# Estrutura de dados
## Orientação a Objetos

## Métodos Extras

Profª. Drª. Alana Morais

# Fizeram o exercício do final da aula passada?

Praticar faz a diferença …

# Introdução

- Já vimos como funciona o *Duck Typing*
- Já entendemos que podemos substituir a herança por composição em alguns casos.

- Podemos fazer a nossa playlist se passar por outro tipo de objeto específico, sem precisarmos da herança; e não chamamos isso de polimorfismo, e sim de duck typing.

# Lembram do problema do Programa, Série e Filme?

- Digamos que o usuário montou sua lista de programas para o final de semana e deseja percorrê-la.
- Como ele faria isso?

```
for programa in playlist_fim_de_semana.listagem:
    print(programa)
```

# Lembram do problema do Programa, Série e Filme?

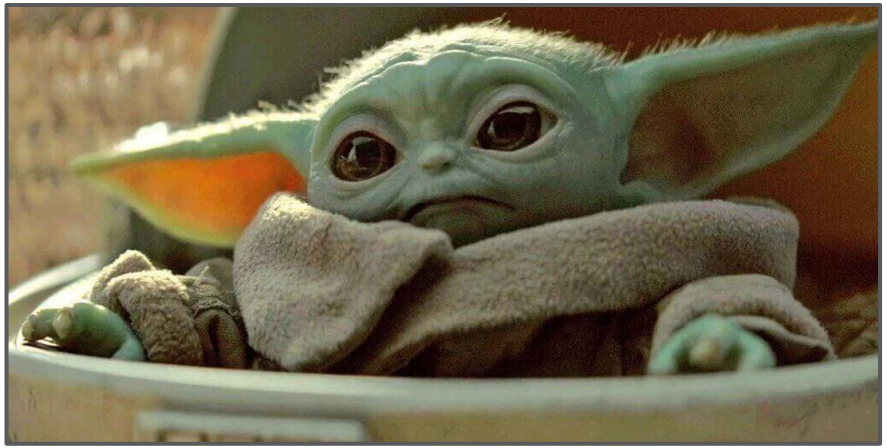
- Em outro ponto, também acessamos listagem para conseguirmos ver a contagem de itens dentro dela:

```
print (f'Tamanho do playlist: {len(playlist_fim_de_semanalistagem)}')
```

- O que não parece muito correto é que, anteriormente, estávamos usando Playlist como algo que pudéssemos iterar, e essa capacidade foi perdida.
- Se removermos .listagem no loop do for e rodarmos o código, receberemos o erro indicando que Playlist não é um objeto iterável.

# Questionamento ...

- Para nós, é desvantajoso perder esta funcionalidade listagem.
- De que maneira poderemos fazer isso, então, sem termos que recorrer à herança?



# *Dunder method*

- Há um método mágico - um dunder method - que, ao ser implementado, permite que a classe seja considerada um objeto iterável: o __getitem__().
- Este método define algo que é iterável e precisaremos receber um item para que este seja repassado à lista interna que estamos utilizando, isto é, **programas**.

# __getitem__()

```
def __getitem__(self, item):
    return self._programas[item]
```

# __getitem__()

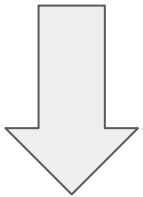
- Assim, repassamos um *item* para a nossa lista interna de programas e, se rodarmos o código, já conseguimos iterar por todos os itens da playlist.
- Além disso, conseguimos fazer outras operações, como o seguinte trecho:

```
print(vingadores in playlist_fim_de_semana)
```

# __len__()

- E se eu quisesse ver o tamanho da lista sem o elemento listagem:

```
print (f'Tamanho do playlist: {len(playlist_fim_de_semanalistagem)}')
```



```
print (f'Tamanho do playlist: {len(playlist_fim_de_semana)}')
```

- Como isso poderia funcionar?

