Manipulação de dados: Pandas e MySQL

Prof.: Leandro Bezerra Marinho

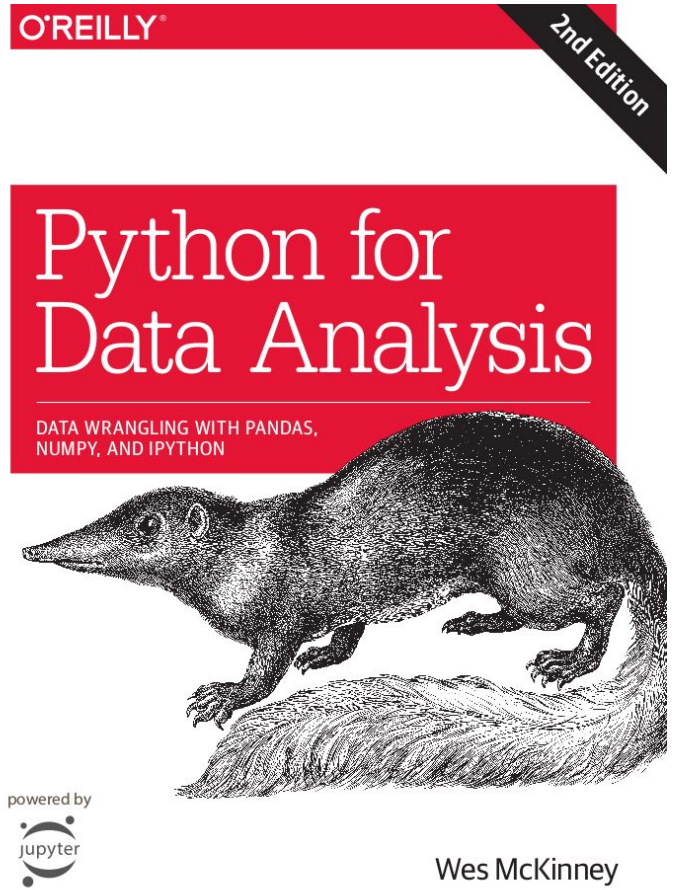
Objetivo

- Aprender os principais comandos da biblioteca Pandas.
- Entender os conceitos básicos de um banco de dados relacional.
- Compreender o SQL usando o MySQL.
- Integrar MySQL com Python.
- Conhecer as principais características do banco de dados não relacional.

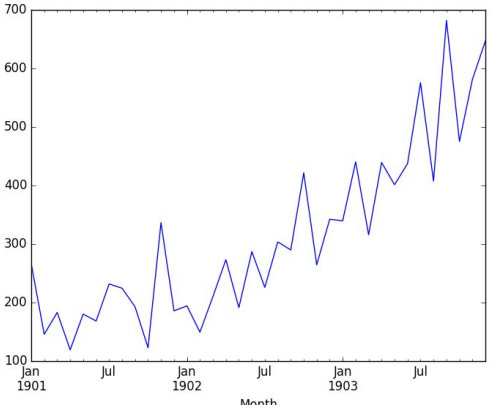
O que é o Pandas?

É uma **biblioteca** open-source para **Python** que pode ser utilizada facilmente para manipulação de **dados**.

https://pandas.pydata.org/



Que tipo de dados?



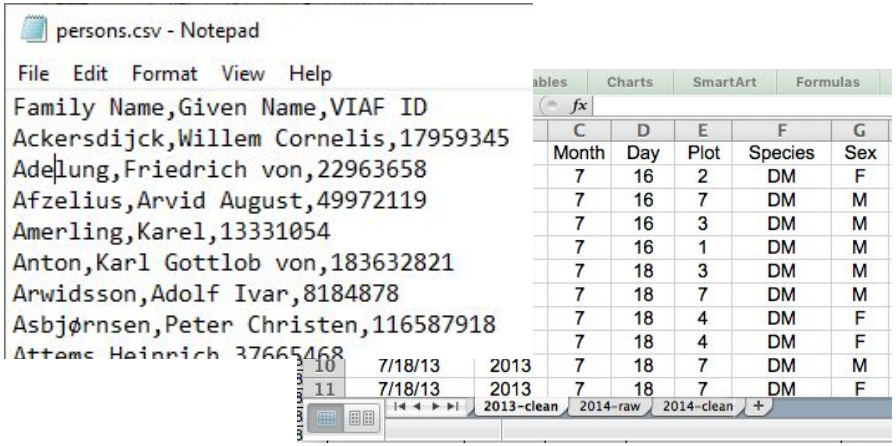
Dados estruturados:

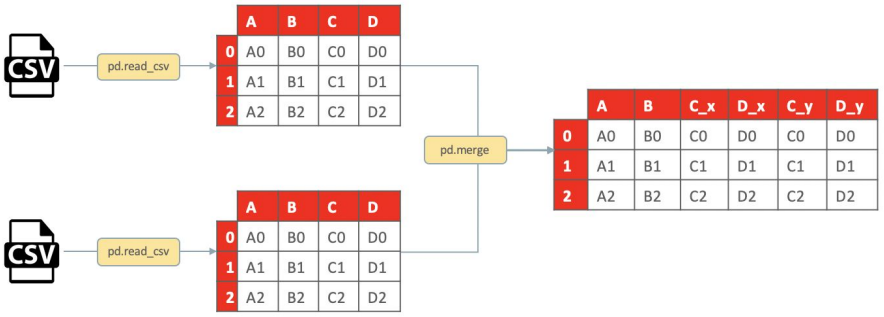
- Dados tabulares ou de planilhas
- Matrizes multidimensionais
- Várias tabelas de dados inter-relacionadas por colunas de chave
- Séries temporais

```
array([[[ 1., 11., 21.],
        [nan, nan, nan],
        [nan, nan, nan]],

       [[ 5., 12., 22.],
        [ 3., 13., 23.],
        [nan, nan, nan]],

       [[ 3., 14., 24.],
        [ 3., 15., 25.],
        [ 1., 16., 26.]]])
```





Como os dados são estruturados?

