Fatec
Rio Claro

Linguagem de Programação II

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

SQL Alchemy

SQL Alchemy

Fatec
Rio Claro

SQLAlchemy é uma biblioteca usada para interagir com uma ampla variedade de bancos de dados. Permite criar modelos de dados e consultas de uma maneira que se assemelha a classes e instruções Python comuns.

Criada por Mike Bayer em 2005, a SQLAlchemy é usada por muitas empresas, grandes e pequenas, e é considerada por muitos como a maneira definitiva de trabalhar com bancos de dados relacionais em Python.

SQL Alchemy

Fatec
Rio Claro

Ela pode ser usada para conectar-se aos bancos de dados mais comuns, como PostgreSQL, MySQL, SQLite, Oracle e muitos outros.

Ela também oferece uma maneira de adicionar suporte a outros bancos de dados relacionais.

O Amazon Redshift, que usa um dialeto personalizado do PostgreSQL, é um ótimo exemplo de suporte a bancos de dados adicionado pela comunidade.

SQL Alchemy

Fatec
Rio Claro

Vantagens do SQL Alchemy

Fatec
Rio Claro

O principal motivo para usar o SQLAlchemy é abstrair seu código do banco de dados subjacente e das peculiaridades SQL associadas a ele.

Ele utiliza instruções e tipos comuns poderosos para garantir que suas instruções SQL sejam criadas de forma eficiente e adequada para cada tipo de banco de dados e fornecedor, sem que você precise se preocupar com isso.

Isso facilita a migração de lógica do Oracle para o PostgreSQL ou de um banco de dados de aplicativo para um data warehouse.

Também ajuda a garantir que a entrada do banco de dados seja sanitizada antes de ser enviada ao banco de dados. Isso evita problemas comuns, como ataques de injeção de SQL.

Vantagens do SQL Alchemy

Fatec Rio Claro

O SQLAlchemy também oferece bastante flexibilidade, fornecendo dois modos principais de uso: Linguagem de Expressão SQL (comumente chamada de Core) e ORM.

Esses modos podem ser usados separadamente ou em conjunto, dependendo da sua preferência e das necessidades da sua aplicação.

SQLAlchemy Core e Linguagem SQL

Fatec Rio Claro

A Linguagem de Expressão SQL é uma forma Pythonica de representar instruções e expressões SQL comuns, sendo apenas uma leve abstração da linguagem SQL típica.

Seu foco é o esquema do banco de dados em si; no entanto, é padronizada de forma a fornecer uma linguagem consistente para um grande número de bancos de dados de back-end.

A Linguagem de Expressão SQL também atua como base para o ORM SQLAlchemy.

ORM SQLAlchemy

Fatec
Rio Claro

ORM SQLAlchemy é semelhante a muitos outros mapeadores relacionais de objetos (ORMs) que você pode ter encontrado em outras linguagens. Ele se concentra no modelo de domínio da aplicação e utiliza o padrão Unidade de Trabalho para manter o estado do objeto.

Ele também fornece uma abstração de alto nível sobre a Linguagem de Expressão SQL, permitindo que o usuário trabalhe de forma mais idiomática. Você pode combinar o uso do ORM com a Linguagem de Expressão SQL para criar aplicações muito poderosas.

O ORM utiliza um sistema declarativo semelhante aos sistemas de registro ativo usados por muitos outros ORMs, como o encontrado em Ruby on Rails.

Escolhendo entre SQLAlchemy Core e ORM

Fatec
Rio Claro

Antes de começar a construir aplicações com SQLAlchemy, você precisará decidir se usará principalmente o ORM ou o Core.

A escolha de usar SQLAlchemy Core ou ORM como a camada de acesso a dados dominante para uma aplicação geralmente se resume a alguns fatores e preferências pessoais.

Os dois modos usam sintaxe ligeiramente diferente, mas a maior diferença entre Core e ORM é a visão dos dados como esquema ou objetos de negócio.

Escolhendo entre SQLAlchemy Core e ORM

Fatec
Rio Claro

O SQLAlchemy Core possui uma visão centrada em esquema, que, assim como o SQL tradicional, é focada em tabelas, chaves e estruturas de índice.

Se destaca em data warehouse, relatórios, análises e outros cenários onde ser capaz de controlar rigorosamente a consulta ou operar em dados não modelados é útil.