# __len__()

- Ou seja, não existe len() em Playlist.
- Isto porque há um dunder method que poderá ser implementado para que a lista se comporte como um sized, uma ideia de algo que possui tamanho, e que então precisará implementar um __len__() para que o len() externo possa funcionar em nossa classe.

```
def __len__(self):
  return len(self._programas)
```

# Python Data Model

- No Python Data Model, todo objeto em Python pode se comportar de forma a ser compatível e mais próximo à linguagem, e de toda a ideia idiomática dela.
- O len() do Python, por exemplo, se diferencia um pouco de outras linguagens.
- Chamamos o objeto utilizando parênteses quando queremos inicializá-lo, junto ao nome da classe.
- O que define isso também é um método com dois underscores, que faz parte deste Data Model.

# Python Data Model

| Para quê? | Método |
| --- | --- |
| Inicialização | `__init__` |
| Representação | `__str__`, `__repr__` |
| Container, sequência | `__contains__`, `__iter__`, `__len__`, `__getitem__` |
| Numéricos | `__add__`, `__sub__`, `__mul__`, `__mod__` |

# Python Data Model

| Para quê? | Como? |
| --- | --- |
| **Inicialização** | obj = Novo() |
| **Representação** | print(obj), str(obj), repr(obj) |
| **Container, sequência** | len(obj), item in obj, for i in obj, obj[2:3] |
| **Numéricos** | obj + outro_obj, obj * obj |

# Python Data Model

- O __repr__() é utilizado para demonstrar como o objeto foi criado, útil mais para o compilador do Python do que para o usuário final.
- Dentre os métodos que servem para sequências - que são contêineres, para iterações -, há o __contains__(), que também fará o in funcionar.
- Também é possível mudarmos a forma como ele funciona, e implementá-lo de forma mais performática.
- O __iter__() define protocolos de iteração, então, estamos criando o iterador a ser retornado.
- O __len__(), por sua vez, retorna o tamanho da lista, e o __getitem__(), que também já vimos, nos ajuda a percorrermos a lista e pegarmos um item específico dela.

