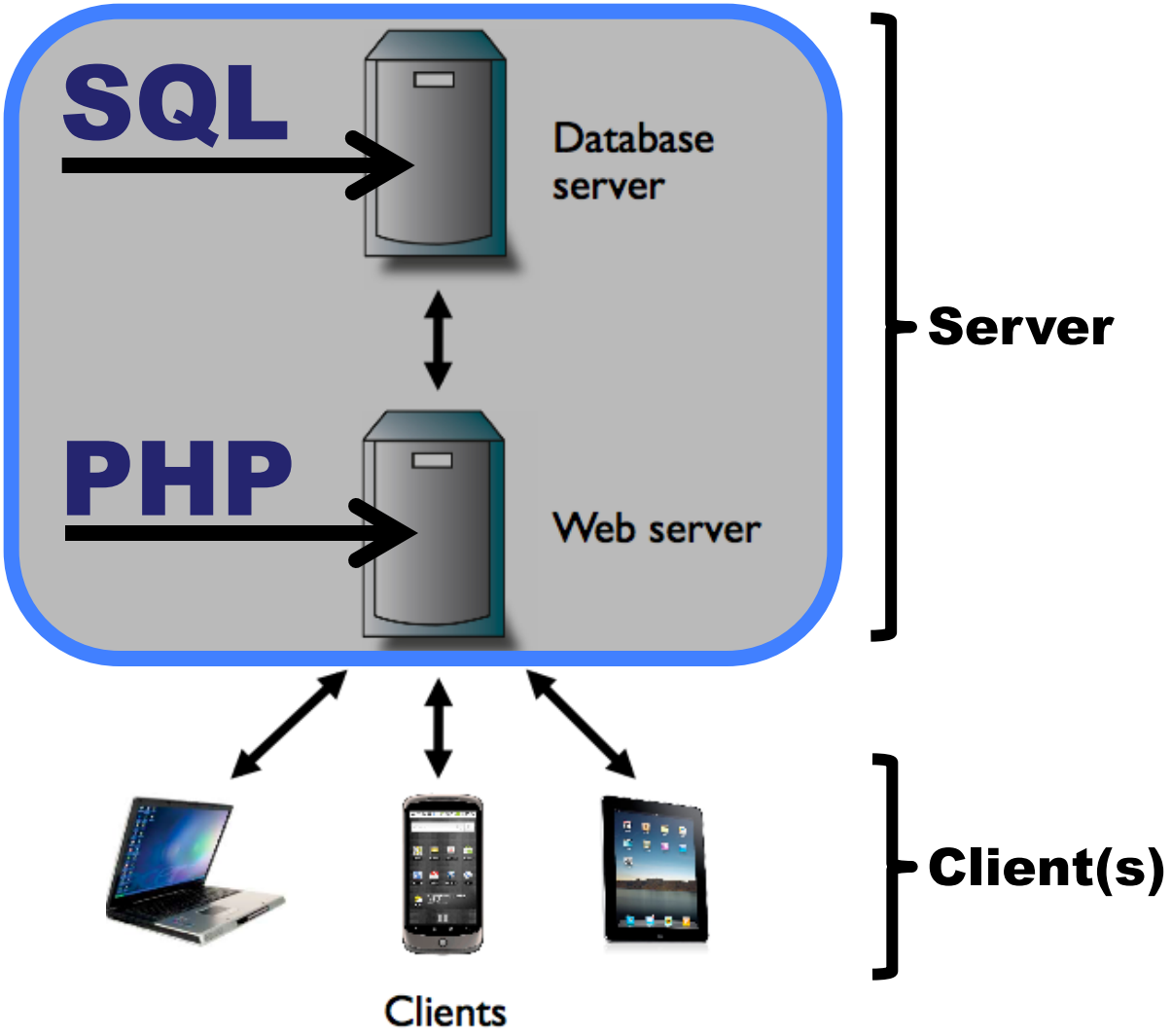
The time has come...

- ...to put it all together:

PHP + SQL !

(= web services)

Web Service = PHP + SQL !



Now, you will learn how to...

