TÉCNICO LISBOA

Bases de Dados

T18 - Desenvolvimento Web e Bases de Dados

Prof. Daniel Faria
Prof. Flávio Martins

Sumário

- Aplicações com Bases de Dados
- Arquitetura de Aplicações Web

TÉCNICO LISBOA

Aplicações com Bases de Dados

Modelos de integração

Estática - Embedded SQL

Embedded SQL (C)

SQL-J (Java)

Dinâmica - Conector de base de dados (Driver)

PHP Data Objects (PDO)

Java DataBase Connectivity (JDBC)

Adaptador PostgreSQL para Python (Psycopg)

Embedded SQL (C/C++)

- Ligação à base de dados:

  EXEC SQL CONNECT

- Declaração de variáveis:

  EXEC SQL BEGIN (END) DECLARE SECTION

- Comandos

  EXEC SQL Statement;

Embedded SQL: Variáveis

```
EXEC SQL BEGIN DECLARE SECTION
char  c_sname[20];
long  c_sid;
short c_rating;
float  c_age;
EXEC SQL END DECLARE SECTION
```

Variáveis de erro (pré-definidas):

- long **SQLCODE** (negativo se existir um erro)
- char[6] **SQLSTATE** (indica o tipo do erro)

Embedded SQL: Exemplo

```
char SQLSTATE[6];

EXEC SQL BEGIN DECLARE SECTION
short c_minrating;
char  c_sname[20];
float  c_age;
EXEC SQL END DECLARE SECTION

c_minrating = random();
EXEC SQL DECLARE c1 CURSOR FOR
  SELECT S.sname, S.age  FROM Sailors S
  WHERE S.rating > :c_minrating
  ORDER BY S.sname;
do {
  EXEC SQL FETCH c1 INTO :c_sname, :c_age;
  printf("%s is %d years old\n", c_sname, c_age);
} while (SQLSTATE != '02000');
EXEC SQL CLOSE c1;
```

Embedded SQL: Prepared Statement

```
char SQLSTATE[6];

EXEC SQL BEGIN DECLARE SECTION
char  c_prepst[80];
char  c_minrating[20];
char  c_sname[20];
float  c_age;
EXEC SQL END DECLARE SECTION

strcpy( c_prepst, "SELECT S.sname, S.age  FROM Sailors S WHERE S.rating > ? ORDER BY S.sname;" );
EXEC SQL PREPARE s1 FROM :prep_st;

strcpy( c_minrating, "4" );
EXEC SQL OPEN c1 USING :c_minrating;

EXEC SQL DECLARE c1 CURSOR FOR s1;
do {
  EXEC SQL FETCH c1 INTO :c_sname, :c_age;
  printf("%s is %d years old\n", c_sname, c_age);
} while (SQLSTATE != '02000');
EXEC SQL CLOSE c1;
```

Dinâmica: Conector de base de dados

O resultado de qualquer interrogação SQL é um **conjunto de tuplos**

As linguagens de programação imperativas como o Python não implementam o tipo de dados **conjunto de tuplos**

O conector de base de dados usa **cursores** como mecanismo de adaptação dos **conjuntos de tuplos** à linguagem imperativa

Conector de bases de dados Python

PEP 249 – Python Database API Specification v2.0

Esta API foi definida para encorajar a similaridade entre os módulos Python que são usados para aceder a bases de dados. Ao fazer isso, esperamos alcançar uma consistência que leve a módulos mais facilmente compreendidos, código que geralmente é mais portátil entre bases de dados e um alcance mais amplo de conectividade a bases de dados através de Python.

Psycopg – Adaptador de PostgreSQL para Python

Psycopg 3 é um adaptador de bases de dados PostgreSQL para a linguagem de programação Python e apresenta uma interface familiar para todos que já usaram o Psycopg 2, mas permite o uso de recursos mais modernos do PostgreSQL e Python, como:

- Suporte para async
- Suporte a COPY de objetos Python
- Um pool de conexão redesenhado
- Suporte para static typing
- Server-side parameters binding
- Prepared statements
- Pipeline de declarações
- Comunicação binária
- Acesso direto às funcionalidades da libpq

Psycopg Prepared Statement: Exemplo

Utilizando os objectos do Psycopg com context managers (i.e. com `with`) garante o fecho automático das conexões e a libertação dos recursos no final do bloco (no psycopg3).