O forte pool de conexões com o banco de dados e as otimizações do conjunto de resultados são perfeitamente adequados para lidar com grandes quantidades de dados, mesmo em múltiplos bancos de dados.

Escolhendo entre SQLAlchemy Core e ORM

Fatec Rio Claro

No entanto, se você pretende se concentrar mais em um design orientado a domínio, o ORM encapsulará grande parte do esquema e da estrutura subjacentes em metadados e objetos de negócio.

Esse encapsulamento pode facilitar a criação de interações com o banco de dados que se assemelham mais a um código Python comum.

Escolhendo entre SQLAlchemy Core e ORM

Fatec Rio Claro

A maioria das aplicações comuns se presta a ser modelada dessa maneira. Também pode ser uma maneira altamente eficaz de injetar design orientado a domínio em uma aplicação legada ou com instruções SQL brutas espalhadas por toda parte.

Os microsserviços também se beneficiam da abstração do banco de dados subjacente, permitindo que o desenvolvedor se concentre apenas no processo que está sendo implementado.

Escolhendo entre SQLAlchemy Core e ORM

Fatec Rio Claro

- Se você estiver trabalhando com um framework que já possui um ORM integrado, mas deseja adicionar relatórios mais poderosos, use o Core.
- Se você quiser visualizar seus dados em uma visão mais centrada no esquema (como usado em SQL), use o Core.
- Se você tiver dados para os quais objetos de negócios não são necessários, use o Core.
- Se você visualizar seus dados como objetos de negócios, use o ORM.
- Se você estiver construindo um protótipo rápido, use o ORM.
- Se você tiver uma combinação de necessidades que realmente poderia aproveitar tanto objetos de negócios quanto outros dados não relacionados ao domínio do problema, use ambos!

SQL Alchemy
Como instalar

Como instalar SQLAlchemy

Fatec
Rio Claro

O SQLAlchemy pode ser usado com Python 2.6, Python 3.3 e Pypy 2.1 ou superior. Recomendo usar o pip para executar a instalação com o comando pip install sqlalchemy. Vale ressaltar que ele também pode ser instalado com easy_install e distutils; no entanto, o pip é o método mais simples.

Durante a instalação, o SQLAlchemy tentará construir algumas extensões em C, que são utilizadas para tornar o trabalho com conjuntos de resultados mais rápido e mais eficiente em termos de memória.

Por padrão, o SQLAlchemy suporta SQLite3 sem drivers adicionais; no entanto, um driver de banco de dados adicional que usa a especificação padrão Python DBAPI (PEP-249) é necessário para se conectar a outros bancos de dados.

Como instalar SQLAlchemy

Fatec
Rio Claro

PostgreSQL

O Psycopg2 oferece amplo suporte para versões e recursos do PostgreSQL e pode ser instalado com *pip install psycopg2*.

MySQL

PyMySQL é minha biblioteca Python preferida para conexão com um servidor de banco de dados MySQL. Ela pode ser instalada com *pip install pymysql*. O suporte ao MySQL no SQLAlchemy requer a versão 4.1 do MySQL ou superior devido ao funcionamento das senhas antes dessa versão.

Outros

O SQLAlchemy também pode ser usado em conjunto com Drizzle, Firebird, Oracle, Sybase e Microsoft SQL Server. A comunidade também forneceu dialetos externos para muitos outros bancos de dados, como IBM DB2, Informix, Amazon Redshift, EXASolution, SAP SQL Anywhere, Monet e muitos outros.

SQL Alchemy
Como usar

Como usar SQLAlchemy

Fatec
Rio Claro

Para nos conectarmos a um banco de dados, precisamos criar um mecanismo (*engine*) SQLAlchemy, que cria uma interface comum com o banco de dados para executar instruções SQL. Ele faz isso encapsulando um conjunto de conexões de banco de dados e um dialeto SQL.