DataFrame: estrutura tabular de N-dimensões, onde cada coluna é um campo da tabela e cada linha um registro.

Series: matriz unidimensional que contém uma sequência de valores acompanhado de seus respectivos índices.

	Name	Team	Number
0	Avery Bradley	Boston Celtics	0.0
1	John Holland	Boston Celtics	30.0
2	Jonas Jerebko	Boston Celtics	8.0
3	Jordan Mickey	Boston Celtics	NaN
4	Terry Rozier	Boston Celtics	12.0
5	Jared Sullinger	Boston Celtics	7.0
6	Evan Turner	Boston Celtics	11.0
index	values	values	values

	Name
0	Avery Bradley
1	John Holland
2	Jonas Jerebko
3	Jordan Mickey
4	Terry Rozier
5	Jared Sullinger
6	Evan Turner
index	values

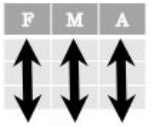
https://github.com/lapisco/pandas-mysql-practices

Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas

In a tidy data set:

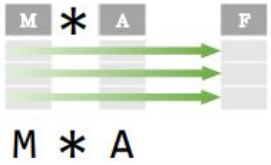


&

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements pandas's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

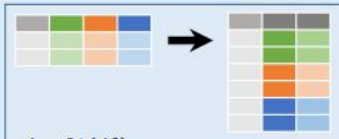
```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
          names=['n','v']))
```

Create DataFrame with a MultiIndex

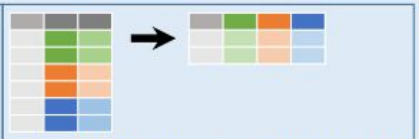
Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

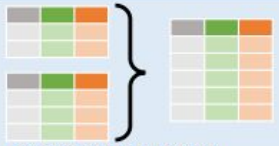
Reshaping Data – Change the layout of a data set



`pd.melt(df)`
Gather columns into rows.



`df.pivot(columns='var', values='val')`
Spread rows into columns.



`pd.concat([df1,df2])`
Append rows of DataFrames

`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df.sort_values('mpg',ascending=False)`
Order rows by values of a column (high to low).

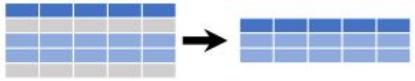
`df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df.sort_index()`
Sort the index of a DataFrame

`df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns=['Length','Height'])`
Drop columns from DataFrame

Subset Observations (Rows)



`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

`df.sample(frac=0.5)`
Randomly select fraction of rows.

`df.sample(n=10)`
Randomly select n rows.

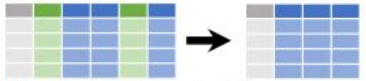
`df.iloc[10:20]`
Select rows by position.

`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

Logic in Python (and pandas)			
<	Less than	!=	Not equal to

Subset Variables (Columns)



`df[['width','length','species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

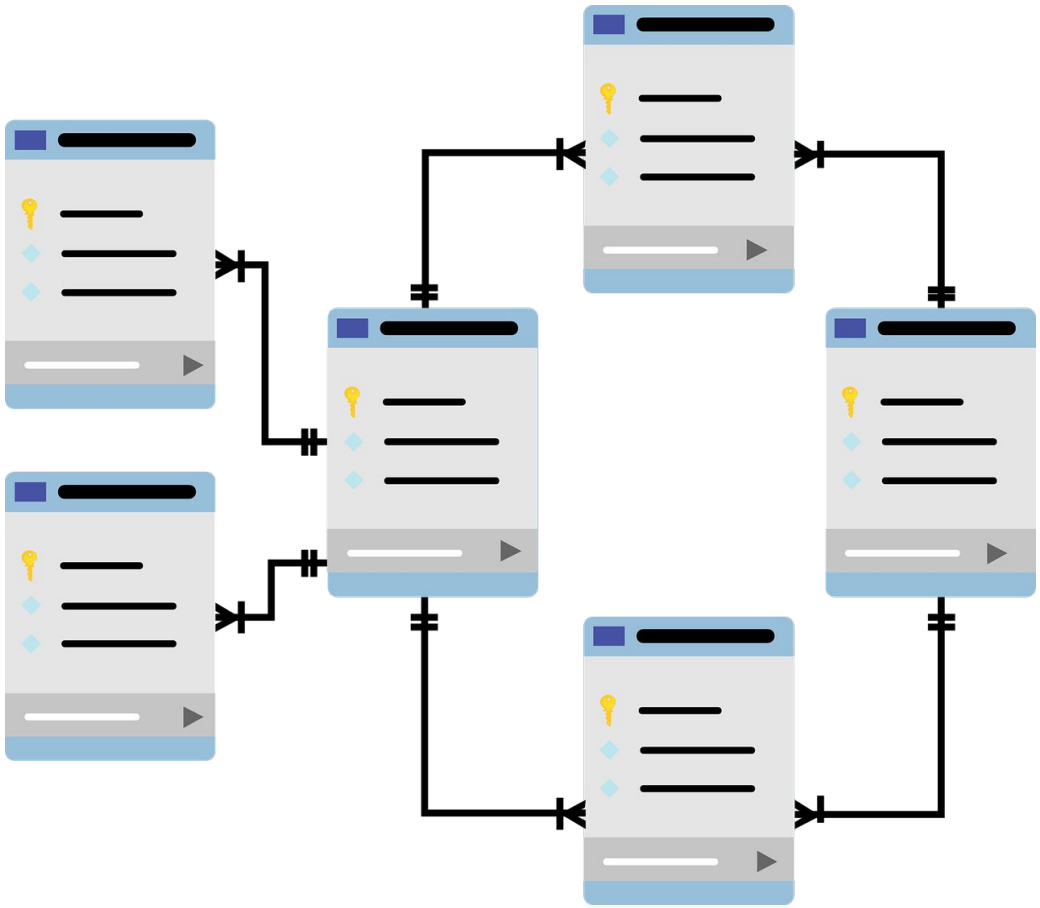
`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