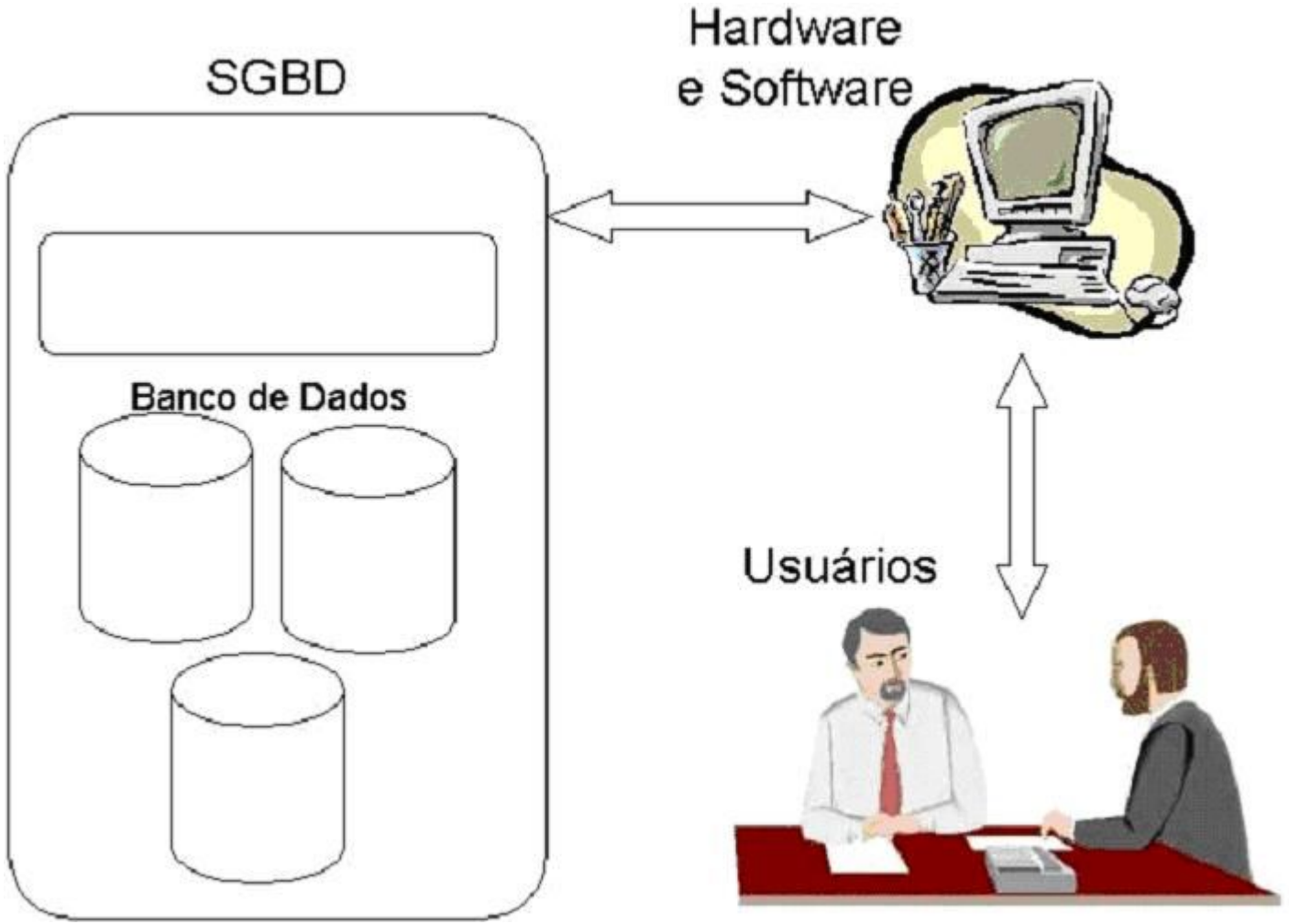
# Banco de Dados

# Por que usar Bancos de Dados?

- Muitos sistemas precisam manter as informações com as quais eles trabalham para permitir consultas futuras, geração de relatórios ou possíveis alterações nas informações.
- Para que esses dados sejam mantidos para sempre, esses sistemas geralmente guardam essas informações em um banco de dados, que as mantém de forma organizada e prontas para consultas.
- Os objetivos de um sistema de banco de dados são o de isolar o usuário dos detalhes internos do banco de dados (promover a abstração de dados) e promover a independência dos dados em relação às aplicações.

# SGBDs

- Já um sistema de gerenciamento de banco de dados (SGBD) é um *software* que possui recursos capazes de manipular as informações do banco de dados e interagir com o usuário.
    - Exemplos de SGBDs são: Oracle, SQL Server, DB2, PostgreSQL, MySQL, etc.
- A maioria dos bancos de dados comerciais são os chamados relacionais (conceito de Entidade-Relacionamento), que é uma forma de trabalhar e pensar diferente ao paradigma orientado a objetos.

