

DATABASE SYSTEMS




Database programming in a web environment



Database System Course, 2016

AGENDA FOR TODAY

Advanced Mysql

-  More than just SELECT
-  Creating tables
-  MySQL optimizations: Storage engines , indexing.

Database programming

-  HelloWorld: Python + MySQL

Recap: DB servers in the web

Web programming architecture

HTTP on a need-to-know basis.

Web APIs: REST ,json, and how to get them

The final project

ADAVANCED MYSQL



More than just SELECT

- CREATE

```
1 CREATE TABLE students
2 (
3   StudentID int,
4   LastName varchar(20),
5   FirstName varchar(20),
6   Image blob
7 );
8
```

- INSERT

```
1 INSERT INTO Students (StudentID, FirstName, LastName, Image)
2 VALUES (309112442, "Robert", "Smith", LOAD_FILE('/~/LittleBobby.png'));
3
```

- UPDATE

```
1 UPDATE Students
2 SET LastName='Tables'
3 WHERE CustomerName='Robert';
4
```

ADAVANCED MYSQL



More than just SELECT

- ALTER

```
1 ALTER TABLE Students
2 ADD DateOfBirth Date;
3
```

```
1 ALTER TABLE Students
2 DROP COLUMN Image;
3 |
```

- DELETE

```
1 DELETE FROM Studnets
2 WHERE FirstName='Robert';
3 |
```

- DROP

```
1 DROP TABLE Students
2 ;
```

ADAVANCED MYSQL



Creating tables:

- **Constraints:**

- NOT NULL - Indicates that a column cannot store NULL value
- UNIQUE - Ensures that each row for a column must have a unique value
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
- FOREIGN KEY - Ensure the referential integrity of the data in one table to match values in another table
- CHECK - Ensures that the value in a column meets a specific condition
- DEFAULT - Specifies a default value for a column

ADAVANCED MYSQL



Creating tables

- Constraints:

```
1 CREATE TABLE Studnets
2 (
3   StudentID int NOT NULL AUTO_INCREMENT,
4   FirstName varchar(20) NOT NULL,
5   LastName varchar(20) NOT NULL,
6   Image blob,
7   PRIMARY KEY (StudentID),
8   CHECK (StudentID>0)
9 );
```

```
1 CREATE TABLE TeacherAssistants
2 (
3   TeacherID int NOT NULL,
4   TeachingSubject varchar NOT NULL,
5   StudentID int,
6   PRIMARY KEY (TeacherID),
7   FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
8 );
9
```

ADAVANCED MYSQL



Triggers:

- A database trigger is procedural code that is automatically executed in response to certain events on a particular table

- Supported Events:

BEFORE INSERT

AFTER INSERT

BEFORE UPDATE

AFTER UPDATE

BEFORE DELETE

AFTER DELETE

ADAVANCED MYSQL



Triggers:

- Use NEW to refer to a row

Triggers

```
1  -- Trigger DDL Statements
2  DELIMITER $$
3
4  • USE `university` $$
5
6  • CREATE TRIGGER t_grade_check
7    BEFORE INSERT ON takes
8    FOR EACH ROW
9    BEGIN
10     IF NEW.grade < 0 THEN
11         SET NEW.grade = 0;
12     ELSEIF NEW.grade > 100 THEN
13         SET NEW.grade = 100;
14     END IF;
15 END; $$
```


ADAVANCED MYSQL



MySQL Optimizations

- **Storage Engines (Database Engine):**
 - The underlying software performing CRUD operations: Create, Read, Update, Delete.
 - The storage engine implements the data structures and memory usage strategy
 - Many of the modern DBMS support multiple storage engines within the same database
 - MySQL support InnoDB and MyISAM

ADAVANCED MYSQL



MySQL Optimizations (Storage Engines)

- InnoDB:
 - The default general-purpose MySQL storage engine
 - ACID Compliant:
 - **A**tomicity: A transaction (i.e., set of DB operations) is atomic
 - **C**onsistency: Any given database transaction must change affected data only in allowed ways (Triggers, Constraints)
 - **I**solation: Concurrent transactions are isolated from one to another
 - **D**urability: The ability of the system to recover committed transaction updates if either the system or the storage media fails
- Main Features:
 - ♦ Takes care of data integrity
 - ♦ Row-level locking

ADAVANCED MYSQL



MySQL Optimizations (Storage Engines)

- MyISAM (Indexed Sequential Access Method)
 - **Storage paradigm:**
 - ◆ Each entry points to a record in the data file, and the pointer is offset from the beginning of the file
 - ◆ This way records can be quickly read, especially when the format is FIXED
 - ◆ Inserts are easy too, because new rows are appended to the end of the data file
 - ◆ However, delete and update operations are more problematic: deletes must leave an empty space, or the rows' offsets would change; the same goes for updates, as the length of the rows becomes shorter;
 - **Main features:**
 - ◆ Non Transactional (Does not support foreign keys)
 - ◆ Fits for Read Mostly environments (because of the table level locking mechanism)

ADAVANCED MYSQL

MySQL Optimizations (Storage Engines)

- Other storage engines:
 - ◆ **Memory:** Stores all data in RAM, for fast access in environments that require quick lookups of non-critical data.
 - ◆ **Archive:** These compact, unindexed tables are intended for storing and retrieving large amounts of seldom-referenced historical, archived, or security audit information.
 - ◆ **NDB** (also known as NDBCLUSTER): This clustered database engine is particularly suited for applications that require the highest possible degree of uptime and availability.
 - ◆ **Merge:** Enables a MySQL DBA or developer to logically group a series of identical MyISAM tables and reference them as one object. Good for VLDB environments such as data warehousing.
 - ◆ **Federated:** Offers the ability to link separate MySQL servers to create one logical database from many physical servers. Very good for distributed environments.