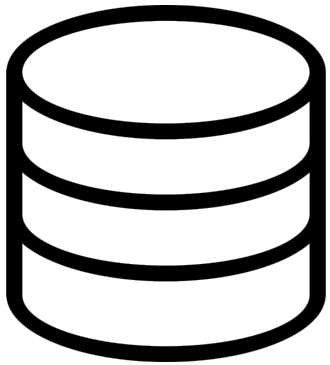
regex (Regular Expressions) Examples	
'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
'^(?!Species$).*'	Matches strings except the string 'Species'

`df.loc[:,'x2':'x4']`

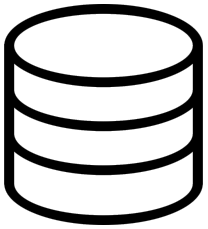
https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Banco de Dados Relacional



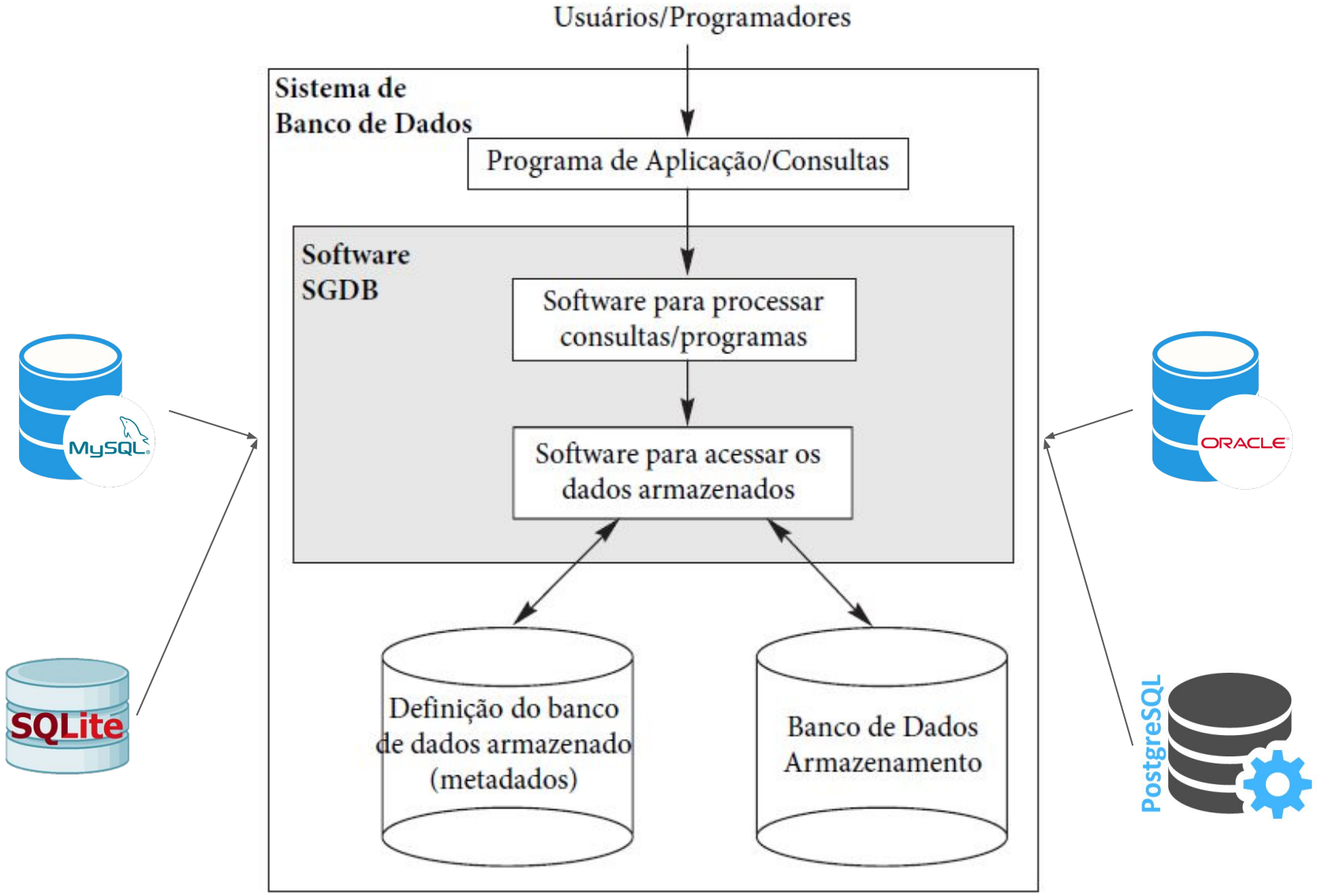


Banco de Dados Relacional



- Tipo de **banco de dados** que **armazena** e fornece acesso a dados **relacionados** entre si.
- São baseados no **modelo relacional (**tabelas).
- Cada **linha** na tabela é um **registro** com um identificador exclusivo chamada chave.
- Colunas da tabela contém **atributos** dos dados e cada registro tem um **valor** por atributo.
- Usa a Linguagem de Consulta Estruturada (SQL).