- connect() retorna uma Connection
  - conn.cursor() retorna Cursor
  - conn.commit() e conn.rollback()
- Executar SQL
  - cur.execute()
  - cur.executemany()
- Ir buscar conjuntos de tuplos
  - cur.fetchone()
  - cur.fetchmany()

```python
# Note: the module name is psycopg, not psycopg3
import psycopg

# Connect to an existing database
with psycopg.connect("dbname=test user=postgres") as conn:

    # Open a cursor to perform database operations
    with conn.cursor() as cur:

        # Pass data to fill a query placeholders and let Psycopg perform
        # the correct conversion (no SQL injections!)
        cur.execute(
            "INSERT INTO test (num, data) VALUES (%s, %s)",
            (100, "abcdef"))

        # Query the database and obtain data as Python objects.
        cur.execute("SELECT * FROM test")
        cur.fetchone()
        # will return (1, 100, "abcdef")

        # You can use `cur.fetchmany()`, `cur.fetchall()` to return a list
        # of several records, or even iterate on the cursor
        for record in cur:
            print(record)

        # Make the changes to the database persistent
        conn.commit()
```

Psycopg
com context manager

Utilizando os objectos do Psycopg com context managers (i.e. com `with`) garante o fecho automático das conexões e a libertação dos recursos no final do bloco (no psycopg3).

- É inserido um record na tabela
- Ao sair do contexto de `with` .. as conn
  - commit()
  - close()

```
with psycopg.connect() as conn:

    cur = conn.cursor()

    cur.execute("SELECT count(*) FROM my_table")
    # This function call executes:
    # - BEGIN
    # - SELECT count(*) FROM my_table
    # So now a transaction has started.

    cur.execute("INSERT INTO data VALUES (%s)", ("Hello",))
    # This statement is executed inside the transaction

# No exception at the end of the block:
# COMMIT is executed.
```

Psycopg
sem context manager

Utilizando os objectos do Psycopg sem usar context managers (i.e. sem `with`) não garante que o fecho das conexões e a libertação dos recursos no final do bloco.

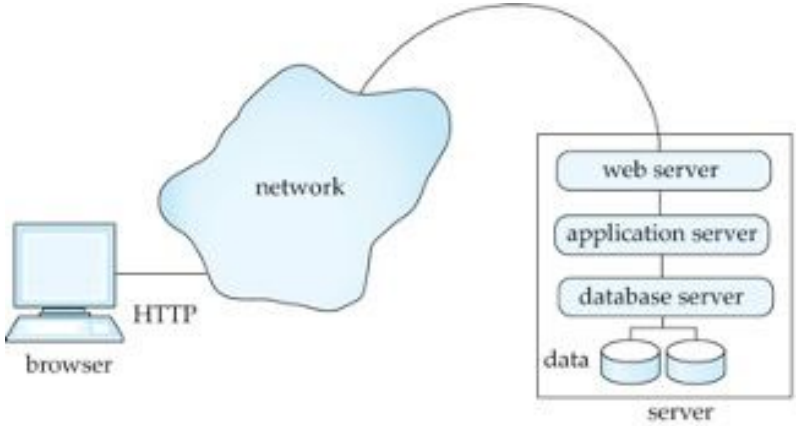
- Não é inserido nenhum tuplo na tabela
- É necessário chamar conn.**commit()**