ADAVANCED MYSQL



MySQL Optimizations: Indexing

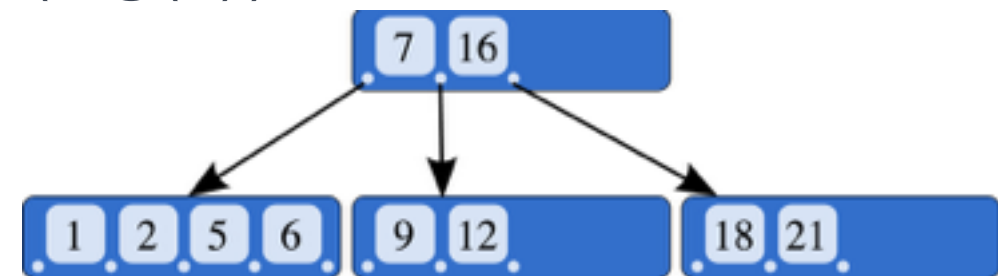
- If you don't use indexes:
 - ◆ Your DB is small (or)
 - ◆ Your DB is slow
- Indexes are used to find rows with specific column values quickly
- Can be single or multi-column.
- Can use only part of the data:
- **Examples:**
 - `CREATE INDEX last ON Students (LastName)`
 - `CREATE INDEX full_name ON Students (FirstName, LastName)`
 - `CREATE INDEX part_of_name ON Students (LastName(5));`

ADAVANCED MYSQL



MySQL Optimizations: Indexing

- Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows
- Updates cost more...
- Storing indexes:
 - **B-tree** (Search, insert and delete are $O(\log(n))$)
 - R-tree (Spatial Data)
 - Hash tables
 - Inverted lists (mapping words/numbers to DB entries)



DB DESIGN:TIPS AND TRICKS

 **Schema Design:** You will have/already had a dedicated class on DB design principles don't worry

1. **Use primary keys:**

- They have special indexes in InnoDB for fast lookups
- If your table is big and important, but does not have an obvious column or set of columns to use as a primary key:
 - Create a separate column with auto-increment values to use as the primary key.
 - These unique IDs can serve as pointers to corresponding rows in other tables when you join tables using foreign keys.

2. **Use foreign keys:**

- Mostly for data integrity
- For optimisation: Large tables Vs. Many small tables
 - Consider splitting your less-frequently used data into separate tables
 - Each small table can have a primary key for fast lookups of its data, and you can query just the set of columns that you need using a join operation.
 - Queries might perform less I/O and take up less cache memory because the relevant columns are packed together on disk.

DB DESIGN:TIPS AND TRICKS



Schema Design: You will have/already had a dedicated class on DB design principles don't worry

3. **Use indexes *when appropriate*:**

- They take more storage and update costs more
- Multi column Vs. Single column: It depends on the query ('Or' vs.'And')
- For full text search use a reverse index.
- Rebuild indexes after your DB is stable.

4. **Choose your storage engine:**

- Or not. Now InnoDB is quite a standard....

5. **Use correct data types:**

- Smallest as possible to minimize disk space












6. **Use “NOT NULL” as often as possible**

- Enabling better use of indexes and eliminating overhead for testing whether each value is NULL

7. **Normalisation?** (Avoiding redundant data by using unique IDs)

- To save disk space, do it. For fast retrieval: Don't.

AGENDA FOR TODAY

-  Advanced Mysql
 -  More than just SELECT
 -  Creating tables
 -  MySQL optimizations: Storage engines , indexing.
-  **Database programming**
 -  HelloWorld: Python + MySQL
-  **Recap: DB servers in the web**
-  **Web programming architecture**
-  **HTTP on a need-to-know basis.**
-  **Web APIs: REST ,json, and how to get them**
-  **The final project**

DB PROGRAMMING

HELLOWORLD



Using a `mysqDB` (python 2.7x) or `mysqlclient` (python 3.x)

- Install `mysqlDB` or `mysqlclient` via PIP

```
→ ~ sudo pip install Mysql-python
The directory '/Users/amitso/Library/Caches/pip/http' or its parent directory is not owned by the current
user and the cache has been disabled. Please check the permissions and owner of that directory. If executi
ng pip with sudo, you may want sudo's -H flag.
The directory '/Users/amitso/Library/Caches/pip' or its parent directory is not owned by the current user
and caching wheels has been disabled. check the permissions and owner of that directory. If executing pip
with sudo, you may want sudo's -H flag.
Collecting Mysql-python
  Downloading MySQL-python-1.2.5.zip (108kB)
    100% |#####| 112kB 548kB/s
Installing collected packages: Mysql-python
  Running setup.py install for Mysql-python ... done
Successfully installed Mysql-python-1.2.5
→ ~
```

DB PROGRAMMING

HELLOWORLD

 Using a **mysqDB** (python 2.7x) or **mysqlclient** (python 3.x)

- In your Python script:
 1. Import MySQLdb
 2. Create a connector to the DB with: server name, user, password , DB name

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb');
```

DB PROGRAMMING

HELLOWORLD

 Using a `mysqDB` (python 2.7x) or `mysqlclient` (python 3.x)

- In your Python script:
 1. Create a **cursor** (`cur = con.cursor()`)
 2. **Execute** a query (`cur.execute("<YOURSQL_QUERY>")`)
 3. **Fetch** the rows in the results (`rows=cur.fetchall()`)

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb')

with con:

    cur = con.cursor(mdb.cursors.DictCursor)
    cur.execute("SELECT * FROM Writers LIMIT 4")

    rows = cur.fetchall()
```

DB PROGRAMMING

HELLOWORLD

 Using a `mysqDB` (python 2.7x) or `mysqlclient` (python 3.x)

- In your Python script:
 1. Working the results:
 1. Reference by position (`row[0], row[1]`)
 2. *Reference by column name (`row["Id"], row["Name"]`)*

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb')

with con:

    cur = con.cursor(mdb.cursors.DictCursor)
    cur.execute("SELECT * FROM Writers LIMIT 4")

    rows = cur.fetchall()

    for row in rows:
        print row["Id"], row["Name"]
```

DB PROGRAMMING

HELLOWORLD

 Using a mysqDB (python 2.7x) or mysqlclient (python 3.x)

- In your Python script:
 1. Fetching row by row:
 1. After execution get the number of results (*cur.rowcount*)
 2. In a FOR loop: Use *fetchone()* to get one row at a time.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb');

with con:

    cur = con.cursor()
    cur.execute("SELECT * FROM Writers")

    for i in range(cur.rowcount):

        row = cur.fetchone()
        print row[0], row[1]
```

DB PROGRAMMING HELLOWORLD

 Using a `mysqDB` (python 2.7x) or `mysqlclient` (python 3.x)