Sistema de Banco de Dados, SGBD e Banco de Dados



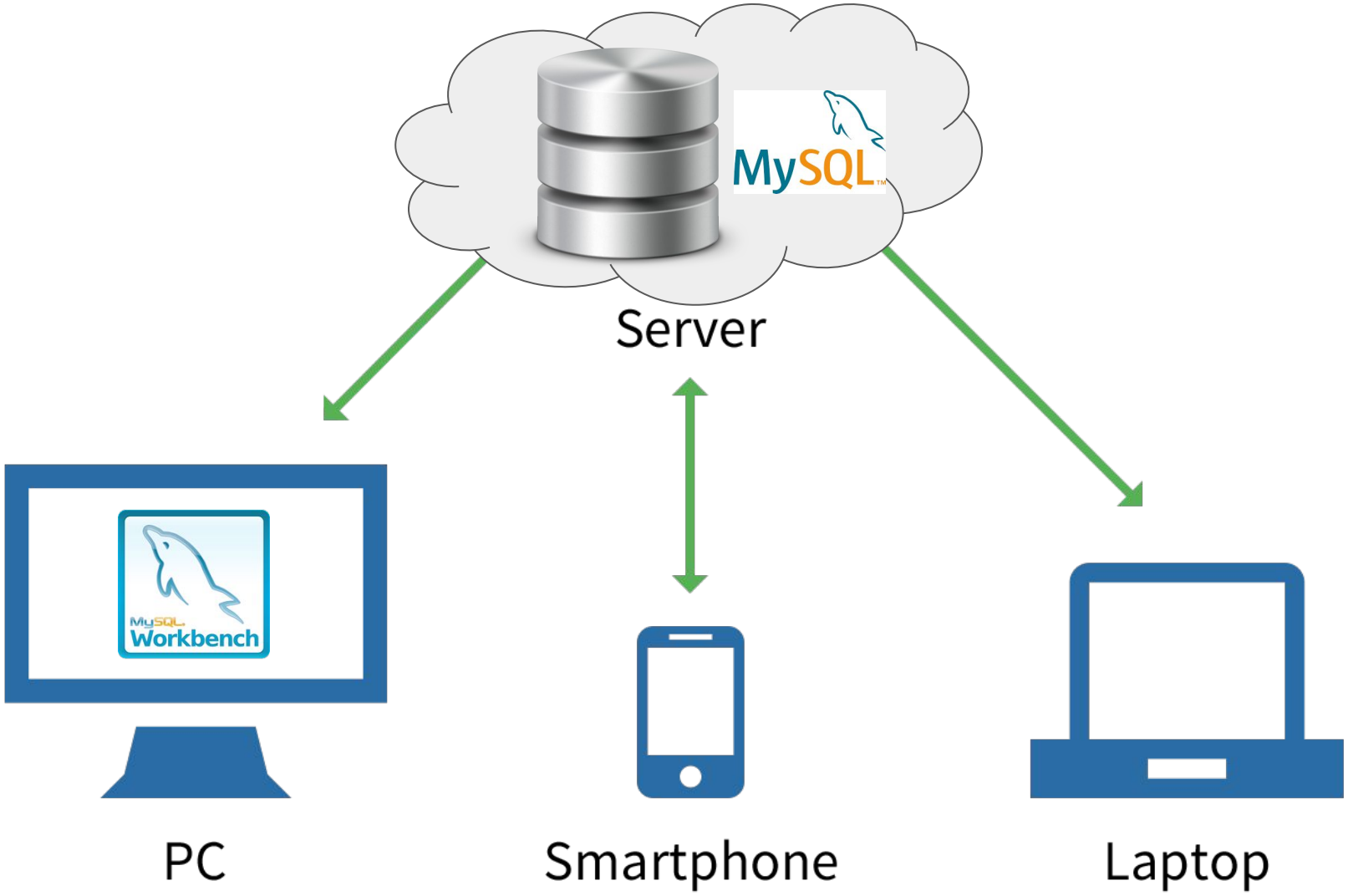
Vantagens do SGBD

- Estrutura de armazenamento e técnicas de pesquisa **eficientes**
- **Redundância** pode ser **reduzida**
- Dados podem ser **compartilhados**
- Suporte a **transações** pode ser fornecido
- **Backup** de recuperação
- Permite **ações** usando **regras**
- **Integridade** pode ser mantida
- **Segurança** pode ser reforçada
- **Separa** armazenamento **físico** do **conceitual**