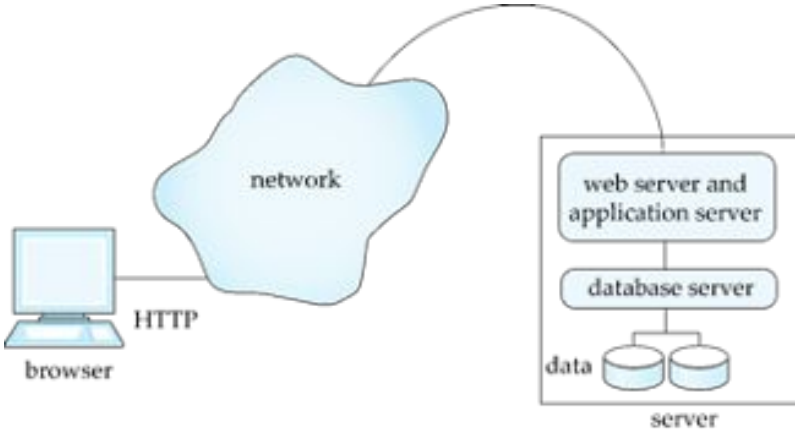
```
conn = psycopg.connect()

# Creating a cursor doesn't start a transaction or affect the connection
# in any way.
cur = conn.cursor()

cur.execute("SELECT count(*) FROM my_table")
# This function call executes:
# - BEGIN
# - SELECT count(*) FROM my_table
# So now a transaction has started.

# If your program spends a long time in this state, the server will keep
# a connection "idle in transaction", which is likely something undesired

cur.execute("INSERT INTO data VALUES (%s)", ("Hello",))
# This statement is executed inside the transaction

conn.close()
# No COMMIT was sent: the INSERT was discarded.
```

