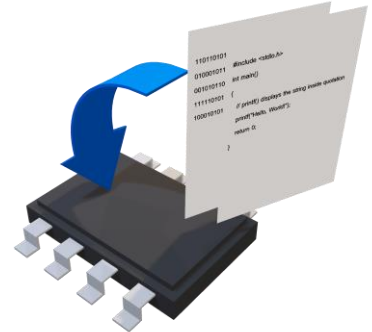




BANCO DE DADOS PARA ENGENHARIA

Prof. Armando L. Keller



Banco de dados e P00

Banco de dados e POO

A programação orientada a objetos é um dos principais paradigmas de programação, muito comummente utilizada em projetos de alta complexidade.

O que faz com que este paradigma seja adotado são as suas principais características (Abstração, encapsulamento, herança e polimorfismo)

Banco de dados e POO

Algumas linguagens de programação com suporte a orientação a objetos:

- C++
- C#
- Python
- Java
- Javascript
- Ruby
- PHP
- ABAP
- Coldfusion
- Dart
- Kotlin
- Object Pascal
- Rust
- TypeScript
- Swift

Banco de dados e POO

Assim como foi feita uma modelagem dos dados para a criação do banco de dados, é possível modelar estes dados como classes para serem utilizadas dentro de um *software* de maneira transparente.

Banco de dados e POO

Este nível de abstração permite que uma determinada pessoa ou grupo de pessoas realize a integração com o SGBD, enquanto outras pessoas podem simplesmente importar um módulo e utilizar as classes sem necessariamente saber como o banco de dados foi montado, onde está sendo armazenado, quais tabelas existem.

Banco de dados e POO

Esta integração pode ser realizada de diversas maneiras, uma delas é com a utilização de bibliotecas de ORM (*Object Relational Mapping*), que realizam o mapeamento dos objetos relacionais.

O ORM pode inclusive criar um banco de dados a partir das estruturas das classes.

Banco de dados e POO

Algumas vantagens do uso de ORM são:

- Não precisar escrever todas as consultas SQL da aplicação manualmente.
- Interoperabilidade entre diferentes SGBD's que podem vir a ter sintaxes SQL diferentes.

Banco de dados e POO

No entanto a grande maioria das bibliotecas ORM possuem limitações como:

- Não possuir herança
- Não possuir polimorfismo
- Não permitir relações N para N entre duas tabelas

Banco de dados e POO

Alguns exemplos de ORM para diferentes linguagens de programação:

- Peewee (Python)
- Pony (Python)
- SQLAlchemy (Python)
- Hibernate (Java)
- OJB (Java)
- Eloquent ORM (PHP)

Banco de dados e POO

Além da utilização de ORMs, é possível implementar as próprias classes, integrando as consultas SQL.

Este procedimento é um pouco mais demorado de ser implementado, no entanto possui menos restrições e alta possibilidade de customização.

Tornando o seu uso final mais simples.

Integração com o SGBD

Integração com o SGBD

Os exemplos demonstrados aqui serão desenvolvidos na linguagem Python com o SGBD MySQL, podendo ser implementados dentro do ambiente de testes fornecido (maquina virtual do Linux).

No entanto os exercícios poderão ser realizados na linguagem de programação que o aluno possuir maior familiaridade.

