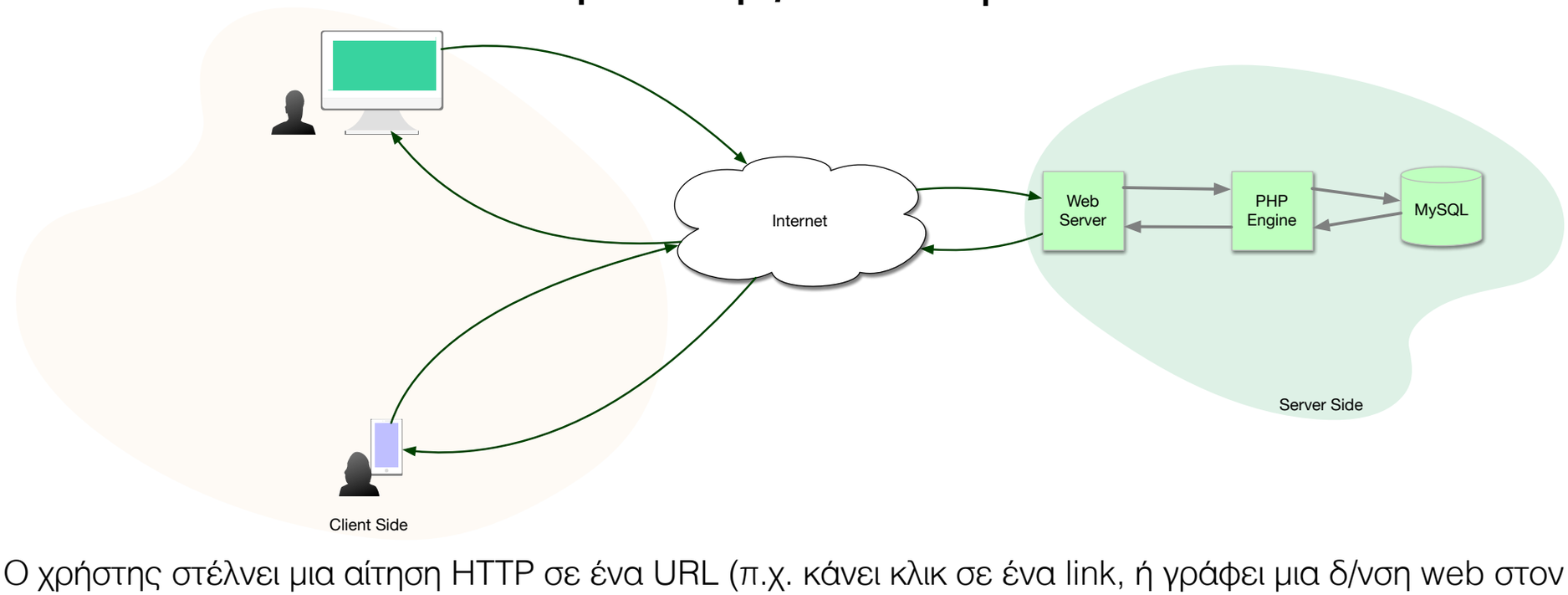
ΤΕΙ Κρήτης
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Θέματα Προγραμματισμού Διαδικτύου
~ MySQL & PHP ~