- 1) ***connect to*** database from PHP
- 2) ***interact with*** database from PHP
- 3) ***combine PHP+SQL*** to make web services
  (with persistent data in a database)

Main PHP+SQL Functions

- Connecting to database:
  - `mysql_connect()`
  - `mysql_select_db()`
- Querying a database (SELECT, INSERT, ...):
  - `mysql_query()`
- Retrieving (SELECT) query results:
  - `mysql_fetch_array()`
- Closing a database connection:
  - `mysql_close()`

Connecting to Database

- Connecting to DB server:

```
<?php
    mysql_connect("mysql.itu.dk", "claus", "pa$$word");
?>
```

- Choosing Database:

```
<?php
    mysql_select_db("students");
?>
```

- Closing connection:

```
<?php
    mysql_close();
?>
```

Connecting to DB (cont'd)

- How to do it in **PHP+SQL**:

```
<?php
   $conn = mysql_connect("mysql.itu.dk", "claus", "pa$$word") ;
   if (!$conn){
      echo "ERROR: Could not connect to mysql.itu.dk!" ;
      exit ;
   }
   $db = mysql_select_db("students") ;
   if (!$db) {
      echo "ERROR: Could not connect to students database!" ;
      exit ;
   }
   echo "Successfully connected to database" ;
   // ... interact with database ...
   mysql_close() ;
?>
```

Connecting to DB (cont'd)

```php
$conn = mysql_connect("mysql.itu.dk", "claus", "pa$$word");
if (!$conn) {
    echo "ERROR: Could not connect to mysql.itu.dk!" ;
    exit ;
}
$db = mysql_select_db("students");
if (!$db) {
    echo "ERROR: Could not connect to students database!" ;
    exit ;
}
// ... interact with database ...
mysql_close();
```

```php
/* SAME THING, ONLY HOPING THERE WERE NO ERRORS! */
mysql_connect("mysql.itu.dk", "claus", "pa$$word");
mysql_select_db("students"); // uses last connexion established
// ... interact with database ...
mysql_close();
```

Interacting w Database

STUDENTS:

name	address
Anna	Somewhere 3
Brian	Homestreet 4
Claire	Nowhere 9b

- Create table:

```
mysql_query("CREATE TABLE students (
                name VARCHAR(20) NOT NULL,
                address VARCHAR(80) NOT NULL
             );");
```

- Insert data:

```
mysql_query("INSERT INTO students (name, address)
             VALUES ('Anna', 'Somewhere 3');");

mysql_query("INSERT INTO students (name, address)
             VALUES ('Brian', 'Homestreet 4');");

mysql_query("INSERT INTO students (name, address)
             VALUES ('Claire', 'Nowhere 9b');");
```

Interacting w Database

STUDENTS:

name	address
~~Anna~~	~~Anywhere 7~~
Brian	Homestreet 4
Claire	Nowhere 9b

- Update record:

```
mysql_query("UPDATE students
             SET address='Anywhere 7' WHERE name='Anna';");
```

- Delete record:

```
mysql_query("DELETE FROM students WHERE name='Anna';");
```

- Note: you can (of course) also use variables:

```
$name = "Anna" ;
$addr = "Anywhere 7" ;
mysql_query("UPDATE students
             SET address='$addr' WHERE name='$name';");

mysql_query("DELETE FROM students WHERE name='$name';");
// perhaps a bit silly to move her just before deleting her :-)
```

How to SELECT

STUDENTS:

name	address
Anna	Somewhere 3
Brian	Homestreet 4
Claire	Nowhere 9b

```
// SELECT everything:
$rows = mysql_query("SELECT * FROM students;");
// ...returns an array with all the rows; i.e.:
// a row for "Anna", a row for "Brian", and a row for "Claire".

$n = sizeof($rows);
echo "There are $n student(s).";

// iterate through this array of rows, one row at a time...:
while ( $row = mysql_fetch_array($rows) ) {
    // "$row" now is an associate array with the actual data:
    $name = $row['name'] ;
    $addr = $row['address'] ;
    echo "<li><b>$name</b> lives: <em>$addr</em></li>" ;
}

echo "<p>printed the entire table!</p>" ;
```

There are 3 student(s).

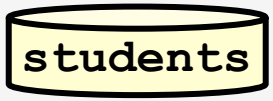
- **Anna** lives: *Somewhere 3*
- **Brian** lives: *Homestreet 4*
- **Claire** lives: *Nowhere 9b*

Printed the entire table!