Uma conexão é uma string especialmente formatada que fornece:

- Tipo de banco de dados (Postgres, MySQL, etc.)
- Dialeto, a menos que seja o padrão para o tipo de banco de dados (Psycopg2, PyMySQL, etc.)
- Detalhes de autenticação opcionais (nome de usuário e senha)
- Localização do banco de dados (arquivo ou nome do host do servidor de banco de dados)
- Porta opcional do servidor de banco de dados
- Nome opcional do banco de dados

Como usar SQLAlchemy

Fatec
Rio Claro

```
from sqlalchemy import create_engine

engine = create_engine('sqlite:///cookies.db')

engine2 = create_engine('sqlite:///:memory:')

engine3 = create_engine('sqlite:////home/cookiemonster/cookies.db')

engine4 = create_engine('sqlite:///c:\\Users\\cookiemonster\\cookies.db')
```

Como usar SQLAlchemy

Fatec
Rio Claro

Para nos conectarmos a um banco de dados, precisamos criar um mecanismo (engine) SQLAlchemy, que cria uma interface comum com o banco de dados para executar instruções SQL. Ele faz isso encapsulando um conjunto de conexões de banco de dados e um dialeto SQL.

Uma conexão é uma string especialmente formatada que fornece:

- Tipo de banco de dados (Postgres, MySQL, etc.)
- Dialeto, a menos que seja o padrão para o tipo de banco de dados (Psycopg2, PyMySQL, etc.)
- Detalhes de autenticação opcionais (nome de usuário e senha)
- Localização do banco de dados (arquivo ou nome do host do servidor de banco de dados)
- Porta opcional do servidor de banco de dados
- Nome opcional do banco de dados

Prática

Fatec
Rio Claro

Observe os códigos

sqlite_teste.py

pg_teste.py

Como usar SQLAlchemy

Fatec
Rio Claro

Para fornecer acesso ao banco de dados subjacente, o SQLAlchemy precisa de uma representação das tabelas que devem estar presentes no banco de dados. Podemos fazer isso de três maneiras:

- Usando objetos *Table* definidos pelo usuário
- Usando classes declarativas que representam suas tabelas
- Inferindo-as a partir do banco de dados

Como usar SQLAlchemy

Fatec
Rio Claro

Os objetos Table contêm uma lista de colunas tipadas e seus atributos, que estão associados a um contêiner de metadados comum.

Existem quatro categorias de tipos que podemos usar dentro do SQLAlchemy:

- Genérico
- Padrão SQL
- Específico do fornecedor
- Definido pelo usuário

O SQLAlchemy define um grande número de tipos genéricos que são abstraídos dos tipos SQL suportados por cada banco de dados de back-end.

Todos esses tipos estão disponíveis no módulo **sqlalchemy.types** e, para sua conveniência, também estão disponíveis no módulo **sqlalchemy**.

Como usar SQLAlchemy

Fatec
Rio Claro

SQLAlchemy	Python	SQL
`BigInteger`	`int`	`BIGINT`
`Boolean`	`bool`	`BOOLEAN` or `SMALLINT`
`Date`	`datetime.date`	`DATE` (SQLite: `STRING`)
`DateTime`	`datetime.datetime`	`DATETIME` (SQLite: `STRING`)
`Enum`	`str`	`ENUM` or `VARCHAR`
`Float`	`float` or `Decimal`	`FLOAT` or `REAL`
`Integer`	`int`	`INTEGER`
`Interval`	`datetime.timedelta`	`INTERVAL` or `DATE from epoch`

Como usar SQLAlchemy

Fatec
Rio Claro

`Interval`	`datetime.timedelta`	`INTERVAL` or `DATE from epoch`
`LargeBinary`	`byte`	`BLOB` or `BYTEA`
`Numeric`	`decimal.Decimal`	`NUMERIC` or `DECIMAL`
`Unicode`	`unicode`	`UNICODE` or `VARCHAR`
`Text`	`str`	`CLOB` or `TEXT`
`Time`	`datetime.time`	`DATETIME`

Metadados

Fatec
Rio Claro

Metadados são usados para unir a estrutura do banco de dados, permitindo acesso rápido dentro do SQLAlchemy. Geralmente, é útil pensar em metadados como uma espécie de catálogo de objetos Table com informações opcionais sobre o mecanismo e a conexão.

Essas tabelas podem ser acessadas por meio de um dicionário, MetaData.tables.

Operações de leitura são thread-safe; no entanto, a construção de tabelas não é totalmente thread-safe.

Os metadados precisam ser importados e inicializados antes que os objetos possam ser vinculados a eles.