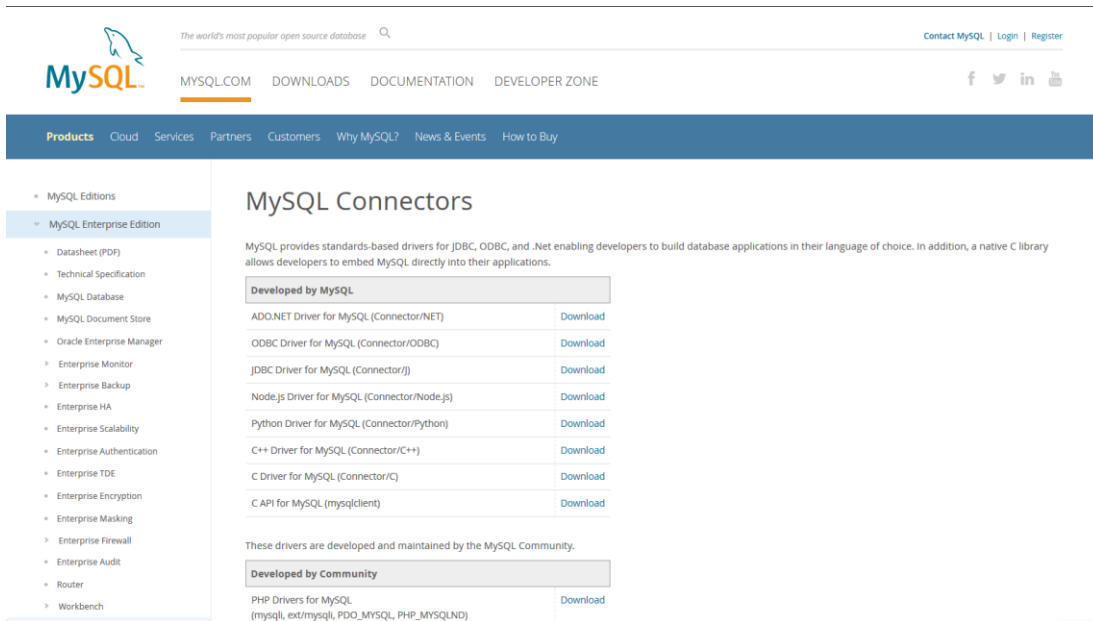
Integração com o SGBD

Para realizar a conexão com um SGBD muitas vezes é necessário a utilização de um conector, fornecido pelo desenvolvedor do SGBD.

Para o MySQL os conectores podem ser obtidos diretamente do site oficial do MySQL, possuindo também versões desenvolvidas pela comunidade.

Integração com o SGBD

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



The screenshot shows the MySQL website's 'MySQL Connectors' page. The header includes the MySQL logo, the tagline 'The world's most popular open source database', and links for 'Contact MySQL', 'Login', and 'Register'. A navigation bar contains links for 'Products', 'Cloud', 'Services', 'Partners', 'Customers', 'Why MySQL?', 'News & Events', and 'How to Buy'. A left sidebar lists various MySQL Editions and Enterprise Edition features. The main content area is titled 'MySQL Connectors' and explains that MySQL provides standards-based drivers for JDBC, ODBC, and .Net. It also mentions a native C library. Below this, there are two sections: 'Developed by MySQL' and 'Developed by Community'. The 'Developed by MySQL' section lists several drivers with 'Download' links: ADO.NET Driver for MySQL (Connector/NET), ODBC Driver for MySQL (Connector/ODBC), JDBC Driver for MySQL (Connector/J), Node.js Driver for MySQL (Connector/Node.js), Python Driver for MySQL (Connector/Python), C++ Driver for MySQL (Connector/C++), C Driver for MySQL (Connector/C), and C API for MySQL (mysqlclient). The 'Developed by Community' section lists 'PHP Drivers for MySQL (mysql, ext/mysql, PDO_MYSQL, PHP_MYSQLND)' with a 'Download' link. At the bottom right, there is a large blue logo for UNISINOS with the text 'UNISINOS DESAFIE O AMANHÃ.'

BANCO DE DADOS PARA ENGENHARIA

Prof. Armando Leopoldo Keller

Integração com o SGBD

Para a implementação dos exemplos é necessário realizar o download do conector para Python.