Στελιος Σφακιανάκης
Εαρινό 2020

Ροή Δεδομένων μεταξύ χρήστη, server, PHP, και βάσης δεδομένων



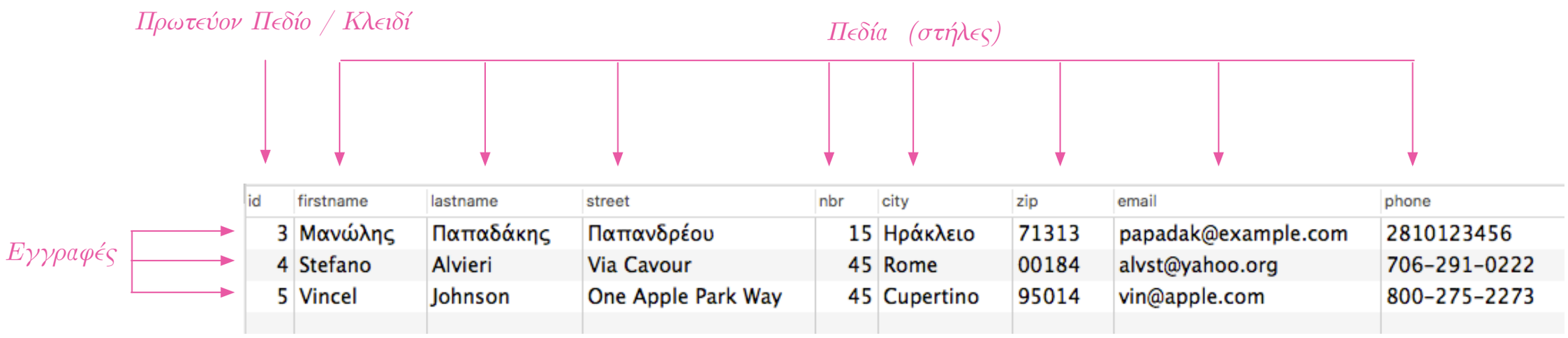
- Ο χρήστης στέλνει μια αίτηση HTTP σε ένα URL (π.χ. κάνει κλικ σε ένα link, ή γράφει μια δ/νση web στον browser του)
- Μέσω των πρωτοκόλλων του διαδίκτυου (DNS, TCP, IP, ...) η αίτηση φτάνει στον web server (π.χ. στον hosting provider σας)
- Ο web server διαπιστώνει ότι το URL που αιτείται αντιστοιχεί σε ένα PHP script (π.χ. bookings.php) , και προωθεί την αίτηση στον PHP interpreter (engine)
- Το PHP εκτελεί το script το οποίο περιέχει εντολές SQL για τη βάση δεδομένων
- Η βάση δεδομένων εκτελεί τις SQL εντολές και επιστρέφει ένα σύνολο αποτελεσμάτων (π.χ. εγγραφές από τον πίνακα κρατήσεων δωματίων του ξενοδοχείου)
- Το script παίρνει τα αποτελέσματα και βάσει αυτών παράγει HTML κώδικα που επιστρέφει στον Web server
- Ο Web Server στέλνει το HTML πίσω στον browser του χρήστη ως απάντηση στην αρχική HTTP αίτηση
- Ο browser εμφανίζει το HTML έγγραφο στο χρήστη.

Βάσεις Δεδομένων

- Ένας έξυπνος τρόπος οργάνωσης και διαχείρισης των δεδομένων
- Υπάρχουν πολλές κατηγορίες:
  - Αντικειμενοστραφείς (Object-oriented)
  - "Εγγραφοστραφείς" (document-oriented), π.χ. XML or JSON-based
  - Βασισμένες σε γράφους (Graph-oriented)
  - ...
  - **Σχεσιακές**
- Οι σχεσιακές βάσεις δεδομένων (ΒΔ) είναι οι πιο δημοφιλείς, καταφέρνοντας να ενσωματώσουν και χαρακτηριστικά από άλλες κατηγορίες (π.χ. JSON storage)
- Από τις σχεσιακές ΒΔ θα ασχοληθούμε με την MySQL που είναι Open Source (αλλά ανήκει στην Oracle πια). Άλλες open source είναι η PostgreSQL και η SQLite, ενώ εμπορικά διαθέσιμες είναι η Oracle, Microsoft SQL Server, και η IBM DB2.

Σχεσιακές Βάσεις Δεδομένων

- Βασική οντότητα: "**Σχέση**" ή αλλιώς .. "πίνακας", δηλ. μια "τετραγωνική" αναπαράσταση όπως ένα φύλλο του Excel
  - Κάθε πίνακας αποτελείται από γραμμές ή "**εγγραφές**"
  - Οι στήλες είναι τα "**πεδία**" του πίνακα. Κάθε πεδίο έχει ένα όνομα και ένα τύπο δεδομένων (π.χ. κείμενο)
  - Κάθε "**κελί**" του πίνακα, δηλ. η τομή μιας γραμμής και μιας στήλης, έχει μία (και μόνο μία σύμφωνα με το σχεσιακό μοντέλο) τιμή
- Οι τιμές ενός ή περισσότερων πεδίων του πίνακα προσδιορίζουν μοναδικά κάθε γραμμή του πίνακα. Το πεδίο (ή τα πεδία αυτά) ονομάζεται Πρωτεύον Κλειδί (**primary key**). Παράδειγμα τέτοιου πεδίου είναι π.χ. το ΑΦΜ ενός ατόμου για τον πίνακα φορολογουμένων της βάσης της Εφορίας, ο αριθμός ταυτότητας ενός πολίτη, το ΑΜΚΑ των ασφαλισμένων, κλπ. Αν δεν υπάρχει ένα τέτοιο πεδίο, μπορούμε να "φτιάξουμε" ένα, π.χ. έναν ακέραιο που αυξάνεται κατά ένα για κάθε εγγραφή που εισάγεται στον πίνακα, όπως φαίνεται στο παρακάτω παράδειγμα:

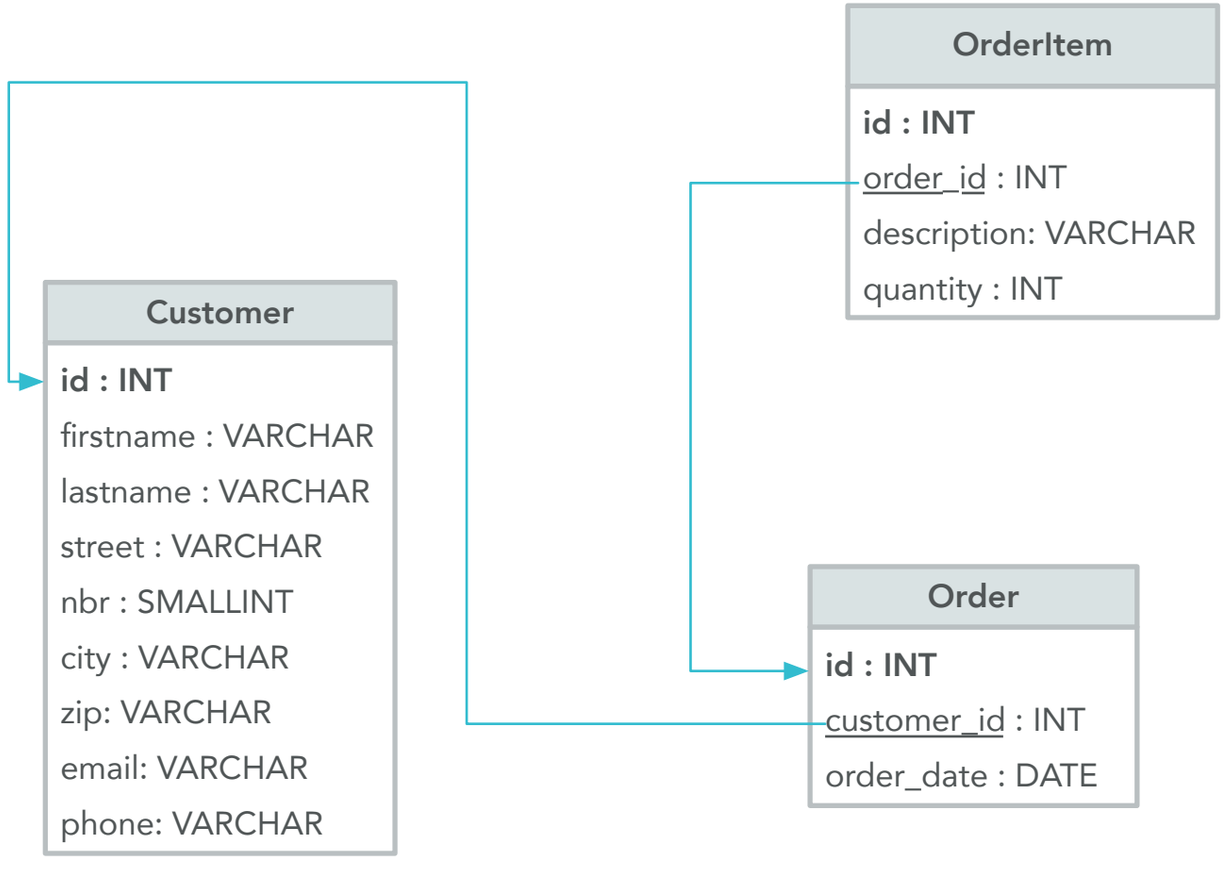


id	firstname	lastname	street	nbr	city	zip	email	phone
3	Μανώλης	Παπαδάκης	Παπανδρέου	15	Ηράκλειο	71313	papadak@example.com	2810123456
4	Stefano	Alvieri	Via Cavour	45	Rome	00184	alvst@yahoo.org	706-291-0222
5	Vincel	Johnson	One Apple Park Way	45	Cupertino	95014	vin@apple.com	800-275-2273

- Για κάθε πίνακα ορίζουμε ποιο είναι το πρωτεύον κλειδί!
  - Έτσι η βάση δεδομένων δεν θα επιτρέψει την εισαγωγή ταυτόσημων εγγραφών
- Επίσης, μας δίνεται η δυνατότητα να *αναφερθούμε* στις εγγραφές του πίνακα από έναν άλλο πίνακα
  - "Ξένα" κλειδιά (Foreign keys) είναι πεδία ενός πίνακα που έχουν τιμές από τα πρωτεύοντα κλειδιά ενός άλλου πίνακα
  - Έτσι επιτρέπουμε τις αναφορές ("συσχετίσεις") μεταξύ των πινάκων