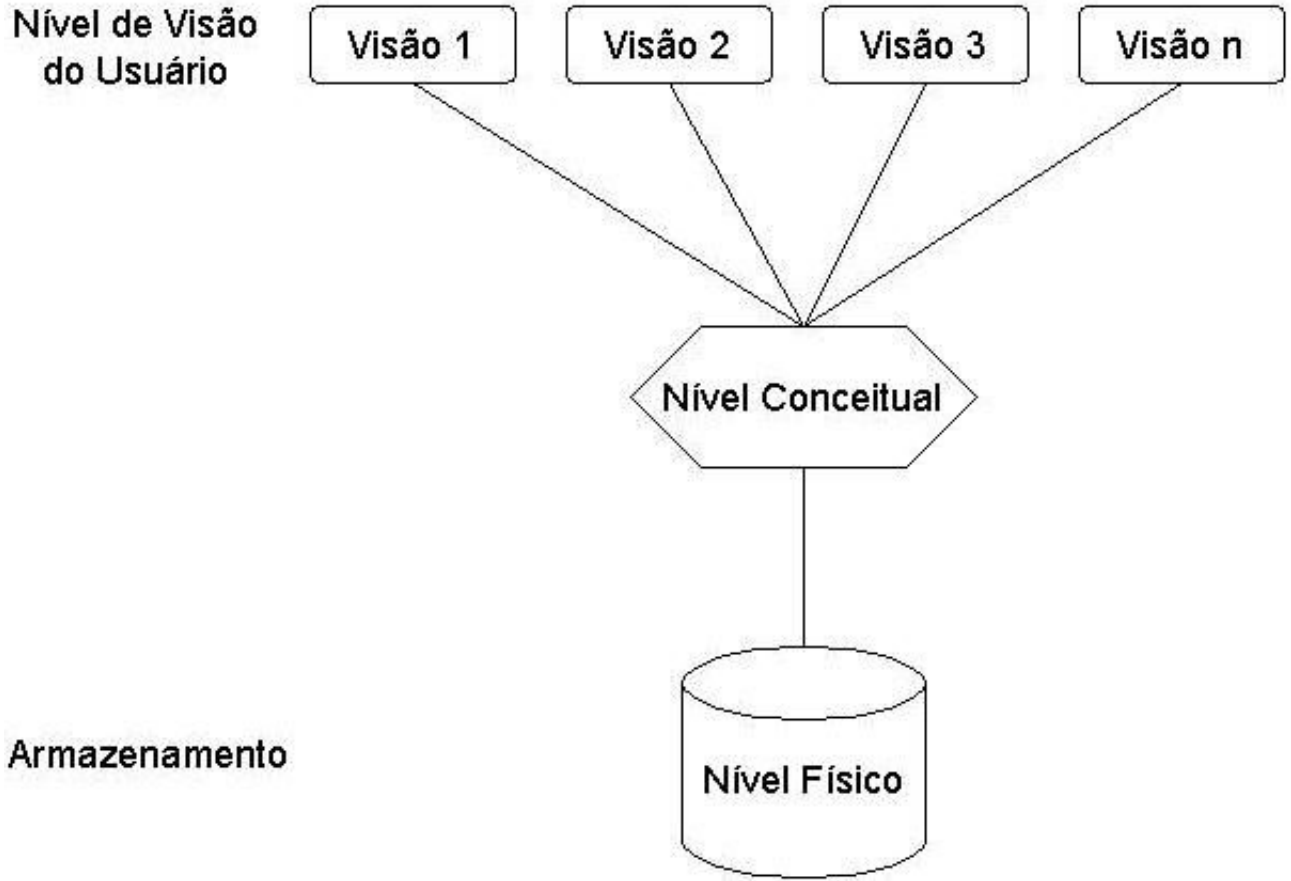

SGBD
Banco de Dados
Hardware
e Software
Usuários

# Bancos de Dados

O sistema de banco de dados deve garantir uma visão totalmente abstrata do banco de dados para o usuário

Esta abstração se dá em três níveis:

- Nível de visão do usuário: as partes do banco de dados que o usuário tem acesso de acordo com sua necessidade individual ou de um grupo de usuários;
- Nível conceitual: define quais os dados que estão armazenados e qual o relacionamento entre eles;
- Nível físico: é o nível mais baixo de abstração, em que define efetivamente de que maneira os dados estão armazenados.