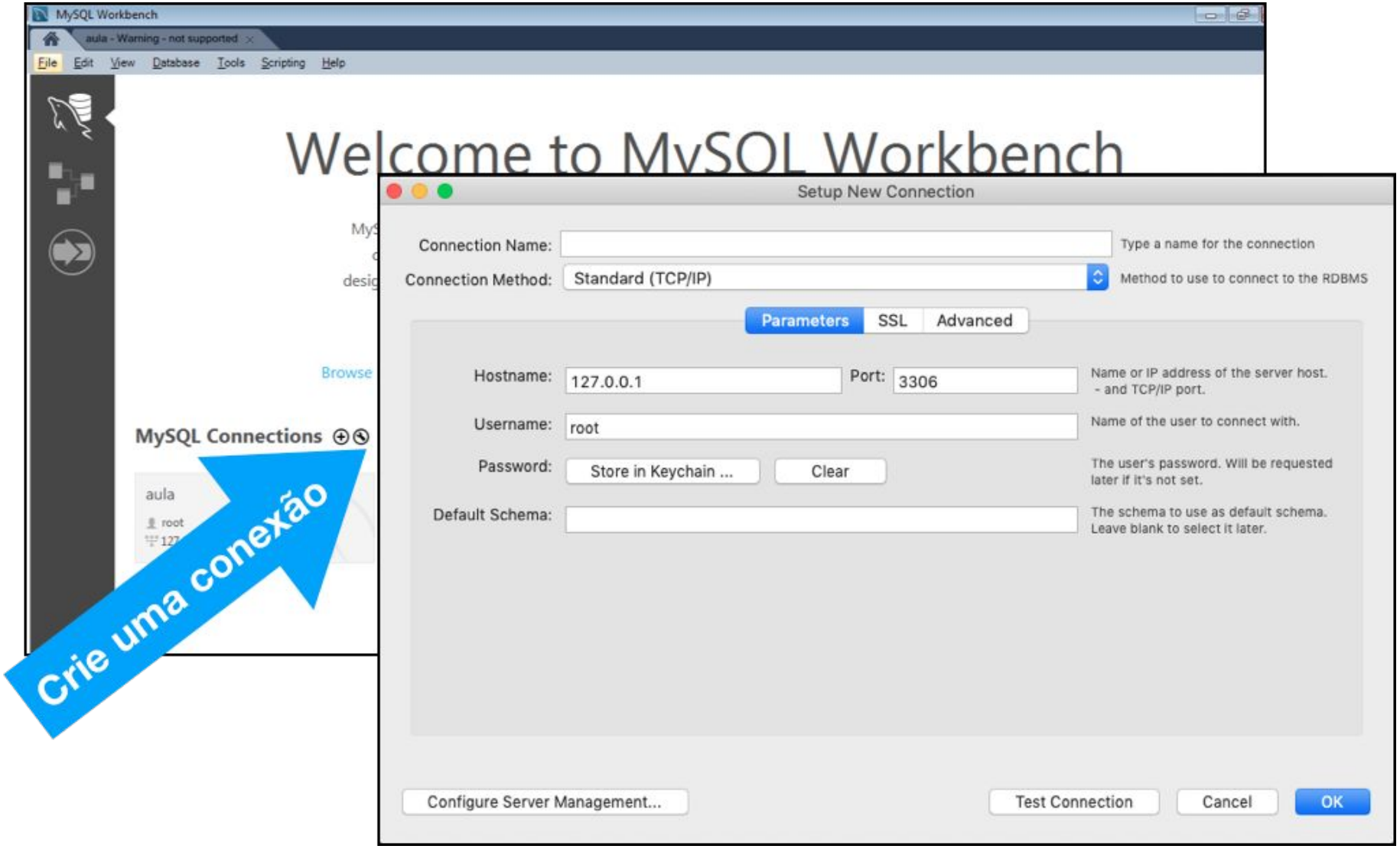
Quando não usar SGBD?

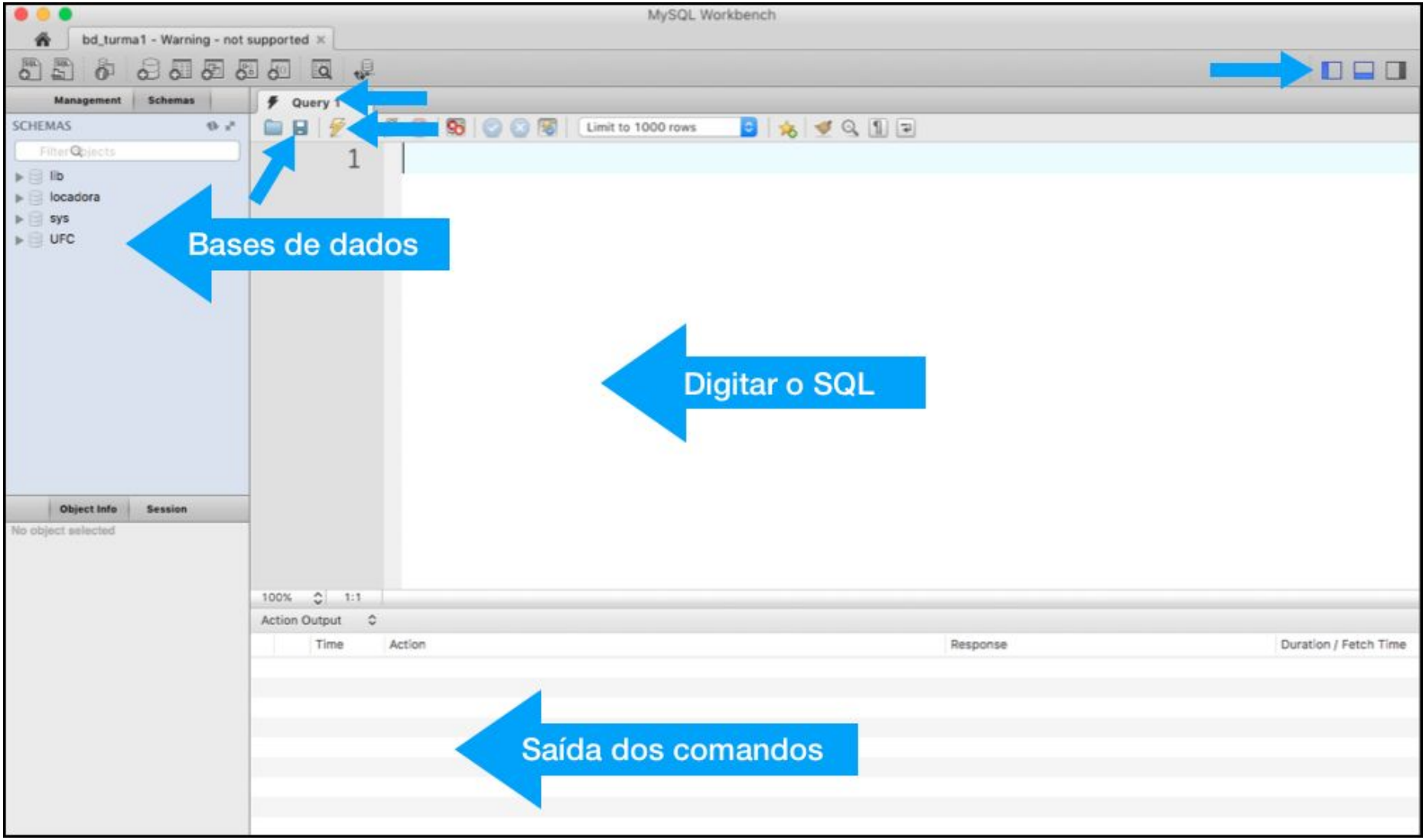
- Aplicações simples com poucas mudanças
- Sistemas embarcados com capacidade de armazenamento limitada
- Nenhum acesso de múltiplos usuários