Table

Fatec
Rio Claro

Objetos de Table são inicializados no SQLAlchemy Core em um objeto MetaData fornecido, chamando o construtor Table com o nome da tabela e os metadados; quaisquer argumentos adicionais são considerados objetos de coluna.

Há também alguns argumentos de palavras-chave adicionais que habilitam recursos que discutiremos mais tarde. Objetos de coluna representam cada campo na tabela.

As colunas são construídas chamando Column com um nome, tipo e, em seguida, argumentos que representam quaisquer construções e restrições SQL adicionais.

Columns

Fatec
Rio Claro

As colunas definem os campos existentes em nossas tabelas e fornecem o principal meio pelo qual definimos outras restrições por meio de seus argumentos de palavras-chave.

Diferentes tipos de colunas têm diferentes argumentos primários.

Por exemplo, colunas do tipo String têm comprimento como argumento primário, enquanto números com um componente fracionário terão precisão e comprimento.

A maioria dos outros tipos não tem argumentos primários.

Prática

Fatec
Rio Claro

Observe os códigos

pg_create_table.py

pg_update_user.py

pg_remove_user.py

Chaves e Restrições

Fatec
Rio Claro

Chaves e restrições são usadas como forma de garantir que nossos dados atendam a determinados requisitos antes de serem armazenados no banco de dados.

Os objetos que representam chaves mais comuns podem ser importados:

from sqlalchemy import PrimaryKeyConstraint

from sqlalchemy import UniqueConstraint

from sqlalchemy import CheckConstraint

O tipo de chave mais comum é a chave primária, que é usada como identificador exclusivo para cada registro em uma tabela de banco de dados e é usada para garantir um relacionamento adequado entre dois dados relacionados em tabelas diferentes.

Índices

Fatec
Rio Claro

Índices são usados para acelerar pesquisas de valores de campo. No código **pg_create_table.py**, criamos um índice na coluna *nome* porque sabemos que pesquisaremos por ela com frequência.

Também podemos definir um índice usando um tipo de construção explícito.

```
from sqlalchemy import Index
Index('ix_cpf', 'cpf')
```