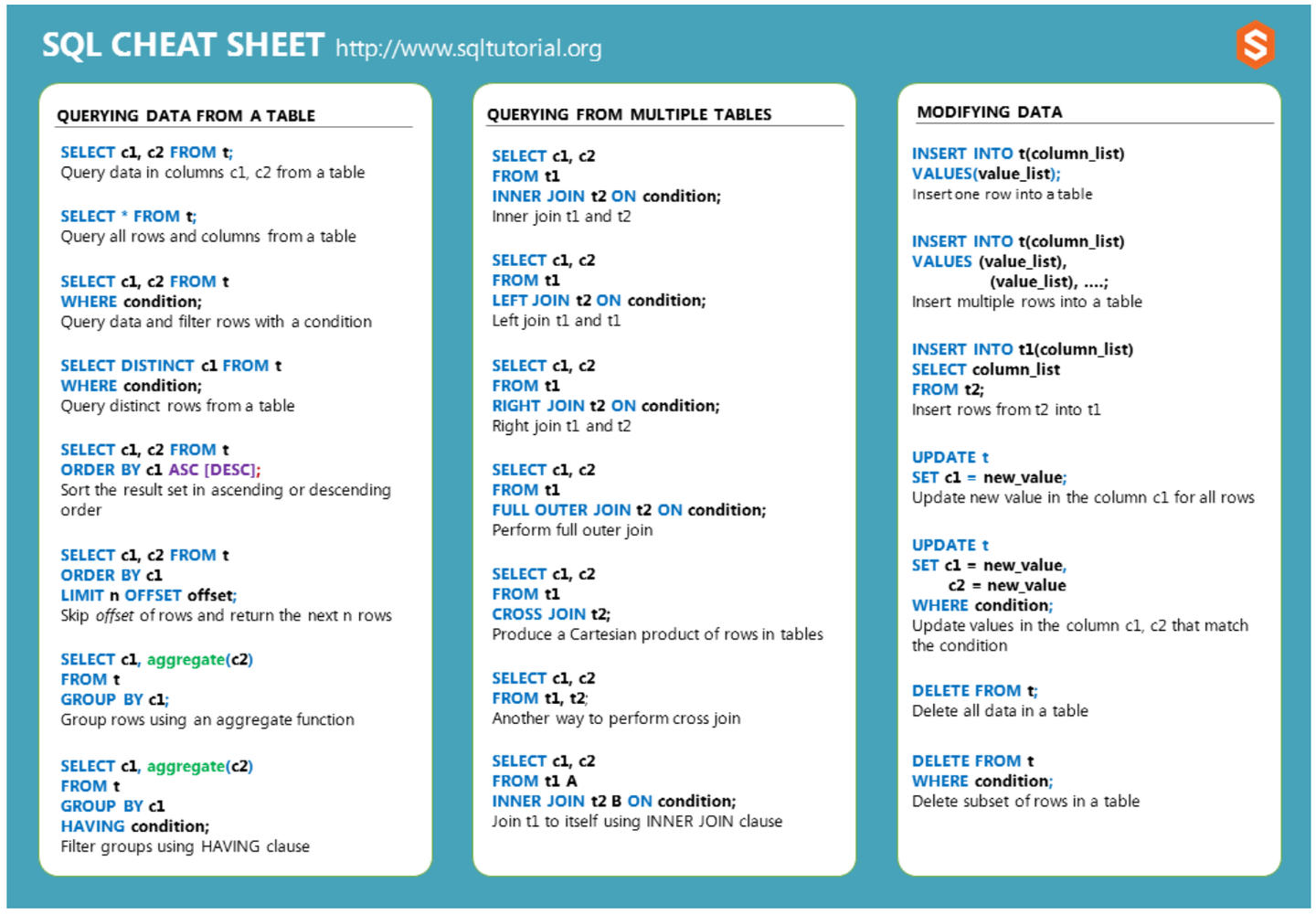
Παράδειγμα

- Ο πίνακας **Customer** έχει τα στοιχεία των πελατών ενός καταστήματος
    - Για κάθε πελάτη κρατάμε το όνομα, επίθετο, δ/νση, κλπ
    - Το **id** είναι το πρωτεύον κλειδί (ακέραιος με αυτόματη αύξηση ("auto-increment"))
- Ο πίνακας **Order** περιέχει τις παραγγελίες.
    - Κάθε παραγγελία έχει ένα **id** (πρωτεύον κλειδί, ακέραιος) και την ημερ/νία που έγινε (**order_date**)
    - Το **customer_id** είναι το **id** του customer που έκανε την παραγγελία (**foreign key**)
- Ο πίνακας **OrderItem** έχει τα περιεχόμενα της παραγγελίας δηλ. ποιο προϊόν (περιγραφή, **description**) και την ποσότητα (**quantity**), το id της παραγγελίας (**order_id**, foreign key), και τον μοναδικό **id** του συγκεκριμένου προϊόντος στην παραγγελία