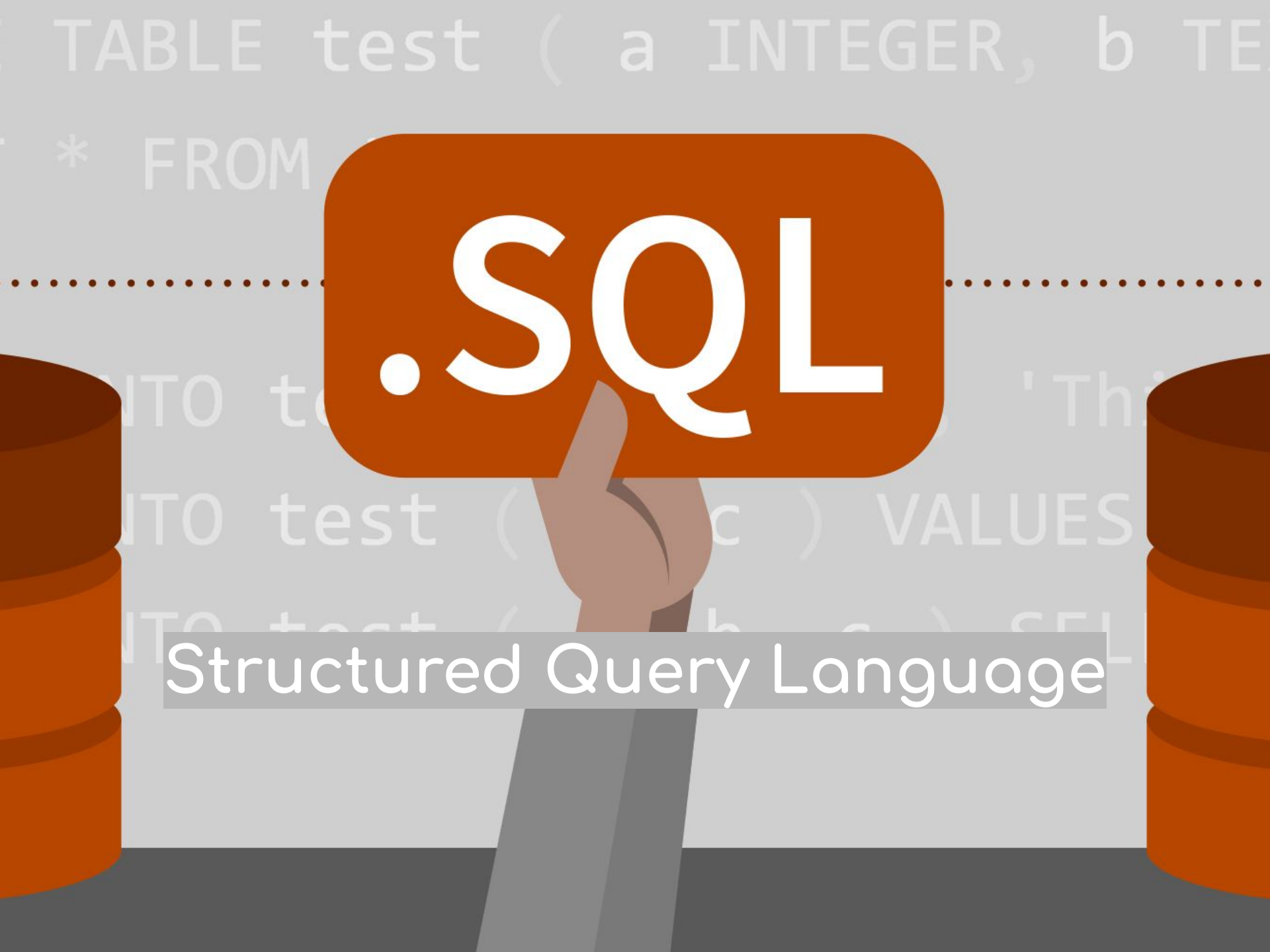
Arquitetura cliente-servidor



MySQL Workbench (cliente)