Relacionamentos e ForeignKeyConstraints

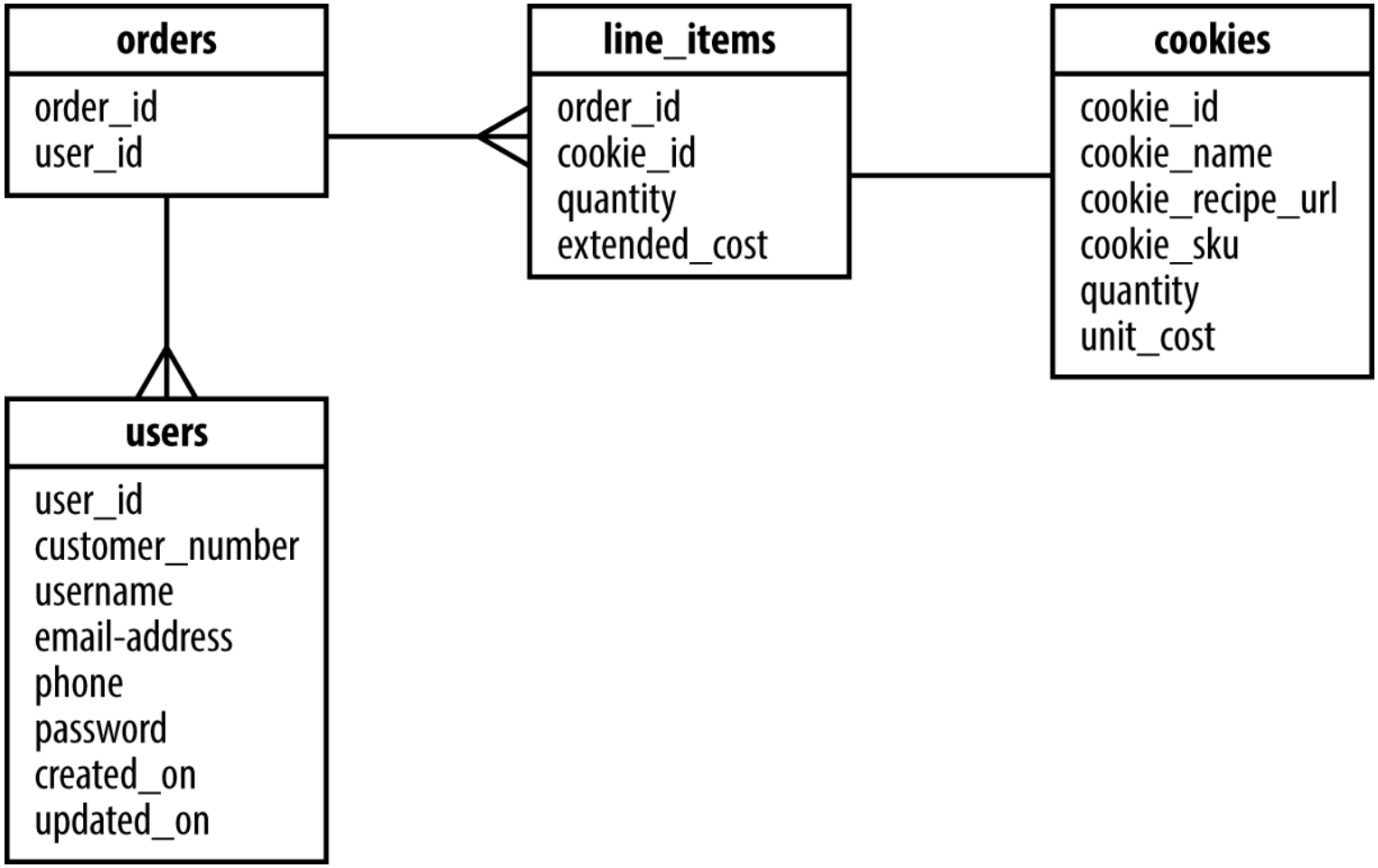
Fatec Rio Claro

Agora que temos uma tabela com colunas com todas as restrições e índices corretos, vamos ver como criamos relacionamentos entre tabelas.

Precisamos de uma maneira de rastrear pedidos, incluindo itens de linha que representam cada cookie e a quantidade pedida.

Relacionamentos e ForeignKeyConstraints

Fatec
Rio Claro



Relacionamentos e ForeignKeyConstraints

Fatec Rio Claro

Agora que temos uma tabela com colunas com todas as restrições e índices corretos, vamos ver como criamos relacionamentos entre tabelas.

Precisamos de uma maneira de rastrear pedidos, incluindo itens de linha que representam cada cookie e a quantidade pedida.

Prática

Fatec
Rio Claro

Observe os códigos

pg_ForeignKeyConstraints.py

pg_ForeignKeyConstraints_remove.py

SQL Alchemy
ORM

SQLAlchemy ORM

Fatec
Rio Claro

No SQLAlchemy Core, criamos um contêiner de metadados e, em seguida, declaramos um objeto Table associado a esses metadados.

No ORM SQLAlchemy, definiremos uma classe que herda de uma classe base especial chamada declarative_base.

A declarative_base combina um contêiner de metadados e um mapeador que mapeia nossa classe para uma tabela do banco de dados.

Ele também mapeia instâncias da classe para registros nessa tabela, caso tenham sido salvos.

SQLAlchemy ORM

Fatec
Rio Claro

Uma classe adequada para uso com o ORM deve fazer quatro coisas:

- Herdar do objeto declarative_base.
- Conter __tablename__, que é o nome da tabela a ser usada no banco de dados.
- Conter um ou mais atributos que sejam objetos Column.
- Garantir que um ou mais atributos constituam uma chave primária.

SQLAlchemy ORM
Sessão

Fatec
Rio Claro

A sessão é a forma como o SQLAlchemy ORM interage com o banco de dados.

Ela encapsula uma conexão com o banco de dados por meio de um mecanismo e fornece um mapa de identidade para objetos que você carrega por meio da sessão ou associa a ela.

O mapa de identidade é uma estrutura de dados semelhante a um cache que contém uma lista exclusiva de objetos, determinada pela tabela e pela chave primária do objeto. Uma sessão também encapsula uma transação, e essa transação permanecerá aberta até que a sessão seja confirmada ou revertida, de forma muito semelhante ao processo descrito em "Transações".