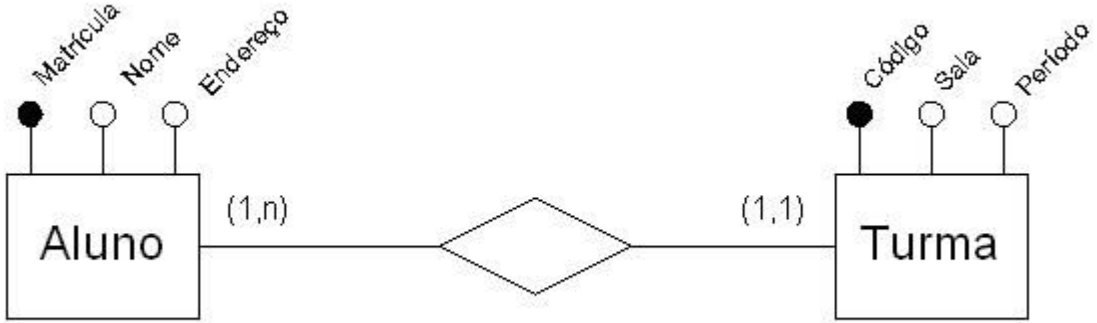

Nível de Visão
do Usuário
Visão 1
Visão 2
Visão 3
Visão n
Nível Conceitual
Armazenamento
Nível Físico

# Projeto de Banco de Dados

- Todo bom sistema de banco de dados deve apresentar um projeto
- O projeto de banco de dados se dá em duas fases:
  - Modelagem conceitual;
  - Projeto lógico.
- Estas duas etapas se referem a um sistema de banco de dados ainda não implementado.
  - Para os casos em que o banco de dados já exista, mas é um sistema legado, o processo de projeto de banco de dados se dará por meio da utilização de uma técnica chamada de Engenharia Reversa.

# Modelo conceitual

- É a descrição do BD de maneira independente ao SGBD.
- Define quais os dados que aparecerão no Banco de Dados, mas sem se importar com a implementação que se dará de fato.
  - Uma das técnicas mais utilizadas é a abordagem entidade-relacionamento (ER), que é representada graficamente por meio do diagrama entidade-relacionamento (DER).



# Modelo lógico

- Descreve o BD no nível do SGBD, ou seja, depende do tipo particular de SGBD que será usado.
- O modelo lógico do BD relacional deve definir quais as tabelas e o nome das colunas que compõem estas tabelas (cada BD tem sua sintaxe de comandos).

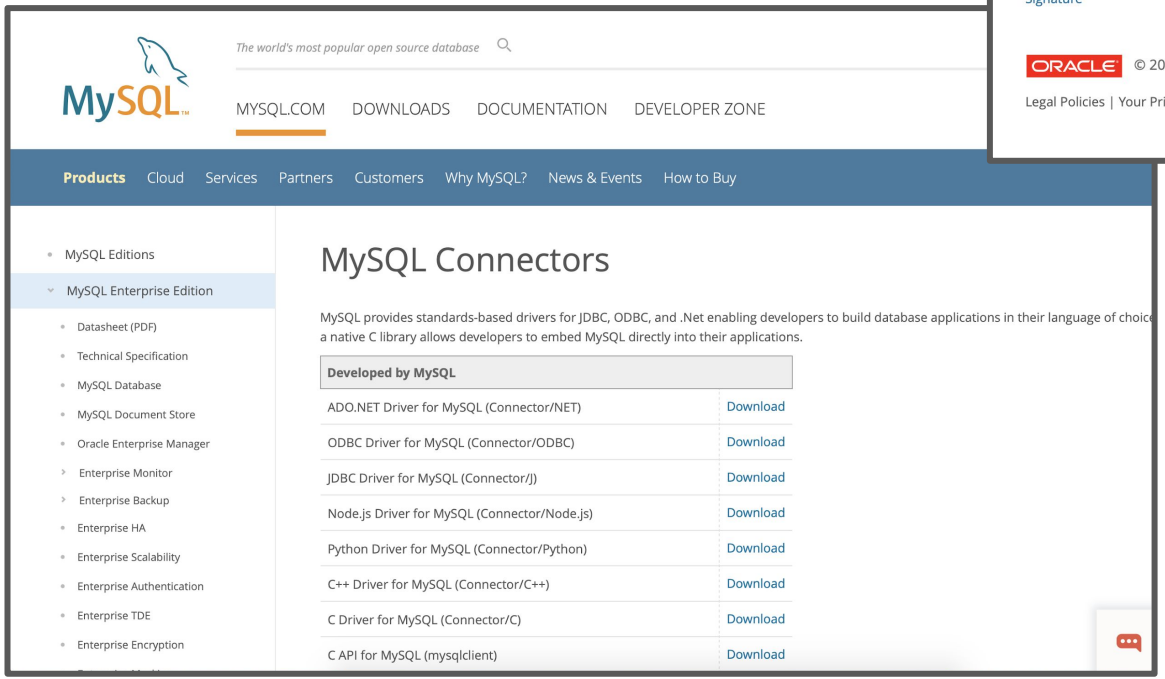
```
Aluno(mat_aluno, nome, endereco)
Turma (cod_turma, sala, periodo)
```