Complete PHP+SQL Example

```php
<html>
    <body>
        <?php
mysql_connect("mysql.itu.dk", "claus", "pa$$word");
mysql_select_db("students");

mysql_query("DROP TABLE students;");

// CREATE students TABLE:
mysql_query("CREATE TABLE students (
                name VARCHAR(20) NOT NULL,
                address VARCHAR(80) NOT NULL
            );");

// INSERT student records:
mysql_query("INSERT INTO students (name, address) VALUES ('Anna', 'Somewhere 3');");
mysql_query("INSERT INTO students (name, address) VALUES ('Brian', 'Homestreet 4');");
mysql_query("INSERT INTO students (name, address) VALUES ('Claire', 'Nowhere 9b');");

// SELECT everything and print it out nicely:
$rows = mysql_query("SELECT * FROM students;");

while ($row = mysql_fetch_array($rows)) {
    $name = $row['name'] ;
    $addr = $row['address'] ;
    echo "<li><b>$name</b> lives: <em>$addr</em></li>" ;
}

mysql_close();

        ?>
    </body>
</html>
```

mysql.itu.dk:



~~STUDENTS:~~

STUDENTS:

name	address

STUDENTS:

name	address
Anna	Somewhere 3
Brian	Homestreet 4
Claire	Nowhere 9b

- **Anna** lives: *Somewhere 3*
- **Brian** lives: *Homestreet 4*
- **Claire** lives: *Nowhere 9b*

Counting records?

STUDENTS:

name	address
Anna	Somewhere 3
Brian	Homestreet 4
Claire	Nowhere 9b

- Either do the counting in **PHP**:

```
$rows = mysql_query("SELECT * FROM students;");
$n = 0;
while ($row = mysql_fetch_array($rows)) {
    $n++;
}
echo $n ;
```

3

```
$rows = mysql_query("SELECT * FROM students;");
$n = sizeof($rows);
echo $n ;
```

3

- Alternatively, do the counting in **SQL**:

```
$rows = mysql_query("SELECT COUNT(*) as N FROM students;");
$row = mysql_fetch_array($rows); // retrieve the row
$n = $row['N'] ;
echo $n ;
```

N
3

3

Agenda

- Repetition of SQL
- Foreign Keys
- SQL + PHP = web services !
- Let's make a Web Service !

Making Web Services

4-Step design process...:

1) Design data model:

- what info should be stored?
- how should it be represented?

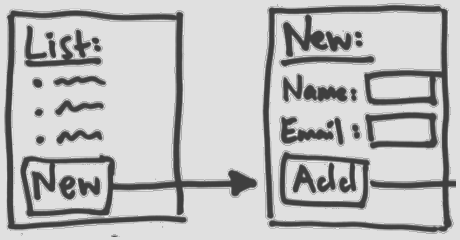
2) Develop data transactions:

- how is data ***read*** (from the DB)?
- how is data ***written*** (to the DB)?
- how is data ***modified*** (in the DB)?

```
SELECT * FROM students
WHERE year = 2012;
```

3) Develop site map and forms:

- use HTML (and CSS) for the user interface (UI)



4) Program it all in SQL + PHP:

- **SQL:** perform DB transactions
- **PHP:** "glue" UI + DB together

```
<?php  int x = 0;
       echo $x;  ?>
```

Making Web Services

4-Step design process...:

1) Design data model:

maillist:

name	email
Claus	brabrand@itu.dk
Barack	obama@hotmail.com
John	jdoe@notmail.com

2) Develop data transactions:

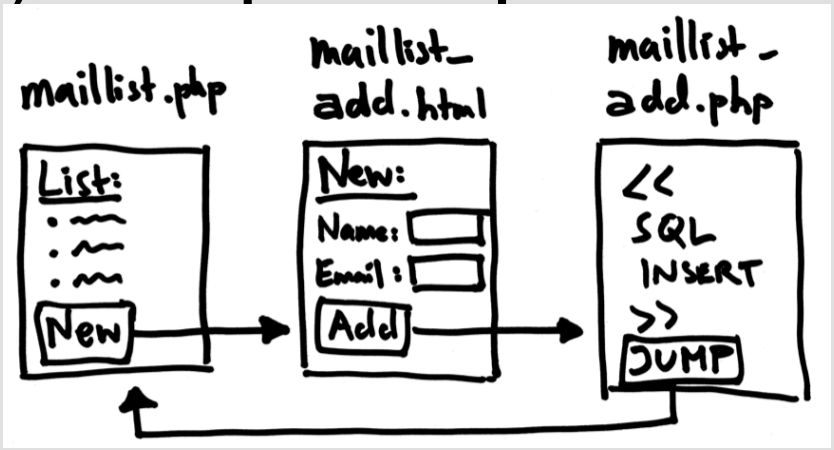
Show list of emails:

```
SELECT * FROM maillist ;
```