- In your Python script:
 - I. Working with user input: with regular string manipulation

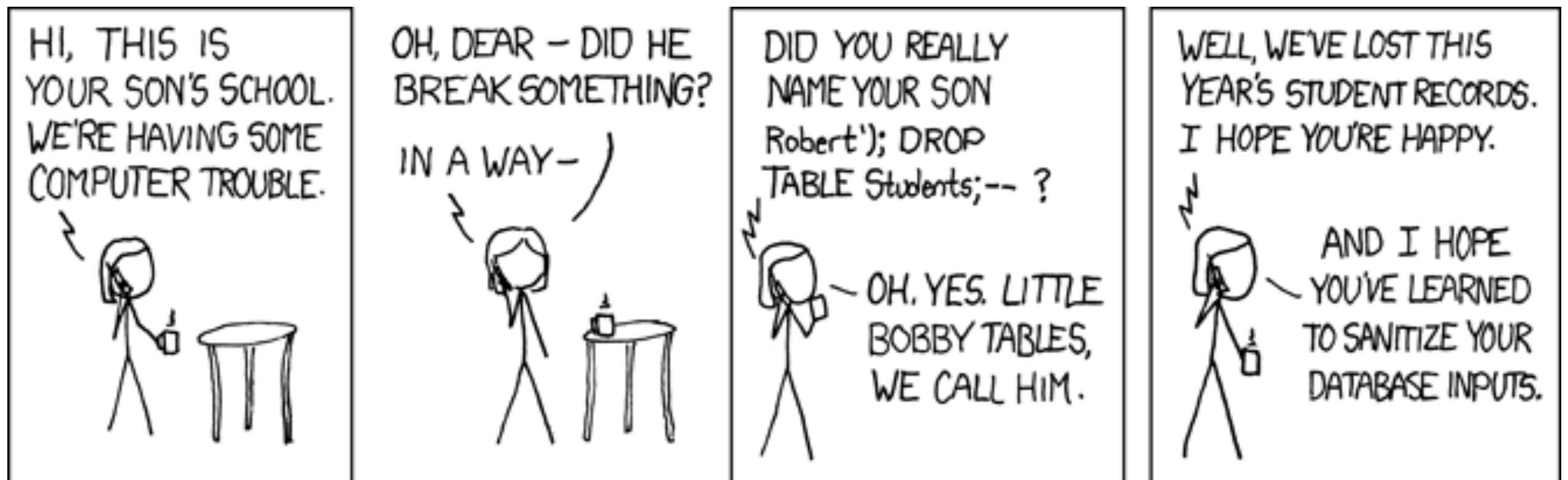
```
>>> student_name = raw_input("Enter a student name")
```

```
>>> query="SELECT * from Students WHERE FirstName = %s" % (student_name)
```

```
>>> Cur.execute(query)
```

LITTLE BOBBY TABLES

```
>>> student_name = raw_input("Robert');DROP TABLE Students; --")  
>>> query="SELECT * from Students WHERE FirstName = %s" % (student_name)  
>>> Cur.execute(query)
```



DB PROGRAMMING

HELLOWORLD

 Using a `mysqDB` (python 2.7x) or `mysqlclient` (python 3.x)

- In your Python script:
 - I. Using a “**Prepared Statement**” to:
 - Prevents the reparsing of SQL statements
 - Used for statements executed more than once

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import MySQLdb as mdb

con = mdb.connect('localhost', 'testuser', 'test623', 'testdb')

with con:

    cur = con.cursor()

    cur.execute("UPDATE Writers SET Name = %s WHERE Id = %s",
                ("Guy de Maupasant", "4"))

    print "Number of rows updated:", cur.rowcount
```

DB PROGRAMMING HELLOWORLD



Performing C U D operations:

- Commit() if everything went well
- Rollback() if there is something wrong

```
#!/usr/bin/python

import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# Prepare SQL query to UPDATE required records
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1
      WHERE SEX = '%c'" % ('M')

try:
    # Execute the SQL command
    cursor.execute(sql)
    # Commit your changes in the database
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()

# disconnect from server
db.close()
```

DB PROGRAMMING

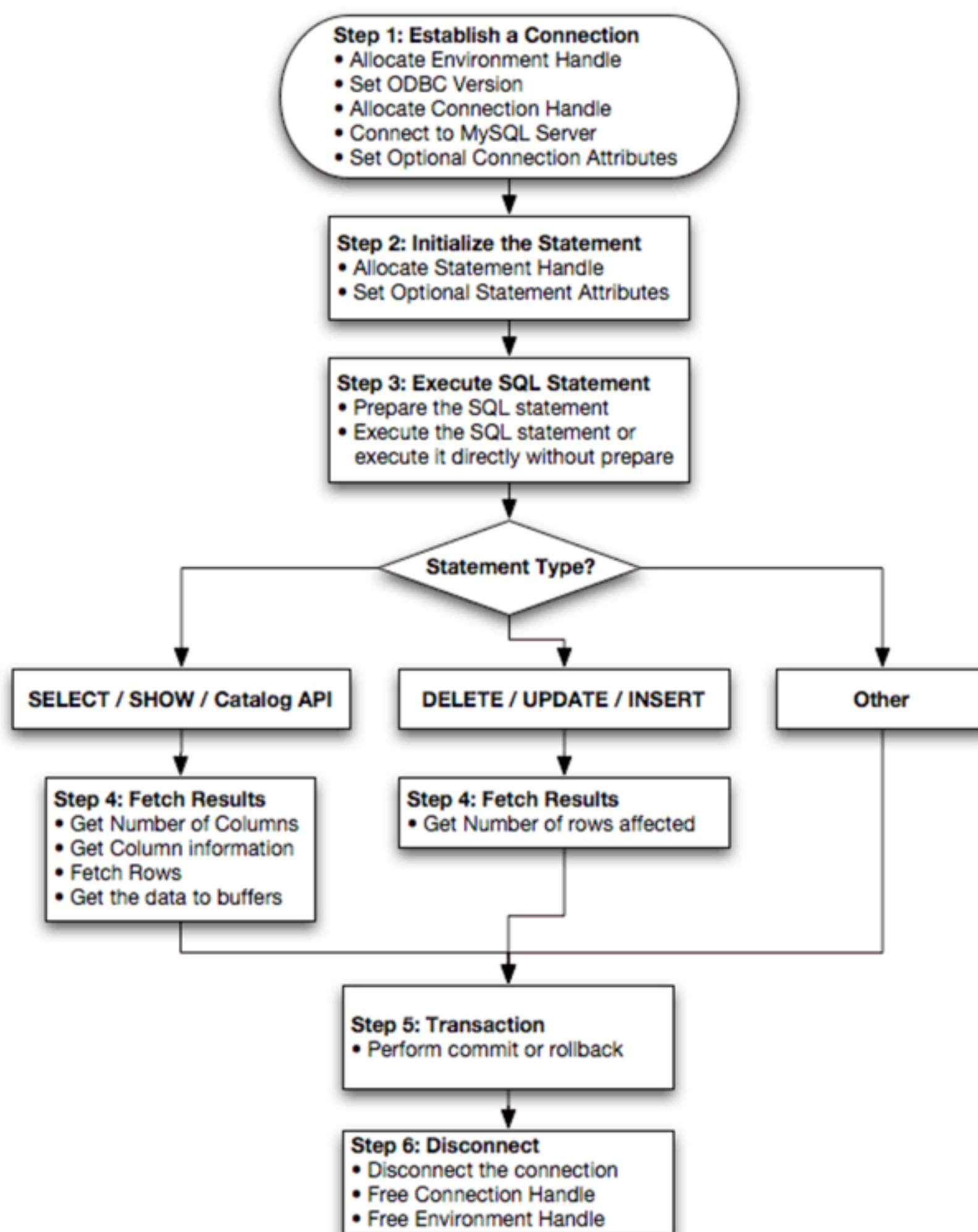
HELLOWORLD



Performing C U D operations:

- Using **Batch** CUD operations to boost performance:
 - If it not fast enough, auto-commit might be **ON**.
 - Add “SET autocommit 0;” to your SQL transaction.