Developed by MySQL	
ADO.NET Driver for MySQL (Connector/NET)	Download
ODBC Driver for MySQL (Connector/ODBC)	Download
JDBC Driver for MySQL (Connector/J)	Download
Node.js Driver for MySQL (Connector/Node.js)	Download
Python Driver for MySQL (Connector/Python)	Download
C++ Driver for MySQL (Connector/C++)	Download
C Driver for MySQL (Connector/C)	Download
C API for MySQL (mysqlclient)	Download

Integração com o SGBD

Selecione a versão adequada ao seu sistema operacional
(no caso da maquina virtual Ubuntu Linux)

General Availability (GA) Releases

Connector/Python 8.0.18

Select Operating System:
Ubuntu Linux

Select OS Version:
All
All
Ubuntu Linux 19.10 (Architecture Independent)
Ubuntu Linux 19.10 (x86, 64-bit)
Ubuntu Linux 18.04 (Architecture Independent)
Ubuntu Linux 18.04 (x86, 32-bit)
Ubuntu Linux 18.04 (x86, 64-bit)
Ubuntu Linux 16.04 (x86, 32-bit)
Ubuntu Linux 16.04 (x86, 64-bit)
Ubuntu Linux 16.04 (Architecture Independent)

Looking for previous GA versions?

8.0.18	187.6K	Download
MD5: c6a976b24f768f4bcb0f5310d57bcc2c Signature		
8.0.18	187.1K	Download
MD5: 651de8c2edab7ec112d46dcccffb4dac2 Signature		
8.0.18	1.8M	Download
MD5: 0b705251920103cceb36ed18cbfe9460 Signature		

(mysql-connector-python-cext_8.0.18-1ubuntu19.10_amd64.deb)

Integração com o SGBD

Após a instalação do conector é necessário instalar o pacote do conector do mysql para o python, para isto será utilizado o gerenciador de pacotes do python.

Para isto, no terminal insira o seguinte comando:

```
pip3 install mysql-connector
```

Integração com o SGBD

Em Python as é recomendado que as bibliotecas que façam a interface com os SGBDs sigam a PEP 249 (Python Database API Specification v2.0) disponível em <https://www.python.org/dev/peps/pep-0249/>

Integração com o SGBD

A instalação pode ser verificada realizando a importação do modulo.

```
import mysql.connector
```

Integração com o SGBD

A conexão será realizada através do conector, que recebe como parâmetros os dados de conexão.

Exemplo:

```
import mysql.connector
```

```
db = mysql.connector.connect(  
    host = "127.0.0.1",  
    user = "usuario",  
    passwd = "#Senha123"  
)
```

Integração com o SGBD

Uma vez que a conexão foi estabelecida, as consultas serão executadas através de um cursor.

Como a conexão foi armazenada na variável db, podemos gerar o cursor a partir dela.

```
cursor = db.cursor()
```

Integração com o SGBD

Uma vez que a conexão foi estabelecida, as consultas serão executadas através de um cursor.

Como a conexão foi armazenada na variável db, podemos gerar o cursor a partir dela.

```
cursor = db.cursor()
```