MySQL Workbench
bd_turma1 - Warning - not supported
Management
Schemas
SCHEMAS
lib
locadora
sys
UFC
Query 1
Limit to 1000 rows
1
Bases de dados
Digitar o SQL
Object Info
Session
No object selected
100%
1:1
Action Output
Time
Action
Response
Duration / Fetch Time
Saída dos comandos

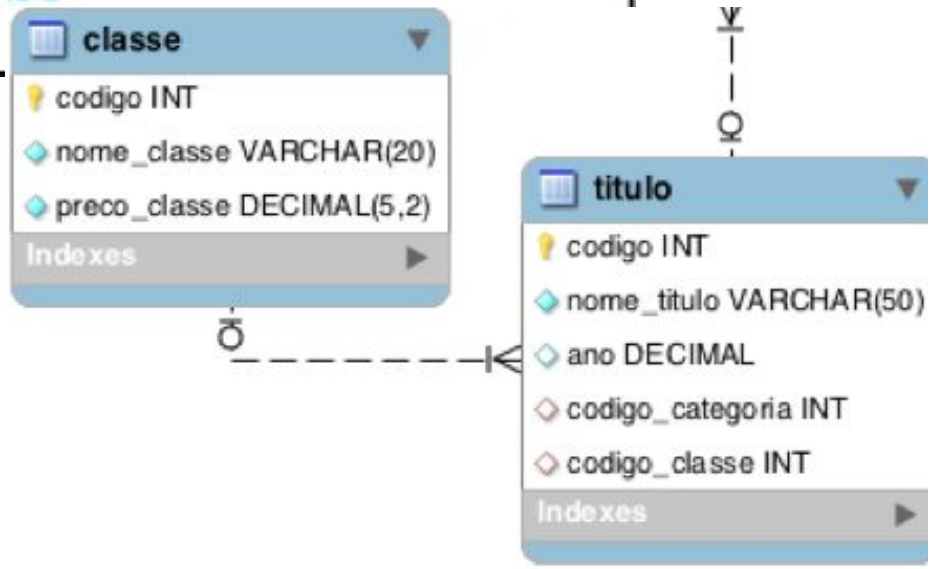

.SQL
Structured Query Language

Structured Query Language
(Linguagem de Consulta Estruturada)

- Não é só consulta!
- Possui recursos para **definição** da estrutura de dados, **modificação** e **exclusão** de dados, etc.
- Duas principais subdivisões:

Data Definition Language

```
CREATE TABLE titulo(
codigo                INT PRIMARY KEY,
nome_titulo           VARCHAR(50) NOT NULL,
ano                   NUMERIC(4),
codigo_categoria      INT,
codigo_classe         INT,
CONSTRAINT fk_titulo_categoria FOREIGN KEY (codigo_categoria)
    REFERENCES categoria(codigo)
    ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT fk_titulo_classe FOREIGN KEY (codigo)
    REFERENCES classe(codigo)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```



A **DDL** permite a criação (**CREATE**), remoção (**DROP**) e modificação (**ALTER**) das descrições das tabelas de uma base de dados.

Data Manipulation Language

INSERT

INSERT INTO projeto **VALUES** (‘ProductA’, 6, ‘Umirim’, 5);

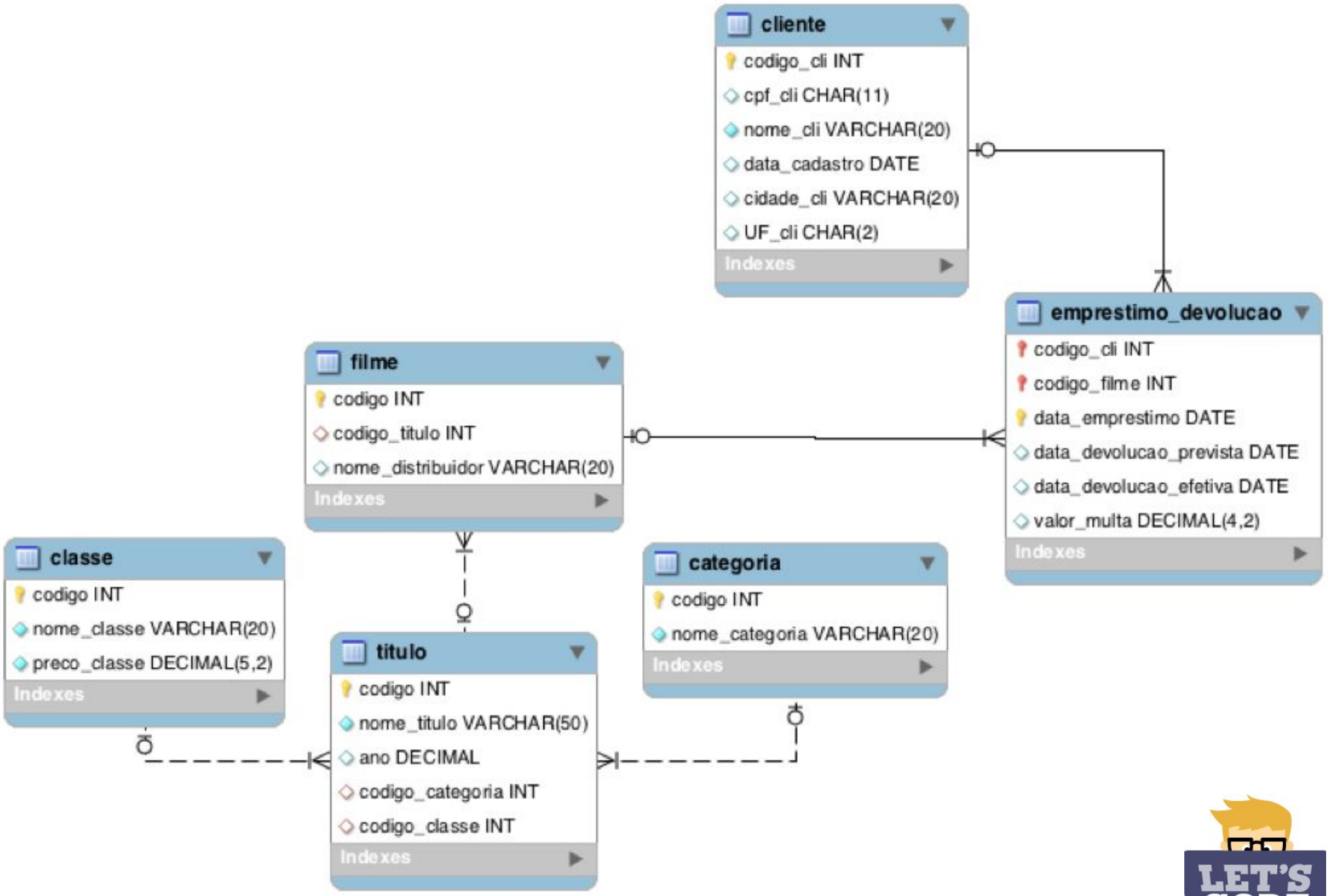
DELETE

DELETE FROM dependente
WHERE nome_dependente = ‘Alice’;

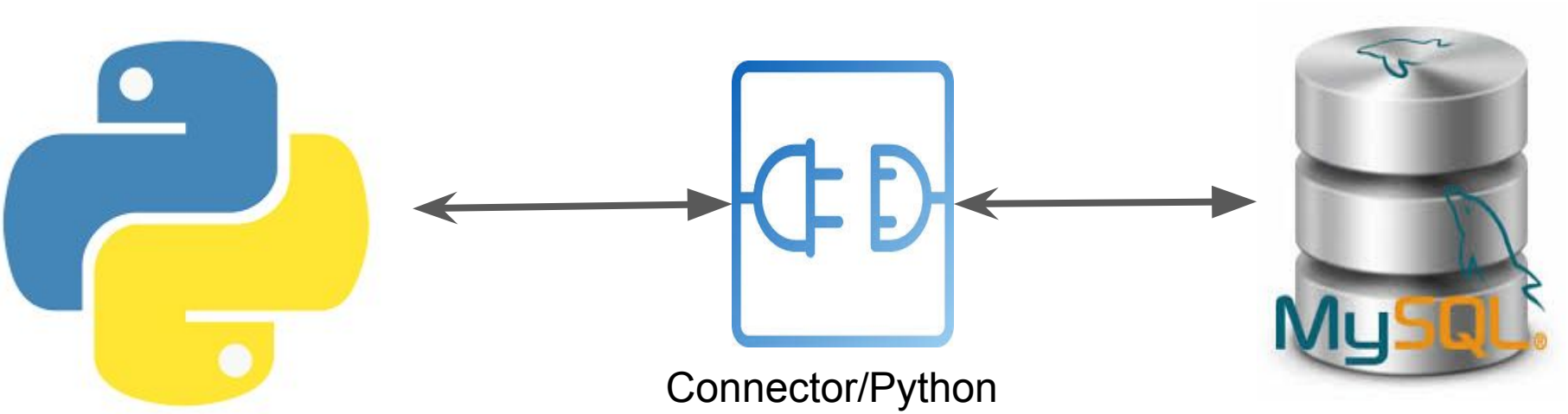
UPDATE

UPDATE projeto
SET plocalizacao = ‘Stafford’, dnum = 5
WHERE pnumero = 10;

```
31  SELECT data_nasc, endereco
32  FROM FUNCIONARIO
33  WHERE sexo = 'M' AND salario > 5000
```


cliente
codigo_cli INT
cpf_cli CHAR(11)
nome_cli VARCHAR(20)
data_cadastro DATE
cidade_cli VARCHAR(20)
UF_cli CHAR(2)
Indexes
emprestimo_devolucao
codigo_cli INT
codigo_filme INT
data_emprestimo DATE
data_devolucao_prevista DATE
data_devolucao_efetiva DATE
valor_multa DECIMAL(4,2)
Indexes
filme
codigo INT
codigo_titulo INT
nome_distribuidor VARCHAR(20)
Indexes
classe
codigo INT
nome_classe VARCHAR(20)
preco_classe DECIMAL(5,2)
Indexes
categoria
codigo INT
nome_categoria VARCHAR(20)
Indexes
titulo
codigo INT
nome_titulo VARCHAR(50)
ano DECIMAL
codigo_categoria INT
codigo_classe INT
Indexes

MySQL com Python



https://github.com/lapisco/pandas-mysql-practices


NoSQL

Banco de dados não relacional

- Padrão de armazenado alternativo ao modelo SQL (relacional).
- Oferecer uma maior **escalabilidade**
- Toda informação é armazenada em um registro
- Ex. do mongodb
  - db.clientes.save({_id: 1, fones: ["93254-8267", "93418-9592"]})

Relacional ou não relacional, qual usar?

- NOSQL (escalabilidade)
  - Muita informação e nem todas elas são tão importantes
- Relacional (confiabilidade e a consistência)

Obrigado pela atenção!