```
con = mysql.connect(  
    host="localhost",  
    user="user",  
    passwd="**",  
    db="db name"  
)  
cur = con.cursor()  
  
for data in your_data_list:  
    cur.execute("data you want to insert: %s" %data)  
  
con.commit()  
con.close()
```



DB PROGRAMMING: GUIDELINES

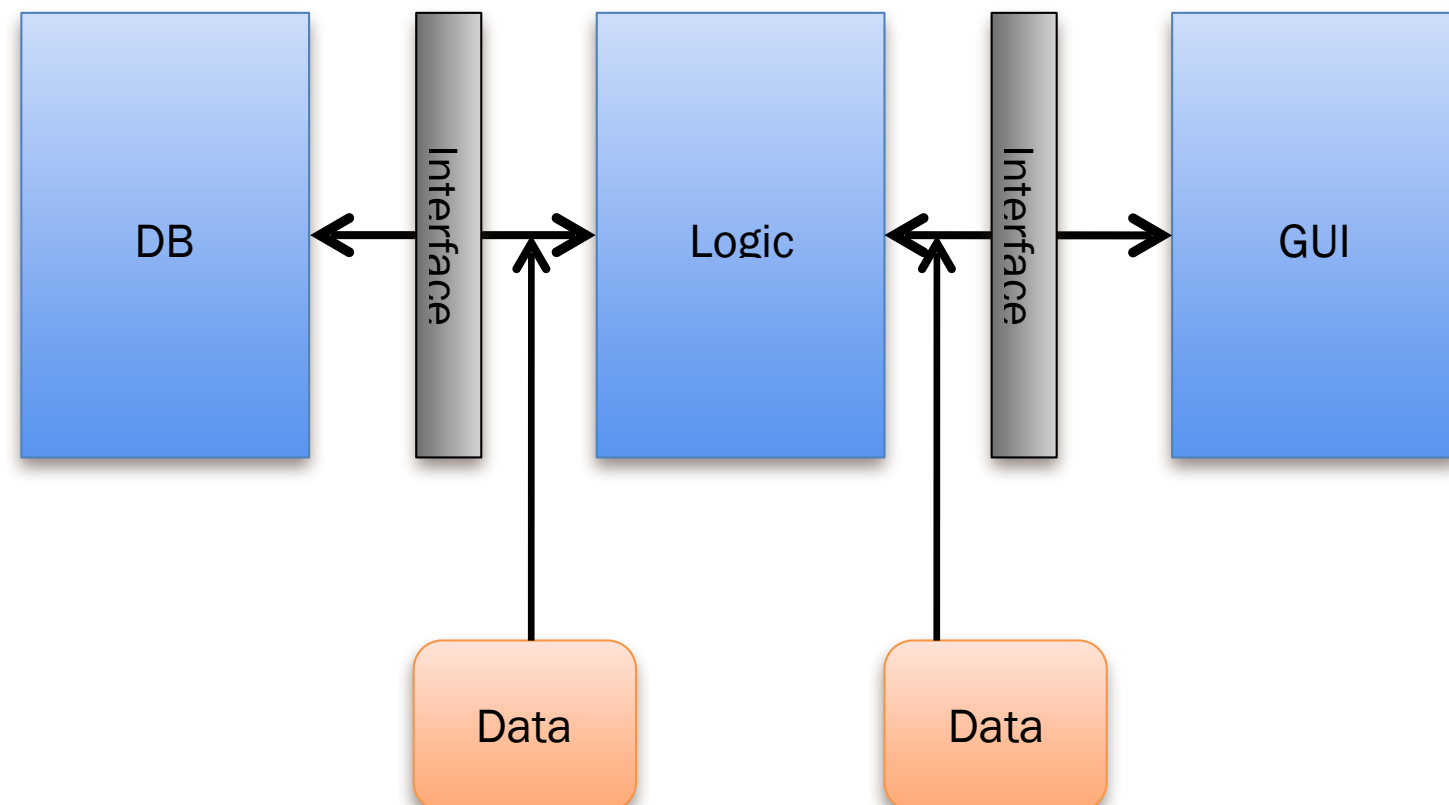
1. Use efficient SQL statements:

- “SELECT * FROM Students” vs “ SELECT `FirstName`,`LastName` FROM Students”












2. Secure your code

- Prepared statements
- Input sanitation.
- Define MySQL users correctly

3. Separate the DB layer from the UI layer:



AGENDA FOR TODAY

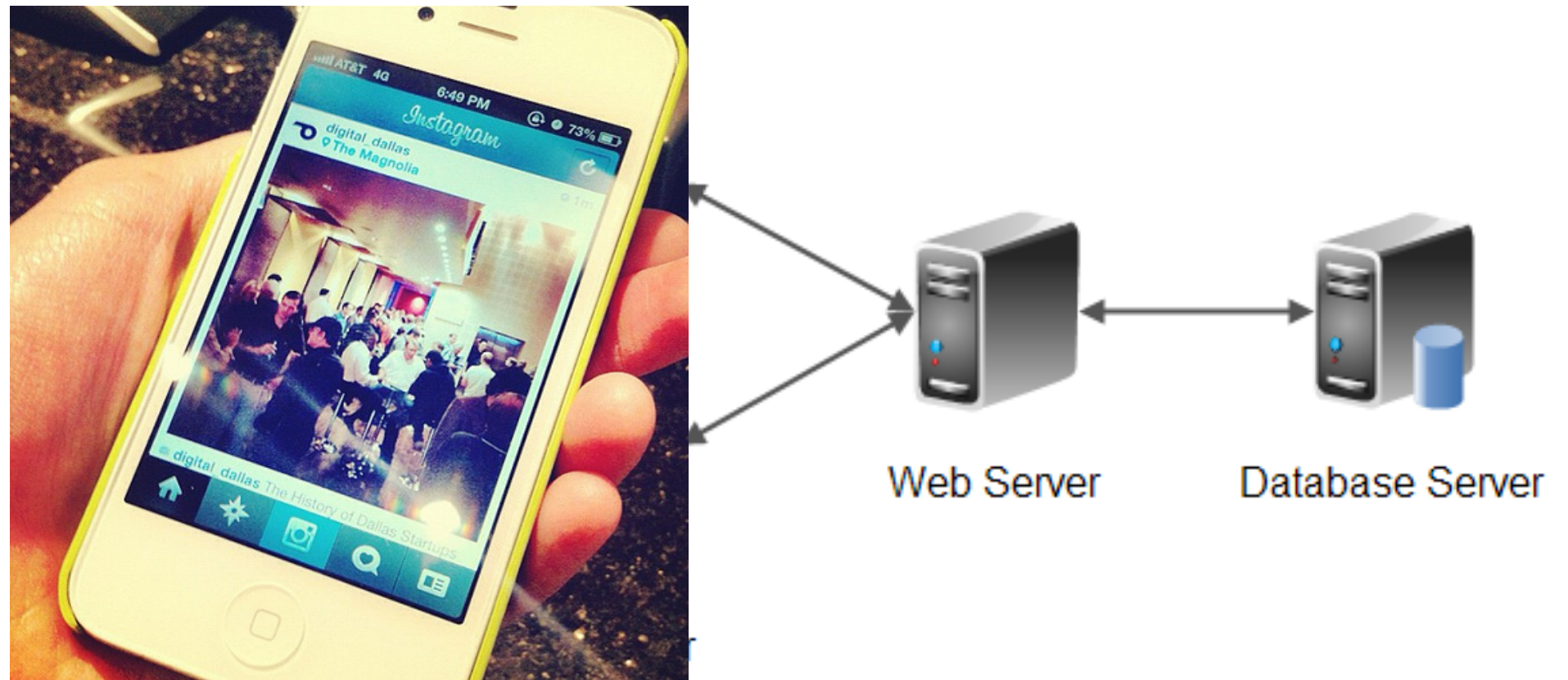
-  Advanced Mysql
 -  More than just SELECT
 -  Creating tables
 -  MySQL optimizations: Storage engines , indexing.
-  Database programming
 -  HelloWorld: Python + MySQL
-  **Recap: DB servers in the web**
-  **Web programming architecture**
-  **HTTP on a need-to-know basis.**
-  **Web APIs: REST ,json, and how to get them**
-  **The final project**

DATABASE ARCHITECTURE ON THE WEB (BRIEF)