SQL

- Structured Query Language, η standard γλώσσα υποβολής ερωτήσεων σε σχεσιακές βάσεις δεδομένων
- 3 είναι οι πιο συνηθισμένες εντολές:
  - `SELECT <field>,..., <field> FROM <table> WHERE ...`
  - `INSERT INTO <table>(<field>,..., <field>) VALUES (<val>,...,<val>)`
  - `UPDATE <table> SET <field>=<val> WHERE ...`



SQL CHEAT SHEET http://www.sqltutorial.org

QUERYING DATA FROM A TABLE

SELECT c1, c2 FROM t;
Query data in columns c1, c2 from a table

SELECT * FROM t;
Query all rows and columns from a table

SELECT c1, c2 FROM t
WHERE condition;
Query data and filter rows with a condition

SELECT DISTINCT c1 FROM t
WHERE condition;
Query distinct rows from a table

SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC];
Sort the result set in ascending or descending order

SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset;
Skip *offset* of rows and return the next n rows

SELECT c1, aggregate(c2)
FROM t
GROUP BY c1;
Group rows using an aggregate function

SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition;
Filter groups using HAVING clause

QUERYING FROM MULTIPLE TABLES

SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition;
Inner join t1 and t2

SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition;
Left join t1 and t1

SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition;
Right join t1 and t2

SELECT c1, c2
FROM t1
FULL OUTER JOIN t2 ON condition;
Perform full outer join

SELECT c1, c2
FROM t1
CROSS JOIN t2;
Produce a Cartesian product of rows in tables

SELECT c1, c2
FROM t1, t2;
Another way to perform cross join

SELECT c1, c2
FROM t1 A
INNER JOIN t2 B ON condition;
Join t1 to itself using INNER JOIN clause

MODIFYING DATA

INSERT INTO t(column_list)
VALUES(value_list);
Insert one row into a table

INSERT INTO t(column_list)
VALUES (value_list),
(value_list), ...;
Insert multiple rows into a table

INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
Insert rows from t2 into t1

UPDATE t
SET c1 = new_value;
Update new value in the column c1 for all rows

UPDATE t
SET c1 = new_value,
c2 = new_value
WHERE condition;
Update values in the column c1, c2 that match the condition

DELETE FROM t;
Delete all data in a table

DELETE FROM t
WHERE condition;
Delete subset of rows in a table

Να διαβάσετε τις 17 πρώτες ενότητες (μέχρι SQL Between) από https://www.w3schools.com/sql/

MySql & PHP

mysqli

- Θα χρησιμοποιήσουμε το `mysqli` που είναι ο "βελτιωμένος" τρόπος επικοινωνίας της PHP με MySQL (το "i" είναι από το "improved" 😀)
  - Πρόκειται για extension ("library") της PHP που (συνήθως) εγκαθίσταται μαζί με το PHP
  - Τεκμηρίωση: https://www.php.net/manual/en/book.mysqli.php
- Υπάρχουν 2 "εκδόσεις" για τις εντολές του mysqli:
  - Ο "διαδικαστικός" (procedural), π.χ. γράφουμε `$connection = mysql_connect(...)`
  - Ο αντικειμενοστραφής (object-oriented), π.χ. `$connection = new mysqli(...)`
- Στη συνέχεια εμείς θα χρησιμοποιήσουμε τον αντικειμενοστραφή τρόπο (ως πιο "μοντέρνο")

Ανάκτηση εγγραφών

- `new mysqli (<host>, <user>, <password>, <dbname>)` : δημιουργεί connection object

- `connect_errno` & `connect_error` : έχουν τιμές σε περίπτωση που αποτύχει η σύνδεση με τη βάση

- `$con->query(...)` : εκτέλεση μιας SQL εντολής, επιστρέφει ένα result object ή FALSE αν αποτύχει

- `$result->num_rows()`: επιστρέφει αριθμό αποτελεσμάτων από ένα SELECT query

- `$result->fetch_assoc()`: επιστρέφει την επόμενη γραμμή (αποτέλεσμα) σε ένα array με τα ονόματα των πεδίων

```php
<?php

$con = new mysqli("127.0.0.1", "root", "my-secret-pw", "myapp");
if ($con->connect_errno) {
    die("Connection error: " . $con->connect_error);
}

$result = $con->query("SELECT * FROM users");

if ($result === false) {
    die("Query failed");
}

echo "Found {$result->num_rows} rows";

while($row = $result->fetch_assoc()) {
    echo $row['name'];
}

?>
```

Εισαγωγή Εγγραφών : ο λάθος τρόπος!