TÉCNICO LISBOA

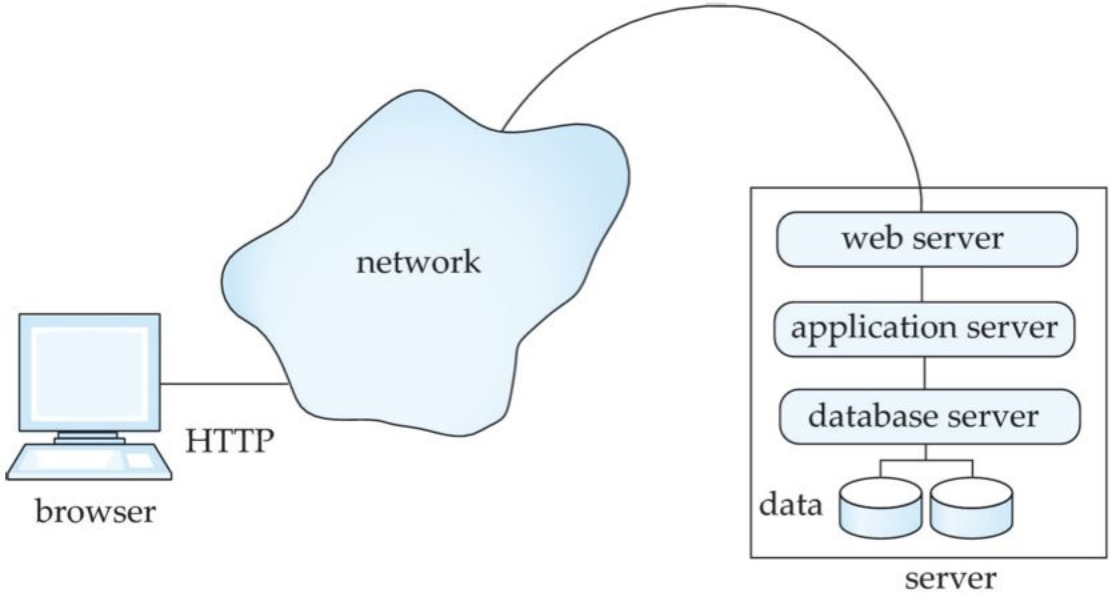
Arquitetura de Aplicações Web

Arquitetura de Aplicações Web





Arquitectura de Aplicações Web



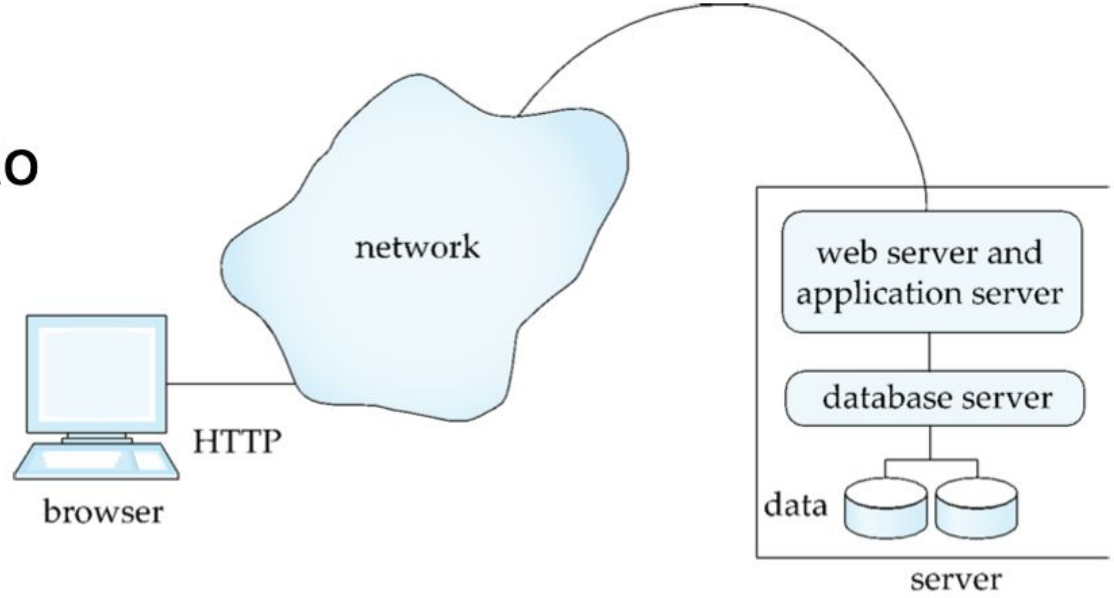
(3 camadas)

Arquitectura de Aplicações Web

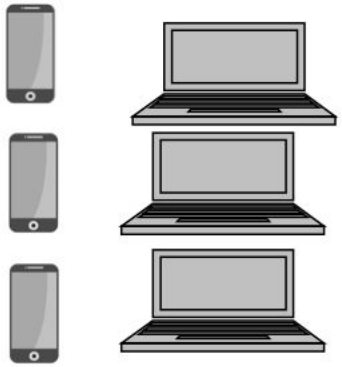
Cada indirecção
entre níveis de abstração

tem custos acrescidos

Arquitectura com 2 camadas



Distribuição Física (3 Fileiras)