🐟 Database server is a **standalone** server.

🐟 Database server is not accessible to web-users (when configured securely)

🐟 Only the web server communicates with the DB.



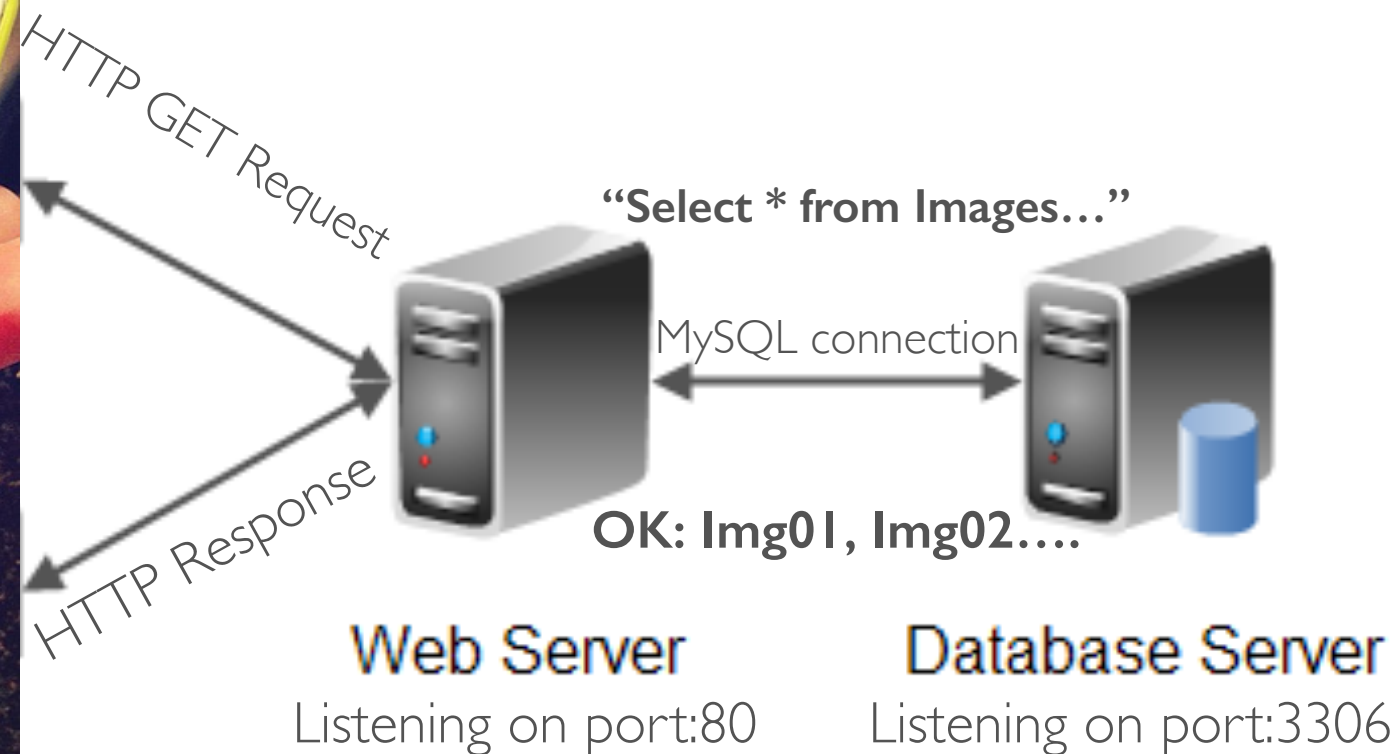
DATABASE ARCHITECTURE ON THE WEB (NETWORK)

🐟 Web browser and web server are communicating via the **HTTP protocol**.

🐟 Web servers (and MySQL clients) are communicating via the **MySQL protocol** (TCP)



Web Browser



WEB PROGRAMMING: DEFINITIONS

Web Server:

A computer program that accepts HTTP requests and return HTTP responses with optional data content.

A computer that runs a computer program as described above.

Most common platforms: **Apache, IIS (Microsoft), Enginex**

Web Client (browser):

A software application for retrieving, presenting, and traversing information resources on the World Wide Web

Usually parses HTML (HyperText Markup Language) , CSS and JavaScript and present it to the user as a **web page**. (More details on the next recitation).