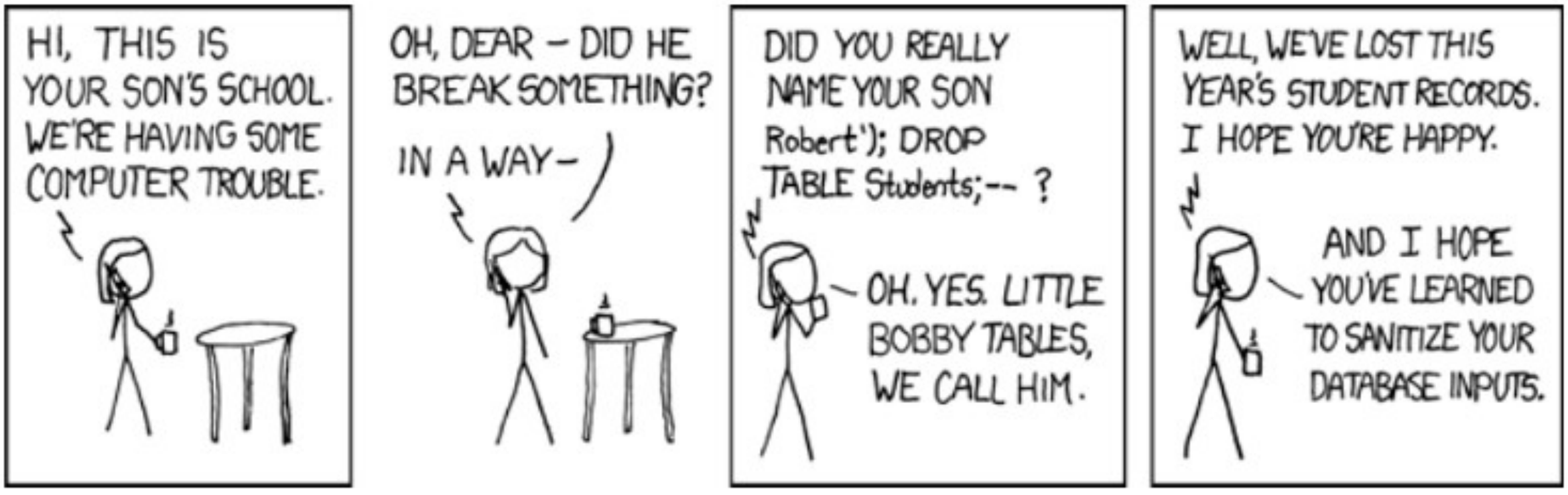
```php
<?php

$con = new mysqli("127.0.0.1", "root", "my-secret-pw", "myapp");
if ($con->connect_errno) {
    die("Connection error: " . $con->connect_error);
}

$name = "Stelios";
$email = "sasfak@gmail.com";
$result = $con->query("INSERT INTO users(name, email) VALUES('{$name}', '{$email}')");

if ($result === false) {
    die("Query failed : {$con->error}");
}

echo "Inserted {$con->affected_rows} rows";
?>
```

Μην το κάνετε αυτό!

- Μπορούμε να ενσωματώσουμε "χειροκίνητα" τα δεδομένα που θέλουμε να εισάγουμε σε ένα INSERT, **αλλά αυτό έχει προβλήματα ασφάλειας** (δείτε επόμενο slide)

- `$con->affected_rows` : ο αριθμός των γραμμών (εγγραφών) που άλλαξαν (αν κάναμε UPDATE), σβήστηκαν (DELETE), ή εισάχθηκαν (INSERT)

Little Bobby Tables



https://www.explainxkcd.com/wiki/index.php/Little_Bobby_Tables

SQL Injection

- Το πρόβλημα αυτό υπάρχει είτε κάνουμε INSERT ή SELECT ή UPDATE
  - Γενικά οποτεδήποτε θέλουμε να εκτελέσουμε μια εντολή SQL με δεδομένα που μας έδωσε ο χρήστης (ή ένα άλλο σύστημα που δεν εμπιστευόμαστε)
  - Το επίσημο όνομα του είναι SQL Injection
- Μια εύκολη (σχετικά) λύση είναι η χρήση *Prepared Statements*

MySQL "προετοιμασμένες δηλώσεις" (Prepared Statements)

- Βασική ιδέα:

  (1) Δηλώνουμε ("ετοιμάζουμε") τις εντολές SQL και βάζουμε ? στις θέσεις που πρέπει να μπουν οι τιμές, π.χ.

```
SELECT * FROM users WHERE name = ? and age > ?
```

  (2) Προσδιορίζουμε τις μεταβλητές οι τιμές των οποίων θα αντικαταστήσουν τα ? μαζί με τους τύπους δεδομένων τους

  (3) Εκτελούμε την εντολή

Εισαγωγή Εγγραφών : ο σωστός τρόπος!