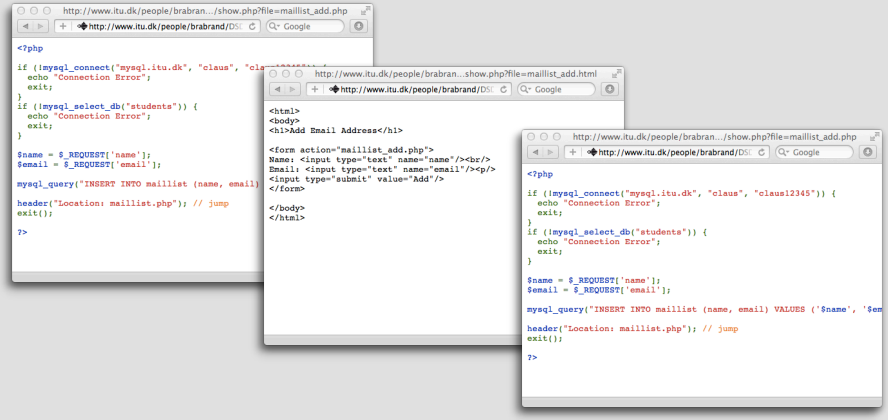
Insert new email:

```
INSERT INTO maillist (name, email)
VALUES ('Claus', 'brabrand@itu.dk');
```

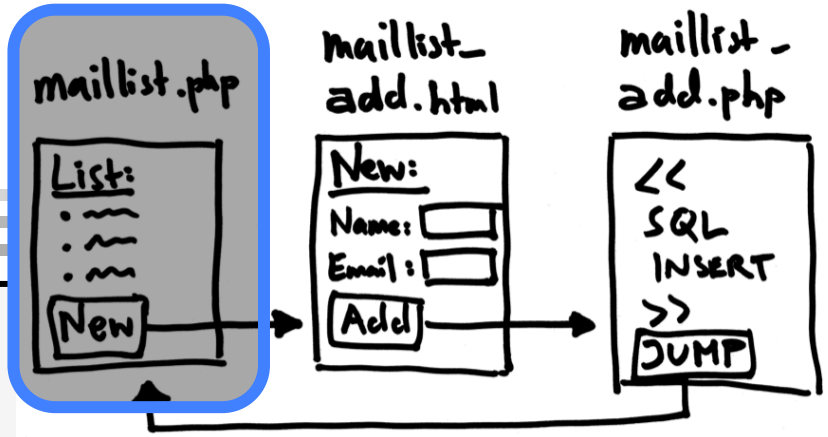
3) Develop site map and forms:



4) Program it all in SQL + PHP:

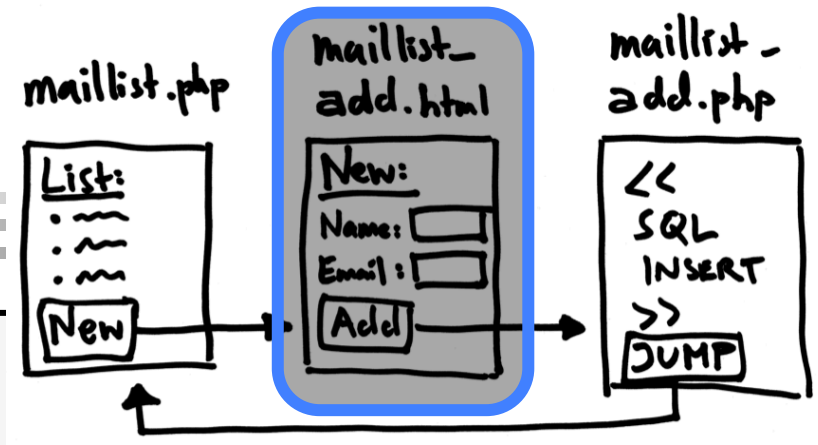


maillist.php



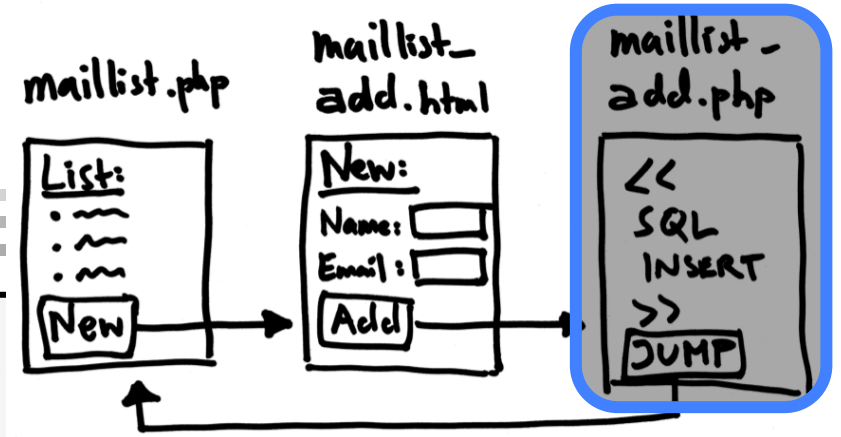

```
<html><body>List:
<ul>
<?php
    include("fn_mydb_connect.php");
    mydb_connect();

    $rows = mysql_query("SELECT * FROM maillist;");
    while ( $row = mysql_fetch_array($rows) ) {
        $name = $row['name'];
        $email = $row['email'];
        echo "<li><a href='mailto:$email'>$name</a></li>" ;
    }
    mysql_close();
?>
</ul>
<a href='maillist_add.html'>NEW</a>
</body></html>
```

maillist_add.html




```
<html>
  <body>
    <h1>Add New Email Address</h1>

    <form action="maillist_add.php">
      Name: <input type="text" name="name"/><br/>
      Email: <input type="text" name="email"/><p/>
      <input type="submit" value="ADD"/>
    </form>
  </body>
</html>
```

maillist_add.php



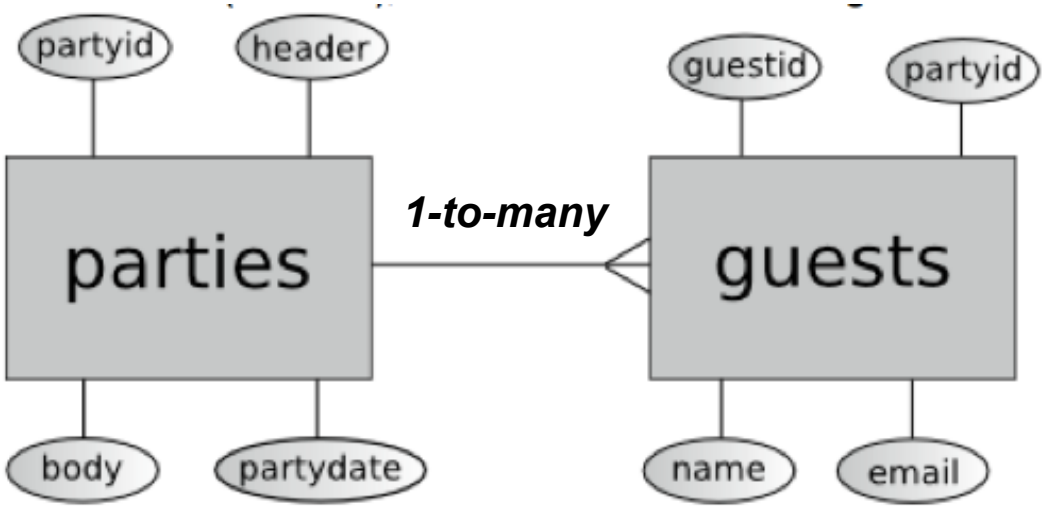

```php
<?php

    include("fn_mydb_connect.php");
    mydb_connect();

    $name = $_REQUEST['name'];
    $email = $_REQUEST['email'];

    mysql_query("INSERT INTO maillist (name, email)
                 VALUES ('$name', '$email');");

    header("Location: maillist.php"); // jump!
    mysql_close();

?>
```

Agenda

- Repetition of SQL
- Foreign Keys
- SQL + PHP = web services !
- Let's make Web Services !
- Assignment + ER Diagrams

ER Diagram

- **ER Diagram:**



Just means that:

- a 'party' consists of: partyid, header, body, partydate
- a 'guest' consists of: guestid, partyid, name, email
- ...and that a party has **many** guests

Any Questions?

(Have a nice weekend)