Web Browsers
JavaScript

HTML, XML
HTTP

Servidor Aplicacional
Java, Python, PHP

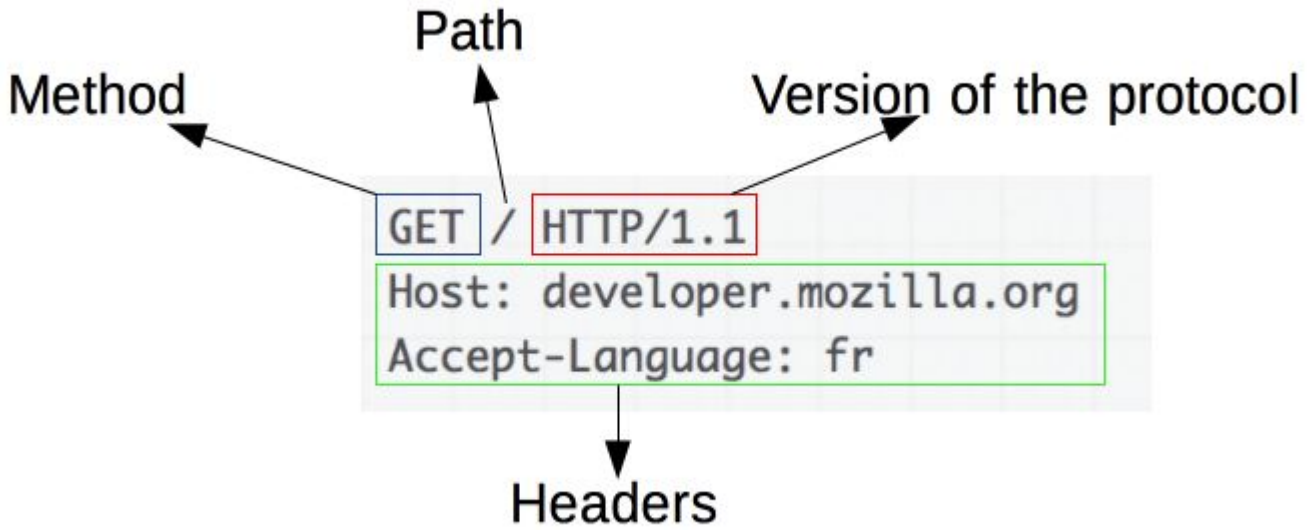
EX: web2.tecnico.ulisboa.pt

SQL
JDBC, ODBC

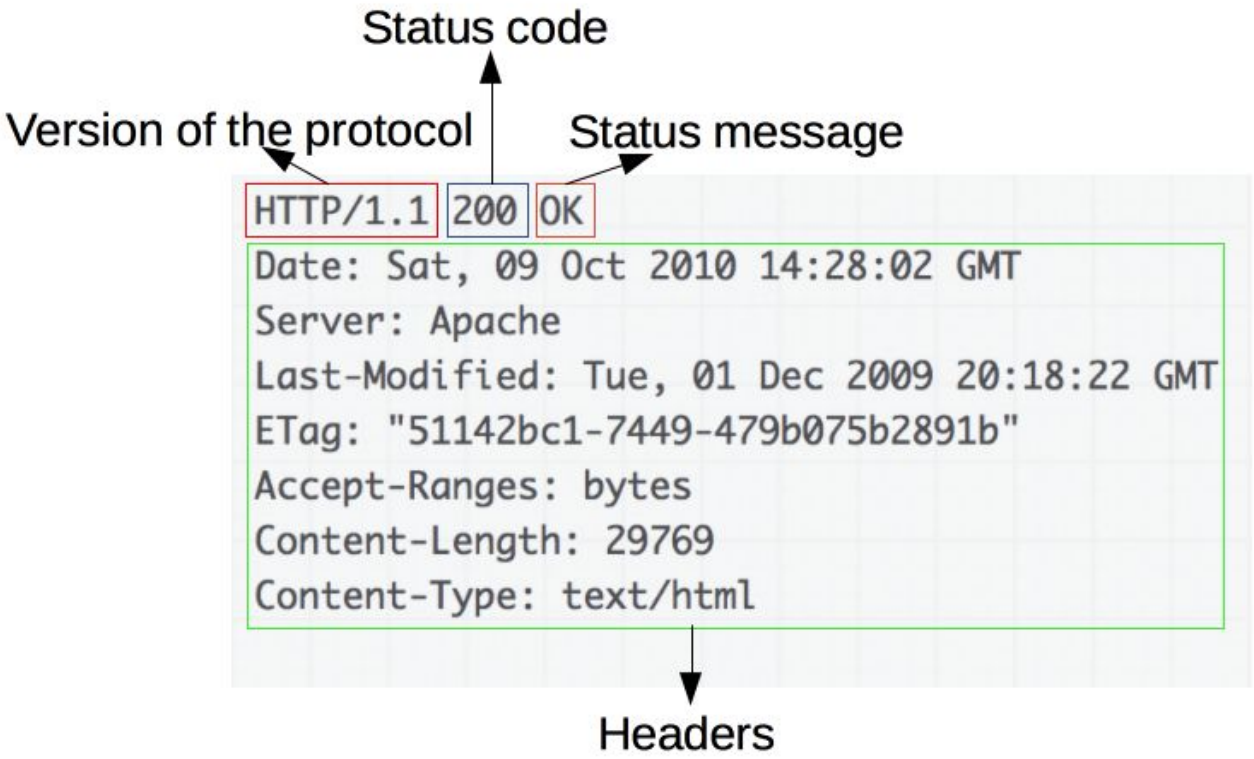
Servidor de Dados – SGBD
MySQL/MariaDB, Postgres

EX: db.tecnico.ulisboa.pt

HTTP Request



HTTP Response



Charset UTF-8

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

Referências

- https://developer.mozilla.org/en-US/docs/Web/HTML
- https://developer.mozilla.org/en-US/docs/Web/HTTP
- https://www.psycopg.org/psycopg3/docs/basic/usage.html
- https://www.psycopg.org/psycopg3/docs/basic/transactions.html