- `$con->prepare("...")`: δέχεται μια SQL εντολή με `?` και επιστρέφει ένα statement object

- `$stmt->bind_param(...)`: κανει bind με τη σειρά τα ? με τις μεταβλητές που δίνονται. Το 1ο όρισμα προσδιορίζει τι τύπου είναι κάθε μεταβλητή (`s` : string, `i`: integer, `d`: double)

- `$stmt->execute()` : εκτελεί την εντολή και επιστρέφει το αποτέλεσμα

```php
<?php

$con = new mysqli("127.0.0.1", "root", "my-secret-pw", "myapp");
if ($con->connect_errno) {
    die("Connection error: " . $con->connect_error);
}

$name = "Stelios";
$email = "sgsfak@gmail.com";

$stmt = $con->prepare("INSERT INTO users(name, email) VALUES(?, ?)");

$stmt->bind_param("ss", $name, $email);

$result = $stmt->execute();

if ($result === false) {
    die("Query failed : {$con->error}");
}

echo "Inserted {$con->affected_rows} rows";
?>
```

Παράδειγμα SELECT prepared statement

Δημιουργία prepared statement

Binding variables στα ?

Εκτέλεση prepared statement

Ανάκτηση αποτελεσμάτων

Αριθμός αποτελεσμάτων

Διάτρεξη αποτελεσμάτων

```php
<?php
$con = new mysqli("127.0.0.1", "root", "my-secret-pw", "myapp");
if ($con->connect_errno) {
    die("Connection error: " . $con->connect_error);
}

$email = "sgsfak@gmail.com";
$stmt = $con->prepare("SELECT * FROM users WHERE email = ?");

if ($stmt === false) {
    die("Prepared query creation failed : {$con->error}");
}

$stmt->bind_param("s", $email);
$result = $stmt->execute();

if ($result === false) {
    die("Query failed : {$con->error}");
}

$result = $stmt->get_result();

echo "Found {$result->num_rows}";

while($row = $result->fetch_assoc()) {
    echo $row['name'];
}
?>
```

INSERTS και auto-increment ids

```php
$name = "Stelios";
$email = "sgsfak@gmail.com";

$stmt = $con->prepare("INSERT INTO users (name, email) VALUES (?,?)");

if ($stmt === false) {
    die("Prepared query creation failed : {$con->error}");
}
$stmt->bind_param("ss", $name, $email);
$result = $stmt->execute();
if ($result === false) {
    die("Query failed : {$con->error}");
}
$id = $con->insert_id;
echo "Row inserted with id = $id";
```

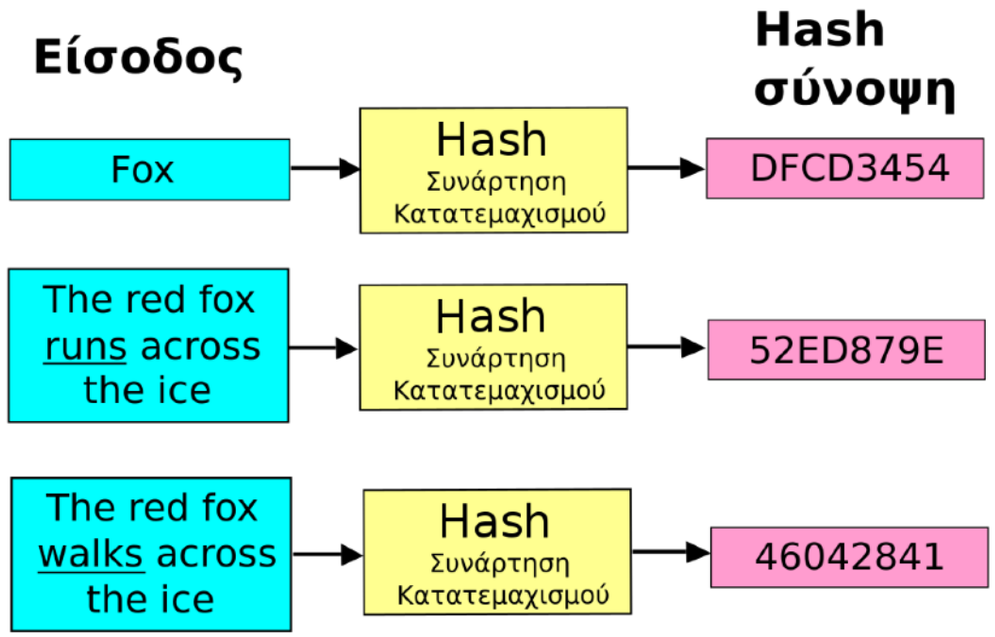
- Σε περιπτώσεις που δεν υπάρχει κάποιο (ή κάποια) από τα πεδία να έχει μοναδικές τιμές και να παίξει το ρόλο του πρωτεύοντος κλειδιού, συνηθίζεται να προσθέτουμε ένα "τεχνητό" πεδίο τύπου Integer με όνομα (συνήθως) `id` το οποίο ορίζεται ως "auto-increment" (δηλ. αυξάνεται κατά 1 αυτόματα με κάθε εισαγωγή εγγραφής)