SQLAlchemy ORM
Sessão

Fatec
Rio Claro

Para criar uma nova sessão, o SQLAlchemy fornece a classe sessionmaker para garantir que as sessões possam ser criadas com os mesmos parâmetros em toda a aplicação.

Isso é feito criando uma classe Session que foi configurada de acordo com os argumentos passados para a fábrica sessionmaker.

A factory sessionmaker deve ser usada apenas uma vez no escopo global da sua aplicação e tratada como uma definição de configuração.

Prática

Fatec
Rio Claro

Observe os códigos

sqlite_ORM.py

pg_ORM1.py

pg_ORM2.py

SQLAlchemy ORM
Sessão

Fatec
Rio Claro

Um flush é como um commit; no entanto, não executa um commit no banco de dados e encerra a transação.

Por isso, as instâncias dcc e mol ainda estão conectadas à sessão e podem ser usadas para executar tarefas adicionais no banco de dados sem acionar consultas adicionais.

Também emitimos a instrução session.flush() uma vez, mesmo tendo adicionado vários registros ao banco de dados.

Isso, na verdade, resulta no envio de duas instruções insert para o banco de dados dentro de uma única transação.

Dica

Se você estiver inserindo vários registros e não precisar acessar os relacionamentos ou a chave primária inserida, use **bulk_save_objects** ou seus métodos relacionados.

Isso é especialmente verdadeiro se você estiver ingerindo dados de uma fonte de dados externa, como um CSV ou um documento JSON grande com matrizes aninhadas.

SQLAlchemy ORM Query

Fatec Rio Claro

Para começar a construir uma consulta, usamos o método query() na instância da sessão.

```
for u in session.query(User).all():
    print(u)
for c in session.query(Cookie).limit(5):
    print(c)
for o in session.query(Order).limit(5):
    print(o)
for l in session.query(LineItems).limit(5):
    print(l)
```

SQLAlchemy ORM Query

Fatec
Rio Claro

Recurso	Método SQLAlchemy	Equivalente SQL
Ordenar	`.order_by()`	`ORDER BY`
Limitar	`.limit()`	`LIMIT`
Filtrar	`.filter()`	`WHERE`
Operadores	`==, !=, >, <, .like(), .in_()`	`=, <>, >, <, LIKE, IN`
Conjunções	`and_(), or_(), not_()`	`AND, OR, NOT`

Fatec
Rio Claro

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Prática

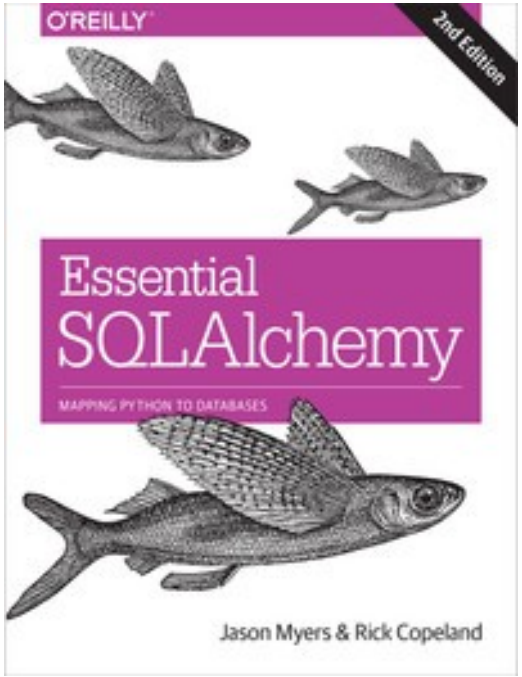
Fatec
Rio Claro

No arquivo usuarios.db há um banco SQLite com 10000 pessoas.

Desenvolva um script que apresente apenas pessoas com 18 anos de idade.

Bibliografia

Myers, Jason; Copeland, Rick. Essential SQLAlchemy: Mapping Python to Databases. 2. ed. Sebastopol, CA: O'Reilly Media, 2015. 208 p. ISBN 978-1491916469.



Bibliografia

VIESCAS, John L.; STEELE, Douglas J.; CLOTHIER, Ben G. Effective SQL: 61 Specific Ways to Write Better SQL. 1. ed. Boston: Addison-Wesley Professional, 2016.