# MySQL

- Banco de dados que usaremos na presente aula.
- É um dos mais importantes bancos de dados relacionais, e é gratuito, além de ter uma instalação fácil para todos os sistemas operacionais.
- É importante entender alguns conceitos básicos sobre Banco de dados antes de começar.
- Geralmente roda na porta 3306
    - Cuidado para não esquecer a senha de root na instalação

# Driver mysql e Python

https://www.mysql.com/products/connector/

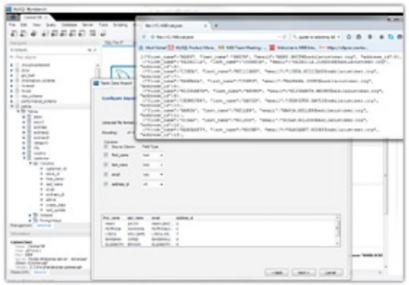



- MySQL Editions
- MySQL Enterprise Edition
  - Datasheet (PDF)
  - Technical Specification
  - MySQL Database
  - MySQL Document Store
  - Oracle Enterprise Manager
  - Enterprise Monitor
  - Enterprise Backup
  - Enterprise HA
  - Enterprise Scalability
  - Enterprise Authentication
  - Enterprise TDE

# MySQL Workbench

## Enhanced Data Migration

Download Now »

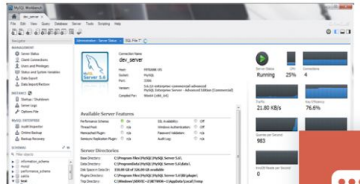


MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more. MySQL Workbench is available on Windows, Linux and Mac OS X.