Most common browser: **Firefox, Google Chrome, Ms Internet Explorer, Safari**

Web API (Application Programming Interface):

A publicly exposed endpoint to a defined request-response message system, (typically expressed in JSON or XML)

WEB PROGRAMMING: DEFINITIONS

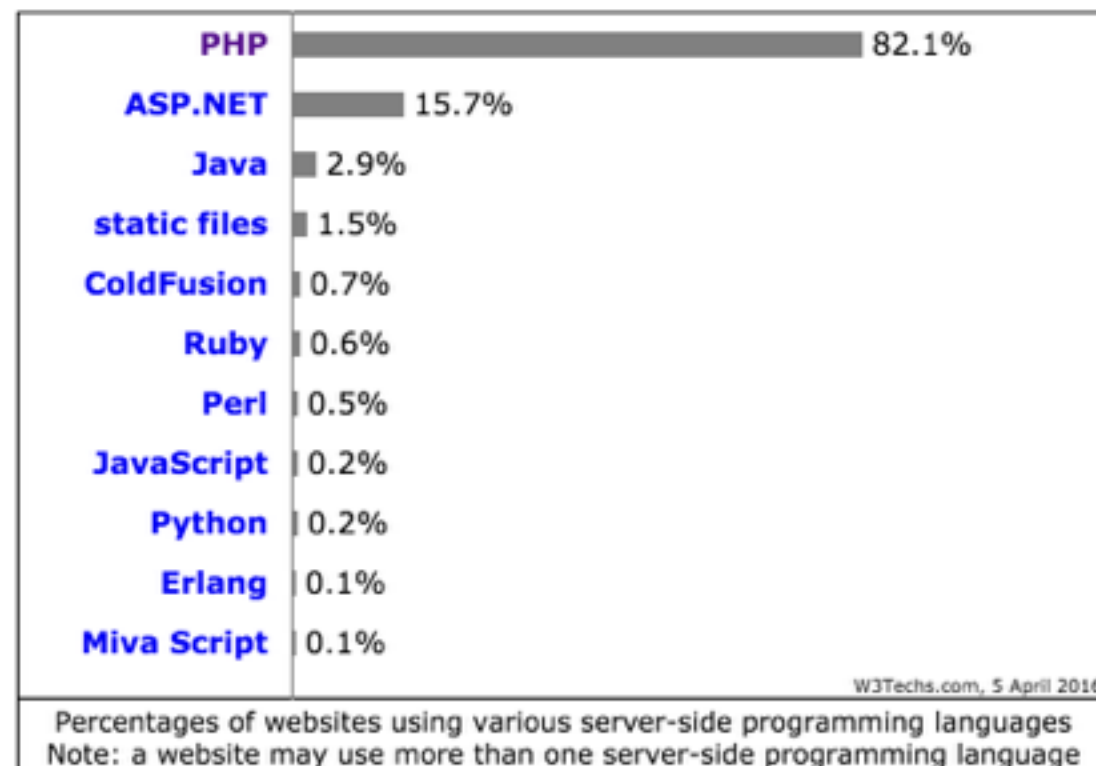
Web Server programming language:

A server-side programming language for executing code that reads HTTP requests and generates HTTP responses.

Designed for the web architecture:

- Multiple clients accessing a web server on the same
- Content is dynamic

Most programming languages can handle HTTP requests (e.g., C, C++, Python, Java etc.)



INTRO TO HTTP

HTTP (Hyper Text Transfer Protocol)

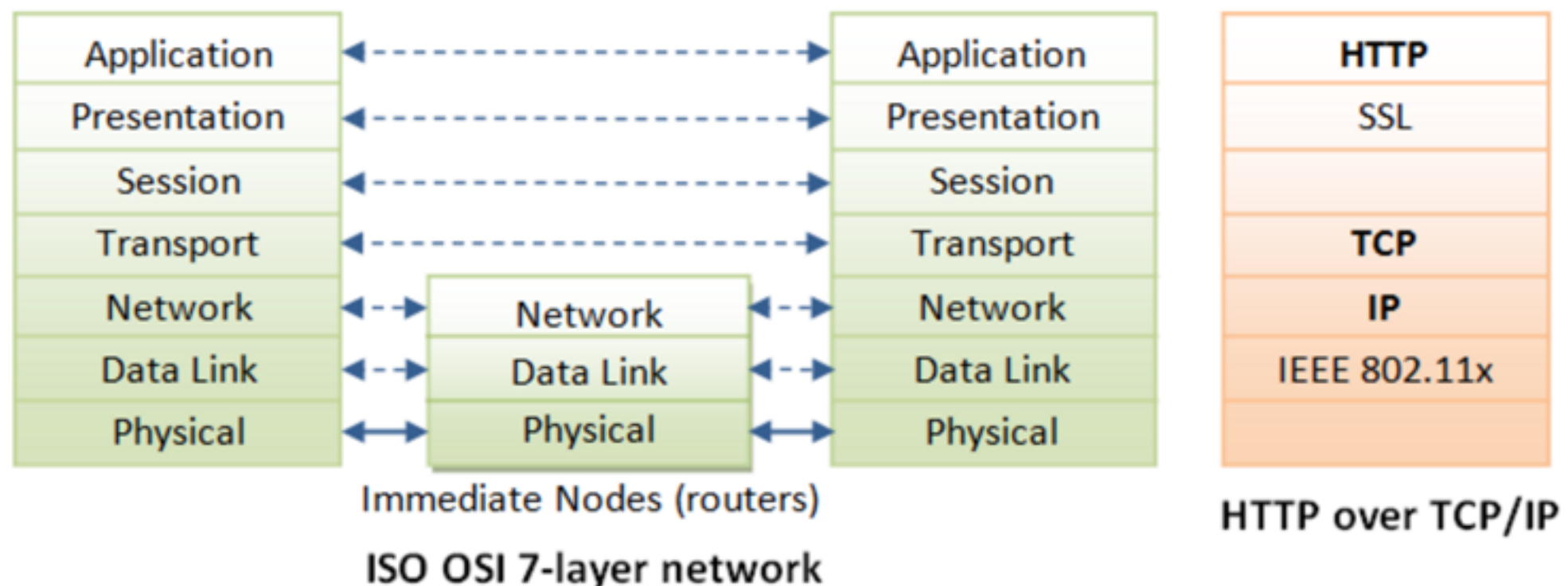
An **application layer** protocol

HyperText: A text displayed on a computer display or other electronic devices with references (hyperlinks) to other text which the reader can immediately access, or where text can be revealed progressively at multiple levels of detail