- `$con->insert_id` : επιστρέφει το id που δημιουργήθηκε αυτόματα από το πιο πρόσφατο INSERT

Αποθηκεύοντας Passwords

- Ποτέ μα ποτέ δεν αποθηκεύουμε κωδικούς σε "clear text" (δηλ. σε μη κρυπτογραφημένη μορφή)
  - Δεν είναι κίνδυνος μόνο για το δικό σας website, σκεφτείτε ότι αρκετοί χρήστες χρησιμοποιούν τον ίδιο κωδικό σε πολλά διαφορετικά sites..
- Η "βέλτιστη πρακτική" είναι να αποθηκεύουμε το *password hash* δηλ. έναν μετασχηματισμό του κωδικού που προκύπτει από μια ασφαλής κρυπτογραφική one-way συνάρτηση (π.χ. bcrypt)

Cryptographic Hash Function



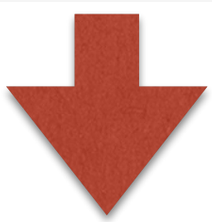
Η **κρυπτογραφική συνάρτηση κατακερματισμού** (cryptographic hash function) είναι μια συνάρτηση κατατεμαχισμού (hash function) ή οποία είναι σχεδιασμένη για να χρησιμοποιείται στην κρυπτογραφία. Γενικά η συνάρτηση κατατεμαχισμού είναι μια μαθηματική συνάρτηση που έχοντας ως είσοδο μια αυθαίρετου μεγέθους ομάδα δεδομένων δίνει έξοδο μια καθορισμένου μεγέθους στοιχειοσειρά (string) (η συμβολοσειρά είναι συνήθως μικρότερη σε μέγεθος από την αρχική είσοδο). Η έξοδος δεν μπορεί με αντιστροφή (με κανένα τρόπο) να μας παράγει την αρχική είσοδο. Η έξοδος αποκαλείται συνήθως "σύνοψη" (digest).

https://el.wikipedia.org/wiki/Κρυπτογραφική_Συνάρτηση_Κατατεμαχισμού

password_hash

- `password_hash("...", PASSWORD_DEFAULT)`: Υπολογίζει τη "σύνοψη" (hash digest) του string που δόθηκε βασει του default αλγορίθμου

```php
$name = "Stelios";
$email = "sgsfak@gmail.com";
$password = "stelios1234"; // My real password

$stmt = $con->prepare("INSERT INTO users (name, email, pwd) VALUES (?,?,?)");

if ($stmt === false) {
    die("Prepared query creation failed : {$con->error}");
}
// Compute password hash:
$hash = password_hash($password, PASSWORD_DEFAULT);
$stmt->bind_param("sss", $name, $email, $hash);

$result = $stmt->execute();
```



name	email	pwd	id
Stelios	sgsfak@gmail.com	$2y$10$T.Biy75wXVoQycqhKAxZ6.4iK00eSjR2dRGXV7HftgVuikKHOxHWe	9

password_verify

- Όταν κάνει login ένας χρήστης πρέπει να επιβεβαιώσουμε το username και το password του.

- Δεν μπορούμε από το αποθηκευμένο hash να βρούμε το password, αλλά μπορούμε να υπολογίσουμε το hash του password που μας δίνει τώρα και να το συγκρίνουμε με το αποθηκευμένο hash!

- Το `password_verify(<pass>, <hash>)` κάνει αυτό ακριβώς: βεβαιώνει ότι αν το 1ο όρισμα γίνει hash-ed τότε είναι το ίδιο με το 2ο όρισμα.

- Το διπλανό παράδειγμα δείχνει όλη τη διαδικασία (όπου ως username χρησιμοποιώ το email)

```php
// Login for user with given email and password:
$email = "sgsfak@gmail.com";
$password = "stelios1234";

// Get the password hash from the Database:
$stmt = $con->prepare("SELECT pwd FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows == 0) {
    echo "Wrong email, we don't know you...";
}
else {
    // We found a user with the given email
    // Now verify his/her password:
    $row = $result->fetch_assoc();
    $hash = $row["pwd"];

    $ok = password_verify($password, $hash);
    if ($ok) {
        echo "OK, you are now logged in!!";
    }
    else {
        echo "Wrong password!! Go away...";
    }
}
```