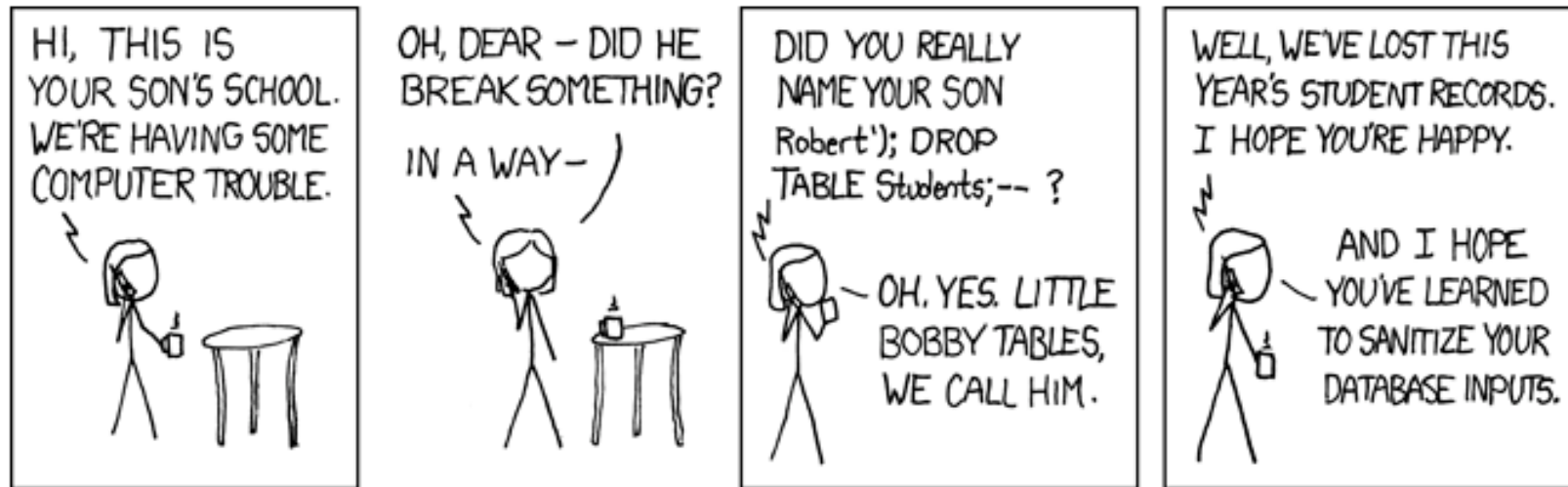
Integração com o SGBD

Os comandos SQL serão executados na função execute, que pertence ao objeto cursor.

Um detalhe importante de ser lembrado é nunca formatar a string da consulta com dados provenientes de informações inseridas fora do código, como em campos disponíveis para os usuários do programa.

Integração com o SGBD

Formatar as strings de consulta com dados do usuário sem o devido tratamento deixa o banco vulnerável a ataques de SQL injection.



https://imgs.xkcd.com/comics/exploits_of_a_mom.png

Integração com o SGBD

Para inserir informações de forma dinâmica em uma consulta, deve-se passar os valores a serem inseridos em uma tupla.

Exemplo:

```
cursor.execute("SELECT * FROM TABELA WHERE id=%s", (42, ))
```

Integração com o SGBD

Para que as consultas que realizam algum tipo de mudança no banco de dados tenham seus valores definitivos salvos no banco, deve-se chamar a função commit disponível no objeto de conexão.

Exemplo:

```
cursor.execute("INSERT INTO Tabela (campo)  
VALUES (%s)",(valor,))  
banco.commit()  
Id_inserido = cursor.lastrowid
```

Criando as classes

Criando as classes

As estratégias para a criação de classes que realizarão a integração com o SGBD dependerão do tipo de aplicação, da quantidade de dados, e da frequência das consultas.

No geral é interessante que a classe seja vinculada com cada um dos registros do banco. Caso estes registros utilizem chaves estrangeiras este relacionamento pode ser modelado de diversas maneiras que devem ser avaliadas caso a caso.

Exemplo para a classe Marinheiros (Moodle)

Exercícios

1 – Com base no exemplo apresentado, implemente (na linguagem de sua preferência) as classes para representar as reservas e os barcos do caso dos marinheiros. Caso utilize outra linguagem, implementar também a classe Marinheiros novamente.

2 - Implemente um software utilizando orientação a objetos que permita consultar informações sobre os marinheiros, barcos, e reservas.

Exercícios

3 - Para o software implementado no exercício 2, acrescente as seguintes funcionalidades:

- Inclusão de novos dados (marinheiros, barcos e reservas), mantendo as restrições das reservas (não permitir que um barco seja retirado se este ainda não tiver sido devolvido).
- Remoção e alteração dos dados
- Devolução de barcos

OBRIGADO.