Based on **Client Requests** of **Resources (URI)** and **Server Response**

Resources to be accessed by HTTP are identified using Uniform Resource Identifiers (URIs).

Can be referring to web pages, media (image/video) or other data objects.



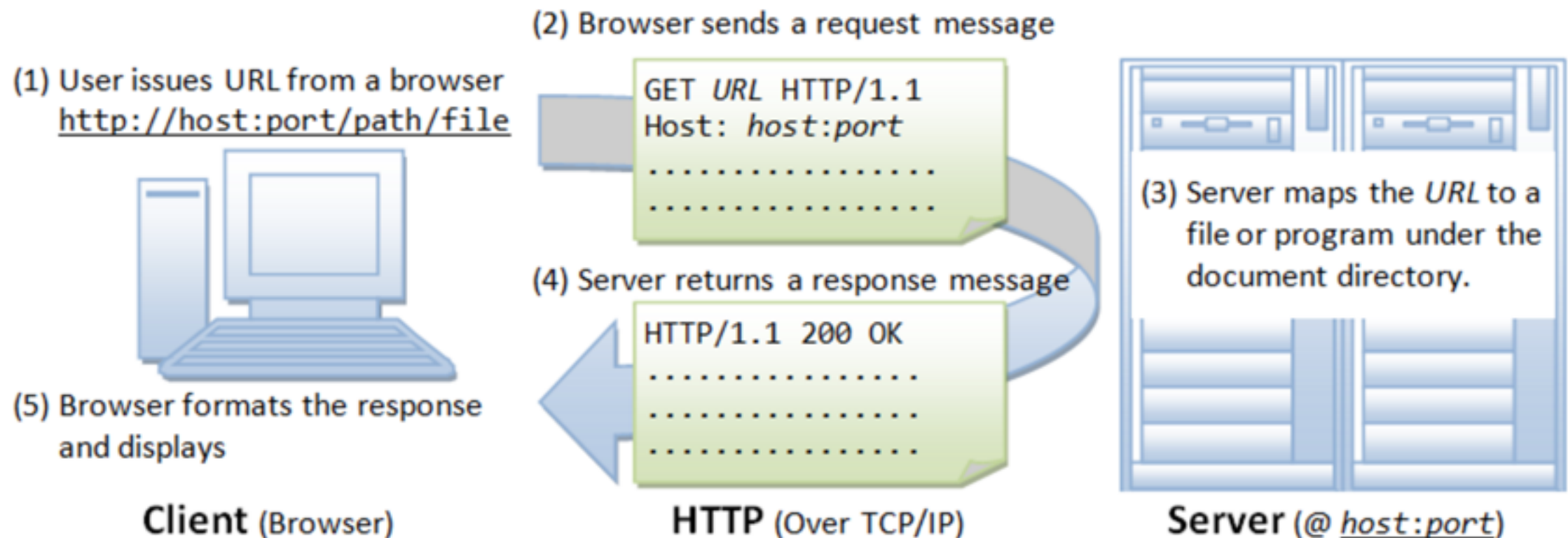
INTRO TO HTTP

HTTP Session

An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80,)

An HTTP server listening on that port waits for a client's request message.

Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own.



INTRO TO HTTP

🐟 HTTP Requests

Most common client requests are **HTTP GET** and **HTTP POST**

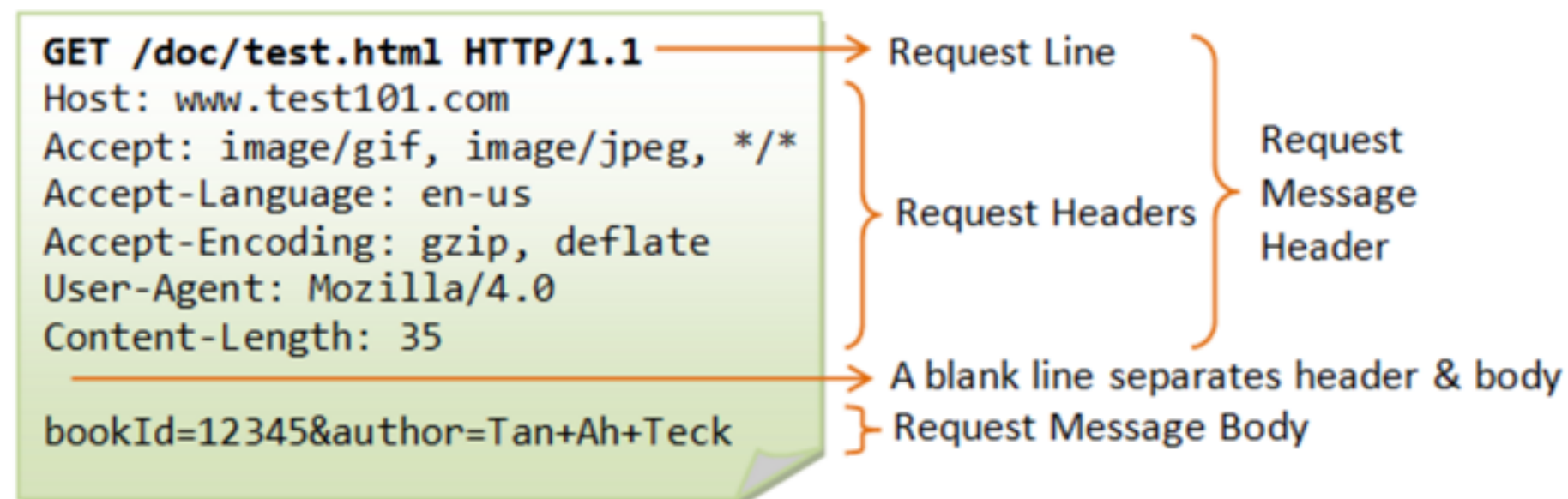
HTTP GET can transfer parameters within the URL

Example: <https://www.google.co.il/?q=database+systems>

HTTP POST is used to post data up to the web server

🐟 HTTP Request headers

Used to pass information to the web server such as language, supported encoding, User-Agent, etc.



INTRO TO HTTP

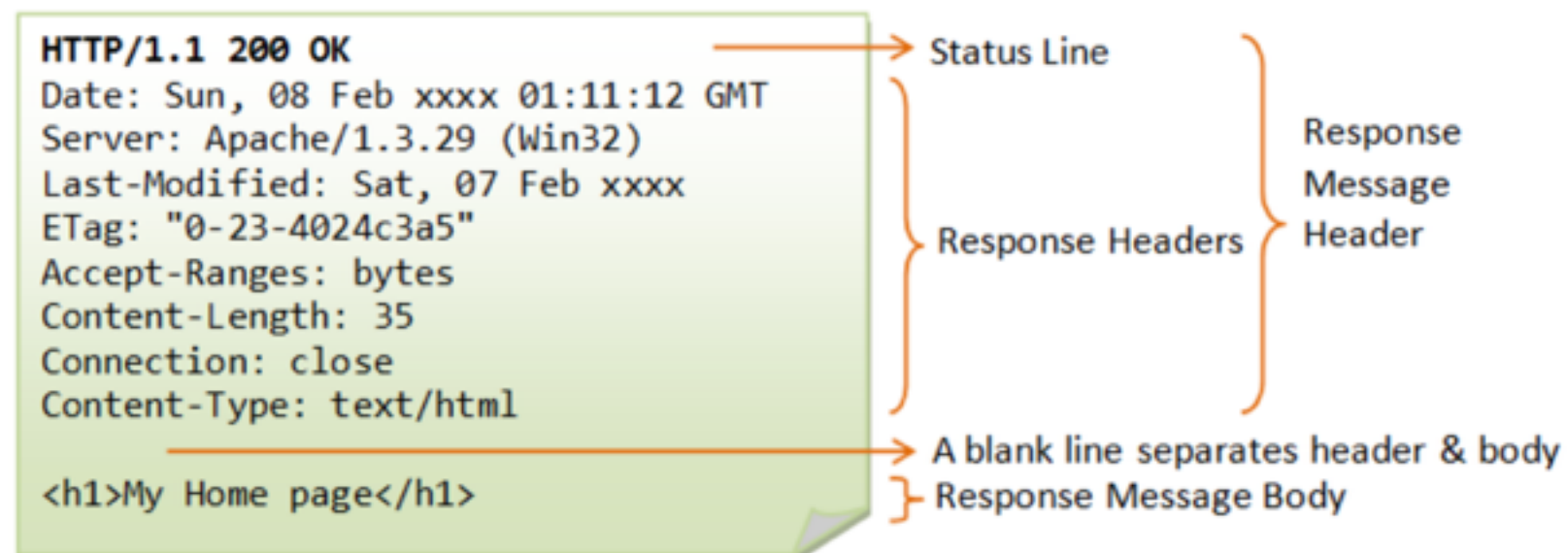
HTTP Response

The first line is called the status line, followed by optional response header(s).

The status line has the following syntax:

- HTTP-version status-code reason-phrase
- HTTP-version: The HTTP version used in this session. Either HTTP/1.0 and HTTP/1.1.
- status-code: a 3-digit number generated by the server to reflect the outcome of the request.
- reason-phrase: gives a short explanation to the status code.

Common status code and reason phrase are "200 OK", "404 Not Found", "403 Forbidden", "500 Internal Server Error".



THE FINAL PROJECT

🐟 Designing and implementing a (cool) web application

🐟 Coding in PHP or Python

🐟 Teams of 4-5 (send me your names)

🐟 Data retrieving and processing from known APIs:

🐟 Facebook API

🐟 Youtube API

🐟 Wikipedia API (DBPedia)

🐟 StackExchange API

🐟 UI is in HTML

ON THE NEXT LECTURE

 The very basics of web programming:

 Installing Xampp (Apache, MySQL, PHP)

 Introduction PHP and server side scripting

 Introduction to HTML, CSS and JavaScript programming

 How to work with web APIs such as Facebook API, YoutubeAPI etc.