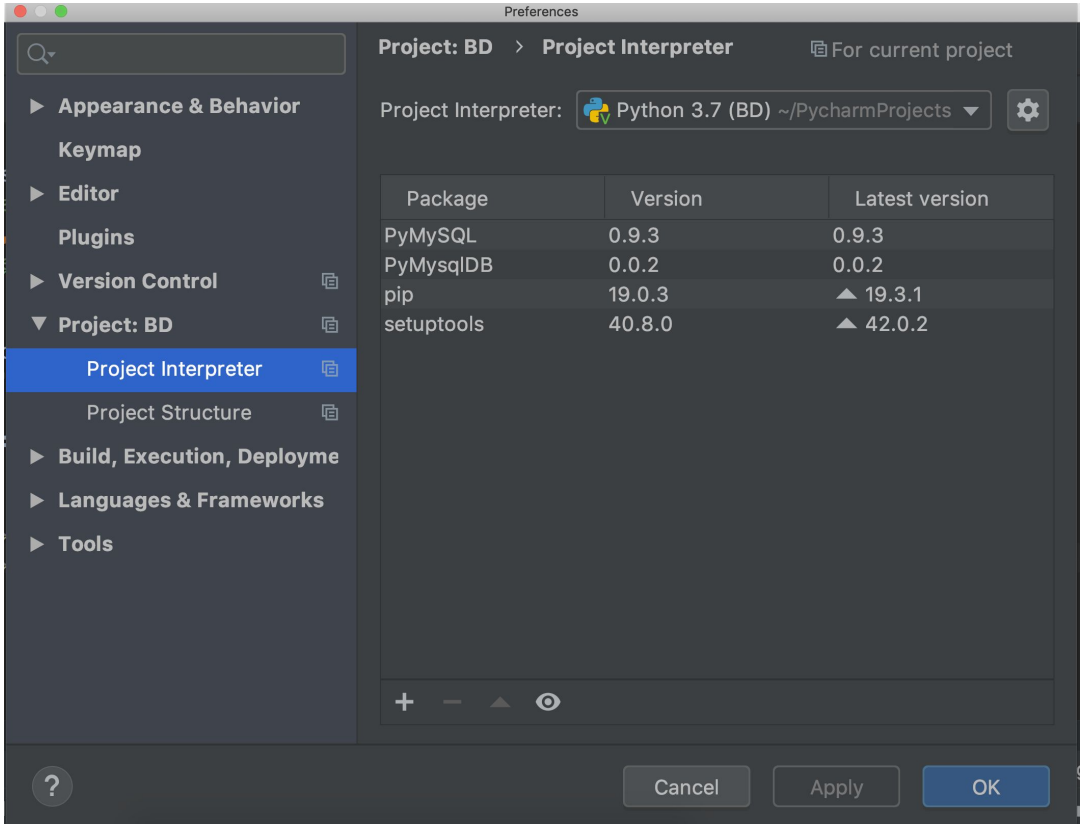
## Design

MySQL Workbench enables a DBA, developer, or data architect to visually design, model, generate, and manage databases. It includes everything a data modeler needs for creating complex ER models, forward and reverse engineering, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort.

MySQL Workbench Home

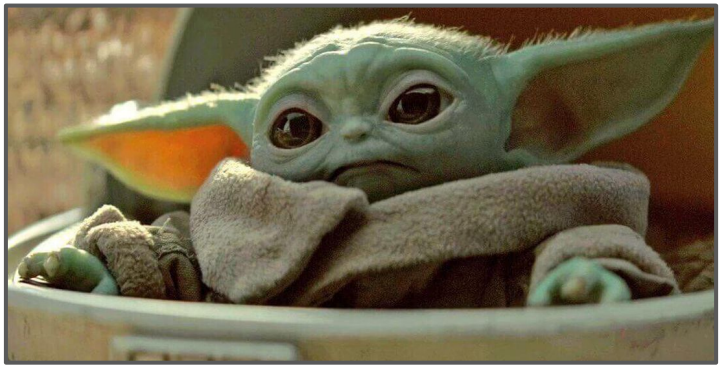


# Pycharm como IDE no exemplo



# Vamos criar nosso banco?

- Primeiro planeje o banco
- Depois crie o banco no SGBD (depois você conseguirá criar um script que fará isso automaticamente)
- Depois alimente e use seu banco



# É possível fazer um série de operações via query

- Criar um banco

```
CREATE DATABASE nome-do-banco;
```

- Ver os bancos criados no seu SGBD

```
SHOW DATABASES;
```

- Para usar o banco no SGBD, é preciso chamá-lo:

```
USE nome-do-banco;
```

# É possível fazer um série de operações via query

- Selecionar os resultados

```
SELECT * FROM clientes;

SELECT id_cliente, nome_empresa FROM clientes;
```

- Inserir novos registros

```
INSERT INTO tbl_editoras (Nome_Editora) VALUES ('O´Reilly');
```

- Remover elementos

```
DELETE FROM clientes WHERE nomeEmpresa = 'GameCorp';
```

# Como eu integro isso com meu programa em Python?

# Inicialmente conecte com seu banco de dados

```
# Conectar o banco de dados
connection = pymysql.connect(host='localhost',
                             user='user',
                             password='passwd',
                             db='db')
```

# Quem quiser entender melhor ….

Precisa estudar um conceito chamado Normalização:

https://pt.stackoverflow.com/questions/151323/o-que-é-normalização-de-banco-de-dados

# Comandos MySQL

- CRUD
  - INSERT INTO usuarios (nome, login, senha)VALUES ('João Carlos', 'joca', 'abc123');
  - SELECT login, senha FROM usuarios;
  - UPDATE usuarios SET senha = '123456' WHERE id_usuario = 1
  - DELETE FROM usuarios WHERE login = 'joca'

Dúvidas?
alanamm.prof@gmail.com